Processing Humanities Multimedia

Garth Evans
Josh Romphf
Welcome to DHSI 2018!

Thanks for joining the DHSI community!

In this booklet, you will find essential course materials prefaced by some useful information about getting settled initially at UVic, finding your way around, getting logged in to our network (after you’ve registered the day before our courses begin), and so on.

Given our community’s focus on things computational, it will be a surprise to no one that we might expect additional information online for some of the classes - your instructors will let you know - or that the most current version of all DHSI-related information may be found on our website at dhsi.org.

To access the DHSI wifi network, simply go into your wireless settings and connect to the “DHSI” network and enter the password “dhsi2018”.

And please don’t hesitate to be in touch with us at institut@uvic.ca or via Twitter at @AlyssaA_DHSI or @DHInstitute if we can be of any help ....
Regional Map of Greater Victoria

Average Frequency
- **Regional Route**: 15–60 minute service with limited stops
- **Frequent Route**: 15 minute or better service, 7am-7pm, Mon-Fri
- **Local Route**: 20–120 minute service

Legend
- Direction of Travel
- Route Name
- Transit Exchange
- UVic X
- Park & Ride Lot (no overnight parking)
- Major Stop

![Map of Greater Victoria with marked stops and routes](image-url)
The 2018 schedule is just about ready! A very few things to confirm, add, etc, but this is the place to be to find out what is happening when / where ...

Sunday, 3 June 2018 [DHSI Registration + Suggested Outings]

If you’re here a day or two before we begin, or staying a day or two afterwards, here are a few ideas of things you might consider doing ....

▼ Suggested Outing 1, Botanical Beach (self-organised; car needed)

A self-guided visit to the wet, wild west coast tidal shelf (and historically-significant former research site) at Botanical Beach: we recommend departing early (around 8.00 am) to catch low tide for a better view of the wonderful undersea life! Consider bringing a packed lunch to nibble-on while looking at the crashing waves when there, and then have an afternoon drink enjoying the view from the deck of the Port Renfrew Hotel.

▼ Suggested Outing 2, Butchart Gardens (self-organised)

A shorter journey to the resplendently beautiful Butchart Gardens and, if you like, followed by (ahem) a few minutes at the nearby Church and State Winery, in the Saanich Peninsula. About an hour there by public bus from UVic, or 30 minutes by car.

▼ Suggested Outing 3, Salt Spring Island (self-organised; a full day, car/bus + ferry combo)

Why not take a day to explore and celebrate the funky, laid back, Canadian gulf island lifestyle on Salt Spring Island. Ferry departs regularly from the Schwartz Bay ferry terminal, which is about one hour by bus / 30 minutes by car from UVic. You may decide to stay on forever ....

▼ Suggested Outing 4, Paddling Victoria’s Inner Harbour (self-organised)

A shorter time, seeing Victoria’s beautiful city centre from the waterways that initially inspired its foundation. A great choice if the day is sunny and warm. Canoes, kayaks, and paddle boards are readily rented from Ocean River Adventures and conveniently launched from right behind the store. Very chill.

And more!

Self-organised High Tea at the Empress Hotel, scooter rentals, visit to the Royal BC Museum, darts at Christies Carriage House, a hangry breakfast at a local diner, whale watching, kayaking, brew pub sampling (at Spinnaker’s, Swans, Moon Under Water, and beyond!), paddle-boarding, a tour of used bookstores, and more have also been suggested!

Psst: Some Suggested Outings

9:00 to 4:00

▼ Early Class Meeting: 4. [Foundations] DH For Department Chairs and Deans (Hickman 120, Classroom)

Further details are available from instructors in mid May to those registered in the class. Registration materials will be available in the classroom.

3:00 to 5:00

DHSI Registration (MacLaurin Building, Room A100)

After registration, many will wander to Cadboro Bay and the pub at Smuggler’s Cove OR the other direction to Shelbourne Plaza and Maude Hunter’s Pub OR even into the city for a nice meal.

Monday, 4 June 2018

Your hosts for the week are Alyssa Arbuckle, Ray Siemens, and Dan Sondheim.

7:45 to 8:15

Last-minute Registration (MacLaurin Building, Room A100)

8:30 to 10:00

Welcome, Orientation, and Instructor Overview (MacLaurin A144)
3. [Foundations] Making Choices About Your Data (MacLaurin D109, Classroom)
4. [Foundations] DH For Department Chairs and Deans (Hickman 120, Classroom)
5. [Foundations] Introduction to Javascript and Data Visualization (Clearihue D132, Classroom)
6. [Foundations] Introduction to Computation for Literary Criticism (Clearihue A105, Lab)
7. Out-of-the-Box Text Analysis for the Digital Humanities (Human and Social Development A160, Lab)
8. Sounds and Digital Humanities (MacLaurin D111, Classroom)
9. 10:15 to Noon
   - Digital Humanities Pedagogy: Integration in the Curriculum (MacLaurin D016, Classroom)
   - Text Processing - Techniques & Traditions (McPherson Library A003, Classroom)
10. 12:15 to 1:15
    - 12:15 to 1:15
      - Lunch break / Unconference Coordination Session (MacLaurin A144)
        - (Grab a sandwich and come on down!)
      - Undergraduate Meet-up, Brown-Bag (details via email)
    - 1:30 to 4:00
      - Classes in Session
        - Institute Panel: Perspectives on DH (or, #myDHis …)
          - Chair: Alyssa Arbuckle (U Victoria) (MacLaurin A144)
          - Milena Radzikowska (Mt Royal C): “Release the Kraken: Story-Driven Prototyping for the Digital Humanities.”
            - Abstract: I have spent the last 15 years of my career designing text analysis tools for use by humanities scholars. In this brief presentation, I propose to share a concept-based approach to interface design for DH.
          - Emily Murphy (U Victoria): “#MyDHis Edgy.”
            - Abstract: I will build upon—or, possibly, perform a misprision of—a tweet by Polina Vinogradova; “#myDHis messy, dusty, edgy, and radically inclusive!” Vinogradova evokes the mess and dust of the archives, the edges that connect nodes of a network, and the political impetus to think of cultural history and community together. I argue that these aspects of DH have a renewed importance as we head into a moment of feminist historiography.
          - Margaret Konkol (Old Dominion U): “Prototyping Mina Loy’s Alphabet with a 3D Printer.”
            - Abstract: This talk discusses the interpretive and methodological implications of using 3D printing technologies to prototype the archival diagrams of a proposed but never constructed plastic segmental alphabet letter kit—a game designed by modernist poet Mina Loy for F.A.O Schwarz. Although intended as a toy for young children, “The Alphabet that Builds Itself,” as a work of “object typography” articulates a theory of language as kinetic, geometric, recombinant, and open to mutation. Alphabetic segments extend into the x, y, and z coordinates in exponential iterations and conjoin with magnets. Combining elements of contemporaneous typefaces like Futura and Gill Sans, which represented modernity’s functional ideals and democratic principles of simplicity, these recombinant letters represent, as this talk argues, Loy’s unpublished modernist poem, an articulation of Loy’s concept of language as a physical fact in which substance, not just form, is semantic.
          - Lee Zickel (Case Western Reserve U): “Comfortably Trepid.”
            - Abstract: #myDHis found outside the well-established, DH-friendly institutions, at an institution that is devoted predominantly to Medicine and Engineering. I, with increasing frequency other DH practitioners and instructors, am not positioned in a DH Lab or Humanities Center, but in ITS. Part teacher, part technologist, part translator, I will briefly discuss my work supporting humanists and social scientists, particularly those who are new to or less comfortable with computational methodologies.
          - Dorothy Kim (Vassar C): “#MyDHis Antifascist.”
            - Abstract: I’ve spent a lot of time in the last 12 months thinking about fascism, digital humanities, its long histories, and what it means to do DH work that centers social justice particularly in this global rise of late fascism. I will speak briefly about DH’s history, including the medieval history related to Busa but how that history really connects to data systems that created the Holocaust and also participated in the Cold War nuclear military complex.

4.10 to 5:00
   - 4:10 to 5:00
Randa El Khatib (U Victoria): "Learning from the Iterative Process."
Abstract: #MyDHis Iterative. In addition to the improvements that come with iterative projects, the iterative process itself is a fruitful area for scholarly inquiry. Within this iterative context, the various teams that I work with and I have been reflecting on and rethinking central DH practices, such as what it means to collaborate, prototype, remix, and implement DH values in our work. In this talk, I will present the various lessons learnt along the way.

Sarah Melton (Boston C): "#MyDHis...People."
Abstract: Taking seriously Miriam Posner’s exhortation to “commit to DH people, not DH projects,” I invite us to reflect on how people are the core of DH. In this brief talk, I will explore the intersections between DH, labor, and infrastructure.

**Tuesday, 5 June 2018**

5:00 to 6:00
Opening Reception (University Club)
We are grateful to Gale Cengage for its sponsorship.

9:00 to Noon
Classes in Session

12:15 to 1:15
Lunch break / Unconference
"Mystery" Lunches
- DHSI Lunchtime Workshop Session (click for workshop details and free registration for DHSI participants)
  - 73. Introduction to ORCID (Digital Scholarship Commons, Classroom).

1:30 to 4:00
Classes in Session

- DHSI Colloquium Lightning Talk Session 1 (MacLaurin A144)
  Chair: James O'Sullivan
  - New Modes of DH and Archival Skills Acquisition in a Graduate Public History Course. Paulina Rousseau (Ryerson U)
  - Walking a Transect: Exploring a Soundscape. John Barber (Washington State U)
  - Centering the Edge Case: Designing Services for Humanities Data Research. Grace Afsari-Mamagani (New York U)
  - Orwellian Vocabulary and the 21st-Century Politics. Ilgin Kizilgunesler (U Manitoba)
  - Making Open Data from a Gray Archive. Sara Palmer (Emory U)

6:00 to 8:00
DHSI Newcomer's Beer-B-Q (Felicitas, Student Union Building)

**Wednesday, 6 June 2018**

9:00 to Noon
Classes in Session

12:15 to 1:15
Lunch break / Unconference
"Mystery" Lunches
- Brown Bag Lecture: Alexandra Branzan Albu (U Victoria): "Visual Recognition of Symbolic and Natural Patterns" (Digital Scholarship Commons, 3rd Floor McPherson Library)
  Abstract: Image-based object recognition is a visual pattern recognition problem; one may characterize visual patterns as either symbolic or natural. Symbolic patterns evolved for human communication; they include but are not limited to text, forms, tables, graphics, engineering drawings etc. Symbolic patterns vary widely in terms of size, style, language, alphabet and fonts; however, literate humans can easily compensate for this variability and instantly recognize most symbolic patterns. On the other hand, natural patterns characterize images of physical structures; they often lack the intrinsic discriminability and structure of symbolic patterns, and vary widely in terms of pose, perspective, and lighting.
  This lecture will explore similarities and differences in approaches designed for recognizing visual and symbolic patterns, and will address the following questions via examples.
  - What are the distinctive characteristics of natural patterns? What dimensions of variability can we infer?
  - What are the distinctive characteristics of symbolic patterns? What dimensions of variability can we infer?

Alexandra Branzan Albu is an Associate Professor with the Department of Electrical and Computer Engineering and cross-listed with Computer Science. Her research interests are related to image analysis, computer vision, and visual computing. She is actively pursuing outreach activities dedicated to increasing the women's presence in electrical engineering and computer science.

1:30 to 4:00
Classes in Session
### Thursday, 7 June 2018

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 AM</td>
<td>DHSI Colloquium Lightning Talk Session 2 <em>(MacLaurin A144)</em></td>
</tr>
<tr>
<td>8:30 AM</td>
<td>Chair: James O'Sullivan</td>
</tr>
</tbody>
</table>
| 4:15 PM    | • Defining a Taxonomy of Abandonment for Online Digital Humanities Projects. Luis Meneses (Electronic Textual Cultures Lab, U Victoria) and Jonathan Martin (King's College London)  
• The Stories We Tell: Representing Gay and Lesbian History through Digital Technologies in the LGLC Project. Nadine Boulay (Simon Fraser University) and Ewan Matthews (Ryerson U)  
• Italian Paleography in the Digital Domain. Isabella Magni (Newberry Library)  
• Digital Humanities, A Question of Ethics. Negar Basiri (Louisiana State U)  
• Writing Poetry in High School. Guadalupe Echegoyen (National Autonomous U Mexico) |
| 6:00 PM    | • "Half Way There!" [An Informal, Self-Organized Birds of a Feather Get-Together] *(Felicitas, Student Union Building)*  
Bring your DHSI nametag and enjoy your first tipple on us! |
| 9:00 AM    | Classes in Session                                                     |
| 12:15 PM   | Lunch break / Unconference "Mystery" Lunches                           |
| 1:00 PM    | • DHSI Colloquium Lightning Talk Session 3 *(MacLaurin A144)*          |
| 4:15 PM    | Chair: James O'Sullivan                                                 |
| 6:00 PM    | • Documenting Deportation: A Collaborative Digital Collection. Paulina Rousseau (Ryerson U)  
• Unleashing the Power of Texts as Networks: Visualizing the Scholastic Commentaries and Texts Archive. Jeffrey Witt (Loyola U Maryland) and Drew Winget (Stanford U)  
• #haunteDH: Punching holes in the International Busa Machine Narrative. Arun Jacob (McMaster U)  
• Text in World: Computational Analysis of Trauma in Genocide Narratives. Nanditha Narayanamoorthy (U York) and Krish Perumal (U Toronto) |
| 7:30 PM    | (Groovy?) Movie Night *(MacLaurin A144)*                               |

### Friday, 8 June 2018 [DHSI; DLFxDHSI Opening]

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00 AM</td>
<td>DHSI Classes in Session</td>
</tr>
<tr>
<td>12:15 PM</td>
<td>DHSI Lunch Reception / Course E-Exhibits <em>(MacLaurin A100)</em></td>
</tr>
<tr>
<td>1:00 PM</td>
<td>DLFxDHSI Registration <em>(MacLaurin A100)</em></td>
</tr>
<tr>
<td>1:30 PM</td>
<td>[DHSI] Remarks, A Week in Review <em>(MacLaurin A144)</em></td>
</tr>
</tbody>
</table>
| 2:00 PM    | • Joint Institute Lecture (DHSI and DLFxDHSI): Bethany Nowviskie (CLIR DLF and U Virginia): "Reconstitute the World: Machine-reading Archives of Mass Extinction"  
Chair: Lisa Goddard (U Victoria) *(MacLaurin A144)*  
Abstract: The basic constitution of our digital collections becomes vastly more important in the face of two understandings: first, that archives of modernity are archives of the sixth great mass extinction of life on our planet; and next, that we no longer steward cultural heritage for human readers alone. In the same way that we people are shaped by what we read, hear, and see, the machine readers that follow us into and perhaps beyond the Anthropocene have begun to learn from "unsupervised" encounters with our digital libraries. What will we preserve for the living generations and artificial intelligences that will come? What do we neglect, or even choose to extinguish? And from an elegiac archive, a library of endings, can we create forward-looking, speculative collections--collections from which to deep-dream new futures? The most extra/ordinary power we possess is the power to make poetry from records of the past. Could it be called on, one day, to reconstitute the world? |
3:30 to 5:00

Joint Reception: DHSI and DLFxDHSI (University Club)
DLFxDHSI Poster/Demo Session

- DHSI Colloquium Poster/Demo Session
  - Mediers as a Colonialist Artifact in Menzies’ Journal. Paula Johanson (U Victoria)
  - Camp Edit: the Institute for the Editing of Historical Documents. Nikolaus Wasmoen (Association for Documentary Editing, U Buffalo), Jennifer Stertzter (Association for Documentary Editing, U Virginia), and Cathy Moran Hao (Association for Documentary Editing, Ramapo C)
  - A Digital Archaeology of Life in Cleveland’s Depression-Era Slums. Charlie Harper (Case Western Reserve U) and Jared Bendis (Case Western Reserve U)
  - Feminist Pest Control: controlling and not controlling nonhuman pests. Lindsay Garcia (C of William and Mary)
  - Legends of the Buddhist Saints. Jonathan S. Walters (Whitman C) and Dana Johnson (Freelance Web Developer)
  - Accessibility in Digital Environments Via TEI-Encoded Uncontracted Braille. Gia Alexander (Texas A&M U)
  - Translation3point0: Why Literary Translation Data Matters. Katie King (U Washington)
  - PoéticaSonora: A Digital Audio Repository Prototype for Latin American Sound Art and Poetry. Aurelio Meza (Concordia U)
  - Beauty and the Book: Pre-Raphaelite Artistic Practice Contained. Josie Greenhill (U Victoria)
  - Poetic Procedures/Digital Deformances. Corey Sparks (California State U, Chico)
  - Miranda, the Folger Shakespeare Library’s new Digital Asset Platform. Meaghan Brown (Folger Shakespeare Library)
  - Living Song Project. Quinn Patrick Ankrum (U Cincinnati) and Elizabeth Avery (U Oklahoma)
  - Digital Frankenstein Variorum. Rikk Mulligan (Carnegie Mellon U)

Saturday, 9 June 2018 [DLFxDHSI + DHSI Conference and Colloquium]

8:30 to 9:00

DLFxDHSI Registration (MacLaurin A100)

9:00 to 5:30

DLFxDHSI UnConference Sessions

- DHSI All Day Workshop Session (click for workshop details and free registration for DHSI participants)

9:00 to 4:00

53. Building Your Academic Digital Identity (MacLaurin D105, Classroom)

- DHSI Colloquium Day Conference (MacLaurin A144)

Welcome

People I: Documenting Online Lives. Chair: Molly Nebiolo (University of New York)
  - Examining Gendered Harassment Online and in Silicon Valley. Andrea Flores (Utica College)
  - This is Just to Say I Have <X> the <Y> in your <Z>: Modernist Memes in an Era of Public Apology. Shawna Ross (Texas A&M University)

Break

People II: Documenting Lives Online. Chair: Dheepa Sundaram (College of Wooster)
  - Youtube Yoga and Ritual on Demand: The Virtual Economics of Hindu Soteriology. Dheepa Sundaram (College of Wooster)
  - The Resemblage Project: Creativity and Digital Health Humanities in Canada. Andrea Charise (University of Toronto) and Stefan Krecsy (University of Toronto)

Lunch

Projects I: Building and Analyzing. Chair: Yannis Rammos (New York University)
  - Building the ARTECHNE Database: New directions in Digital Art History. Marieke Hendriksen (Old Dominion University)
  - The Ineffective Inquisition: The Holy Office’s Sphere of Influence in Early Modern New Spain. Kira Homo (Pennsylvania State University)

Break

Projects II: Mapping and Visualizing. Chair: Innocent Opara (Quimet Institute)
  - Mapping Sarah Sophia Bank’s Numismatic Collection. Erica Hayes (North Carolina State University) and Kacie Wills (University of California, Riverside)
  - Text Mining and Visualizing 18th Century American Correspondence. Ashley Sanders Garcia (University of California, Los Angeles)

Break

Practices: Digital Scholarship on Campus and in the Classroom. Chair: Alyssa Arhuckle (University of Victoria)
## Sunday, 10 June 2018 [SINM + DHSI Registration, Workshops]

### 8:30 to 9:00

**Symposium on Indigenous New Media Registration** *(MacLaurin A100)*

### 9:00 to 5:00

**DHSI Registration** *(MacLaurin A100)*

#### ▼ SINM Sessions

- 9:00 to 4:00
  - 63. Symposium on Indigenous New Media: Reading Group *(Hickman 105, Classroom)*
  - 72. Symposium on Indigenous New Media: Indigitization *(Hickman 120, Classroom)*
  
  Full details here

#### ▼ DHSI All Day Workshop Sessions *(click for workshop details and free registration for DHSI participants)*

- 9:00 to 4:00
  - 53. Building Your Academic Digital Identity *(MacLaurin D105, Classroom)*
  - 54. An Introduction to the Archaeology of 1980s Computing *(MacLaurin D114, Classroom)*

#### ▼ DHSI AM Workshop Sessions *(click for workshop details and free registration for DHSI participants)*

- 9:00 to Noon
  - 55. Regular Expressions *(MacLaurin D111, Classroom)*
  - 56. 3D Visualization for the Humanities *(MacLaurin D101, Classroom)*
  - 58. DH Fieldwork Methods *(MacLaurin D016, Classroom)*
  - 60. Pedagogy of the Digitally Oppressed: Inculcating De-/Anti-/Post-Colonial Digital Humanities *(MacLaurin D107, Classroom)*
  - 61. Introduction to #GraphPoem. Digital Tools for Poetry Computational Analysis and Graph Theory Apps in Poetry *(MacLaurin D101, Classroom)*
  - 62. Creating a CV for Digital Humanities Makers *(MacLaurin D115, Classroom)*

#### ▼ DHSI PM Workshop Sessions *(click for workshop details and free registration for DHSI participants)*

- 1:00 to 4:00
  - 64. Agent-Based Modelling in the Humanities *(MacLaurin D111, Classroom)*
  - 65. Unleash Linux on MacOS *(MacLaurin D010, Classroom)*
  - 66. DHSI Knits: History of Textiles and Technology *(MacLaurin D016, Classroom)*
  - 67. Crowdsourcing as a Tool for Research and Public Engagement *(MacLaurin D109, Classroom)*
  - 69. Web Annotation as Critical Humanities Practice *(MacLaurin D103, Classroom)*
  - 70. Dynamic Ontologies for the Humanities *(MacLaurin D107, Classroom)*
  - 71. Social Media Research in the Humanities *(MacLaurin D101, Classroom)*

### 4:10 to 5:00

**Joint Institute Lecture (DHSI and SINM):**

David Gaertner (U British Columbia): "A Landless Territory?: CyberPowWow and the Politics of Indigenous New Media."

Chair: Deanna Reder (Simon Fraser U)

*(MacLaurin A144)*

Abstract: Following the 1997 launch of Skawennati’s (Mohawk) CyberPowWow, digital space has become a vital new territory for the resurgence of Indigenous storytelling and cultural practice: "We have signed a new treaty," Cree artist Archer Pechawis wrote of this period, "and it is good. We have the right to hunt, fish, dance and make art at www.CyberPowWow.net, .org and .com for as long as the grass grows and the rivers flow." This talk will critically explore the theoretical, cultural, political-economic, and gendered dynamics underwriting the histories and futures of Indigenous new media. Particular attention will be given in examining the ways in which new media and digital storytelling connect to and support key issues in the field of Indigenous studies, such as sovereignty, self-determination, decolonization, and land rights.

### After the day, many will wander to Cadboro Bay and the pub at Smuggler's Cove OR the other direction to Shelbourne Plaza and Maude Hunter’s Pub OR even into the city for a bite to eat.

## Monday, 11 June 2018 [DHSI + SINM]
Your hosts for the week are Ray Siemens and Dan Sondheim.

7:45 to 8:15  DHSI Last-minute Registration (MacLaurin A100)

8:30 to 10:00  DHSI Welcome, Orientation, and Instructor Overview (MacLaurin A144)

9:00 to 4:00  SINM Sessions

- DHSI Classes in Session (click for details and locations)
  - 29. [Foundations] Models for DH at Liberal Arts Colleges (& 4 yr Institutions) (MacLaurin D109, Classroom)
  - 32. Stylometry with R: Computer-Assisted Analysis of Literary Texts (Clearihue A102, Lab)
  - 33. Digital Storytelling (MacLaurin D111, Classroom)
  - 34. Text Mapping as Modelling (Clearihue D131, Classroom)
  - 35. Geographical Information Systems in the Digital Humanities (Clearihue A105, Lab)
  - 36. Open Access and Open Social Scholarship (MacLaurin D114, Classroom)
  - 37. Introduction to Machine Learning in the Digital Humanities (Cornett A229, Classroom)
  - 38. Queer Digital Humanities: Intersections, Interrogations, Iterations (MacLaurin D110, Classroom)
  - 39. Using Fedora Commons / Islandora (Human and Social Development A160, Lab)
  - 40. Documenting Born Digital Creative and Scholarly Works for Access and Preservation (MacLaurin D115, Classroom)
  - 41. Games for Digital Humanists (MacLaurin D016, Classroom & Human and Social Development A170, Lab)
  - 42. XPath for Document Archeology and Project Management (Cornett A128, Classroom)
  - 43. Surveillance and the Digital Humanities (MacLaurin D103, Classroom)
  - 44. Text Analysis with Python and the Natural Language ToolKit (Clearihue A103, Lab)
  - 46. Surveillance and the Digital Humanities (MacLaurin D103, Classroom)
  - 47. Text Analysis with Python and the Natural Language ToolKit (Clearihue A103, Lab)
  - 48. Information Security for Digital Researchers (Clearihue D130, Classroom)
  - 49. Wrangling Big Data for DH (Human and Social Development A150, Lab)
  - 50. Accessibility & Digital Environments (MacLaurin D101, Classroom)
  - 51. Critical Pedagogy and Digital Praxis in the Humanities (MacLaurin D105, Classroom)
  - 52. Drupal for Digital Humanities Projects (MacLaurin D107, Classroom)

10:15 to Noon  DHSI Classes in Session

- 29. [Foundations] Models for DH at Liberal Arts Colleges (& 4 yr Institutions) (MacLaurin D109, Classroom)
- 32. Stylometry with R: Computer-Assisted Analysis of Literary Texts (Clearihue A102, Lab)
- 33. Digital Storytelling (MacLaurin D111, Classroom)
- 34. Text Mapping as Modelling (Clearihue D131, Classroom)
- 35. Geographical Information Systems in the Digital Humanities (Clearihue A105, Lab)
- 36. Open Access and Open Social Scholarship (MacLaurin D114, Classroom)
- 37. Introduction to Machine Learning in the Digital Humanities (Cornett A229, Classroom)
- 38. Queer Digital Humanities: Intersections, Interrogations, Iterations (MacLaurin D110, Classroom)
- 39. Using Fedora Commons / Islandora (Human and Social Development A160, Lab)
- 40. Documenting Born Digital Creative and Scholarly Works for Access and Preservation (MacLaurin D115, Classroom)
- 41. Games for Digital Humanists (MacLaurin D016, Classroom & Human and Social Development A170, Lab)
- 42. XPath for Document Archeology and Project Management (Cornett A128, Classroom)
- 43. Surveillance and the Digital Humanities (MacLaurin D103, Classroom)
- 44. Text Analysis with Python and the Natural Language ToolKit (Clearihue A103, Lab)
- 46. Surveillance and the Digital Humanities (MacLaurin D103, Classroom)
- 47. Text Analysis with Python and the Natural Language ToolKit (Clearihue A103, Lab)
- 48. Information Security for Digital Researchers (Clearihue D130, Classroom)
- 49. Wrangling Big Data for DH (Human and Social Development A150, Lab)
- 50. Accessibility & Digital Environments (MacLaurin D101, Classroom)
- 51. Critical Pedagogy and Digital Praxis in the Humanities (MacLaurin D105, Classroom)
- 52. Drupal for Digital Humanities Projects (MacLaurin D107, Classroom)
- 73. Introduction to ORCID (Digital Scholarship Commons, Classroom).

12:15 to 1:15  Lunch break / Unconference Coordination Session (MacLaurin A144)

1:30 to 4:00  DHSI Classes in Session

- Joint Institute Lecture (DHSI and SINM): Jordan Abel (Simon Fraser U): "Indigeneity, Conceptualism, and the Borders of DH.” Chair: Michelle Brown (U Hawaii) (MacLaurin A144)

Abstract: This talk brings together digital humanities discourses in computational textual analysis and Indigenous Literary Studies to analyze a corpus comprised of every book of Indigenous poetry published in Canada, extending from Pauline Johnson's 1895 book The White Wampum to Marilyn Dumont's 2015 book The Pemmican Eaters. While the main goal of this research project initially centered on the topic modeling of a corpus of Indigenous poetry, the project also addresses the systemic barriers that have prevented such work gaining traction, and likewise attempts to address the specific challenges that Indigenous writing (and in particular Indigenous poetry) present to current Digital Humanities methodologies.

5:00 to 6:00  Joint Reception: DHSI and SINM (University Club)

Tuesday, 12 June 2018

9:00 to Noon  Classes in Session

- Joint Institute Lecture (DHSI and SINM): Jordan Abel (Simon Fraser U): "Indigeneity, Conceptualism, and the Borders of DH.” Chair: Michelle Brown (U Hawaii) (MacLaurin A144)

Abstract: This talk brings together digital humanities discourses in computational textual analysis and Indigenous Literary Studies to analyze a corpus comprised of every book of Indigenous poetry published in Canada, extending from Pauline Johnson's 1895 book The White Wampum to Marilyn Dumont's 2015 book The Pemmican Eaters. While the main goal of this research project initially centered on the topic modeling of a corpus of Indigenous poetry, the project also addresses the systemic barriers that have prevented such work gaining traction, and likewise attempts to address the specific challenges that Indigenous writing (and in particular Indigenous poetry) present to current Digital Humanities methodologies.

12:15 to 1:15  Lunch break / Unconference Coordination Session (MacLaurin A144)

"Mystery" Lunches

- 73. Introduction to ORCID (Digital Scholarship Commons, Classroom).
Wednesday, 13 June 2018

9:00 to Noon
Classes in Session

12:15 to 1:15
Lunch break / Unconference
"Mystery" Lunches

1:30 to 4:00
Classes in Session

4:15 to 5:15
DHSI Colloquium Lightning Talk Session 5 (MacLaurin A144)
Chair: Lindsey Seatter
- Faraway, so close: Has the political environment really changed in Ecuador?. Luis Meneses (Electronic Textual Cultures Lab, U Victoria)
- Re-mixing Melville's Reading: Text Analysis of Marginalia with R and XSLT. Christopher Ohge (U London, School of Advanced Study) and Steven Olsen-Smith (Boise State U)
- Developing Interactive and Open-Source OER: Inquiry-Based Music Theory. Evan Williamson (U Idaho)
- Spatial Humanities and the Web of Everywhere. Ken Cooper (SUNY Geneseo)

6:00 to 7:00
"Half Way There (yet again)!" [An Informal, Self-Organized Birds of a Feather Get-Together] (Felicitas, Student Union Building)
Bring your DHSI nametag and enjoy your first tipple on us!

Thursday, 14 June 2018

9:00 to Noon
Classes in Session

12:15 to 1:15
Lunch break / Unconference
"Mystery" Lunches

1:30 to 4:00
Classes in Session

4:15 to 5:15
DHSI Colloquium Lightning Talk Session 6 (MacLaurin A144)
Chair: Lindsey Seatter
- Composition not Inheritance: Imagining a Functional Digital Humanities. Andrew Pilsch (Texas A&M U)
- Plotting Our Trajectories: Navigating, Situating, and Re-Inventing Research Topoi with R. Sean McCullough (Texas Christian University) and Jongkeyong Kim (Texas Christian U)
- Herb Simon and His Books. Avery Wiscomb (Carnegie Mellon U) and Daniel Evans (Carnegie Mellon U)
- (De/Re)Defining "The Digital": A Decolonial Approach to Digital Humanities. Ashley Caranto Morford (U Toronto) and Arun Jacob (McMaster U)

7:30 to 9:30
(Groovier?) Movie(r) Night (MacLaurin A144)

Friday, 15 June 2018

9:00 to Noon
Classes in Session

12:15 to 1:15
Lunch Reception / Course E-Exhibits (MacLaurin A100)
1:30 to 2:30

(MacLaurin A144)

Abstract: Much has changed and continues to change in digital humanities since the formal establishment of Iter in the Fall of 1997. However, the mandate of the not-for-profit partnership to support “the advancement of learning in the study and teaching of Middle Ages and Renaissance (400–1700) through the development and distribution of online resources” continues to have relevance. This presentation explores the striking challenges faced by Iter and presents our current thinking on the realization of this mandate for the future through a platform with a focus on facilitating the discovery of the academic resources necessary to our work; creating an environment for collaboration, sharing and developing projects; and on enabling the distribution and publication of our scholarship.

2:40 to 3:00

Awards and Bursaries Recognition
Closing, DHSI in Review (MacLaurin A144)

Contact info:
institut@uvic.ca P: 250-472-5401 F: 250-472-5681
Instructors:

Garth Evans (garth.evans@ubc.ca)
Josh Romphf (jromphf@library.rochester.edu)

Description:

From YouTube, to image repositories, to podcasts, to scraping media from web services like eBay, Reddit, and 4Chan the wealth of information available to humanities scholars that falls outside the realm of “traditional sources” is staggering and will continue to increase for the foreseeable future. Traditional scholarly approaches will still have their place among these new media objects but will frequently need to be used in conjunction with methods for handling large volumes of new media. But what are these methods and when/how are they used? This course answers these questions by starting from a basic introduction to media types and their potential research value and then leading the hands-on process for building a pipeline for processing each, from collecting the material through to processing it and finally storing it. Exact sources of the media to be used are still being considered but still images, sound files, and video will all feature prominently. No previous experience working with media files of any type is required but would certainly be an asset.

Introduction:

Welcome everyone! Throughout the week we’ll be experimenting with a variety of media processing technologies. The course will primarily be workshop based, with plenty of time for independent projects and troubleshooting. While we’ll be covering a lot of technologies, one thing we won’t be doing is shooting any video or recording any audio (at least not in the traditional “production” sense). Rather, we’ll be manipulating existing media files, so we highly encourage you to bring some of your own to work with, but we’ll also supply several examples. Please refer to the table below for the week’s structure. We’re intentionally keeping this pretty loose; so if there’s some subject matter you’d like to touch on, please let us know. Given the nature of media processing, many of the tools we’ll be using require “low level” access to the underlying hardware of our machines. Similarly, most of these tools need to be compiled for native processor architectures. What this means is that we may be spending a bit of time installing
some of these tools. Since our hope isn’t for us to be in “compiler hell,” we’ll rely on Compute Canada’s vast resources in order to do a lot of this work remotely. For those that would like to work locally, we’ll also supply some build scripts and instructions well in advance of our class. Finally, we think it’s important to stress that no programming or multimedia experience is required for this course. With a bit of curiosity and patience, we can all have a great time.

**Agenda:**

<table>
<thead>
<tr>
<th></th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Afternoon</strong></td>
<td>From Analog to Digital: A Brief History of Recorded Image and Sound</td>
<td>Digging into Multimedia Objects: Command Line Processing</td>
<td>Working with Media Collections: Data Retrieval (Internet Archive Case Study)</td>
<td>Work on Multimedia Projects</td>
<td>DHSI Project Show and Tell</td>
</tr>
</tbody>
</table>
Digitization of cultural heritage over last 20 years has opened up very interesting possibilities for the study of our cultural past using computational “big data” methods. Today, as over two billion people create global “digital culture” by sharing their photos, video, links, writing posts, comments, ratings, etc., we can also use the same methods to study this universe of contemporary digital culture.

In this chapter I will discuss a number of issues regarding the “shape” of the digital visual collections we have, from the point of view of researchers who use computational methods. They are working today in many fields including computer science, computational sociology, digital art history, digital humanities, digital heritage and Cultural Analytics – which is the term I introduced in 2007 to refer to all of this research, and also to a particular research program of our own lab that has focused on exploring large visual collections.

Regardless of what analytical methods are used in this research, the analysis has to start with some concrete existing data. The “shapes” of existing digital collections may enable some research directions and make others more difficult. So what is the data universe created by digitization, what does it make possible, and also impossible?

The Islands and The Ocean

Before born-digital content, media creators first used physical and later electronic media (video and audio). Starting in the middle 1990s, gradually more and more of this content has being digitized. We can call such content born-analog.

The very first project to digitize cultural texts and make them freely available was Project Gutenberg that started in 1970. Today the largest sites for digitized content include Europeana (over 53 milion “artworks, artefacts, books, videos, and sounds from across
Europe” as of 2016), Digital Public Library of America (over 13 million items as of 2016), HathiTrust (13 million volumes as of 2015), Digital Collections at the Library of Congress and Internet Archive. The latter contains digital collections of various types of media ranging from largest collection of historical software to 10.7 billion historical texts (as of 12/2016).

The sites typically offer a number of useful ways to navigate these massive collections. For example, the Digital Public Library of America (DPLA) supports direct search, view by Timeline, Map view, and Thematic Exhibitions. Both DPLA and Europeana also encourage and help developers create experimental interfaces and apps that expand how their artifacts can be viewed and used. But in terms of using them for Cultural Analytics research, they do have one limitation. While the works in these and other collections can always be viewed online, not all works can be downloaded (or downloaded in mass using an API), because of the restrictions imposed by owners of the works.

The site which in my view is most interesting in this genre is Google Arts & Culture. It has fewer works but the most fluid interface. This site grew from the earlier Google Art Project that worked with many museums to scan artworks and then presented them online in a “virtual museum” interface. Today it offers virtual tours of many museums, millions of digitized artworks and photographs from the past, contemporary art. Media projects and photo stories are also created. The interfaces include zoom, timeline, search by color, thematic exhibitions, and also categories (artists, mediums, art movements, partners, names of objects, and places). When I was exploring the website (July 2016), it was offering 3,000 thematic exhibitions on all kinds of cultural topics. When we started our own Cultural Analytics Lab (culturalanalytics.info) in 2007, it was a bet. While contemporary culture was already well represented on the web, the kinds of large-scale online digital collections with multiple navigation functions and API like Europeana or DPLA did not yet exist. But I assumed that within the next few years, millions of digital images of historical art, photography and other media would become available. However, it was not clear at that time how inclusive they would become.

In the article I wrote about Cultural Analytics in March 2009, I described my experience of trying to use the existing digital image collections available at that time. I was interested in the following question: What did people paint around the world in 1930 – aside from a number of modernist “isms” that encompassed at best 150 artists (working in Paris, Amsterdam, Berlin and a few other cities) who are now included in the Western art historical canon? I was not thinking of “paintings in tens of thousands of small museums in small

---

5 Manovich 2009.
cities,” rather of paintings of nationally “important” artists that have entered in art history canons in their countries.

I did a search on artstor.org – a leading commercial service for digital images of art used in most art history classes in U.S. and also in other countries. In 2009 it already contained close to one million digital images of art, architecture and design. These images came from many important USA museums, art collections, and university libraries. To collect the images of artworks that are outside of the usual Western art historical canon on Artstor, we excluded Western Europe and North America from the search. This left the rest of the world: Eastern Europe, South-East Asia, East Asia, West Asia, Oceania, Central America, South America, and Africa. Not a small area! But when we searched Artstor for paintings done in these parts of the world in 1930, we only found a few dozen images. So, while there were very large numbers of images of paintings of canonical artists from Europe and USA painted in the same year, there were only a few images for a whole continent like East Asia.

This highly uneven distribution of digitized cultural artifacts is not due to Artstor’s choices. Artstor does not digitize images itself. Instead, it makes images available that have been submitted by museums and other cultural institutions. The results of our search reflects what participating museums collect and what they think should be digitized first. In other words, a number of major US collections and a slide library of a major research university (where by 2007 the proportion of Asian students was 45%) together contained only a few dozen paintings created outside of the West in 1930 which were digitized. In contrast, searching for Picasso returned around 700 images. Describing this example, I wrote in this 2009 article:

If this example is any indication, digital art repositories may be amplifying the already existed biases and filters of modern cultural canons. Instead of transforming the “top forty” into “the long tail,” digitization can be producing the opposite effect.

What remains outside of the digitized collections is all the rest: provincial nineteenth century newspapers sitting in some library somewhere; millions of paintings in tens of thousands of small museums in small cities around the world; millions of thousands of specialized magazines in all kinds of fields and areas which no longer even exist; millions of home movies and photographs… This creates a problem for Cultural Analytics, which has a potential to map everything that remains outside the canon – and to begin writing a more inclusive cultural history without “great names.” We want to understand not only the exceptional but also the typical; not only the few “cultural sentences spoken by a few “great men” but the patterns in all cultural sentences spoken by everybody else; what is outside a few great museums rather than what is inside and what has already been discussed extensively and too many times.

---

6 The very first large institutional collection that formed the core of Artstor was the slide library of the University of California, San Diego (UCSD) – the same university where I had been teaching since 1996. The library had over 200,000 slides, and they were all digitized and included in Artstor. In 2009, this was the largest single collection in Artstor. The slides were either directly created by art history faculty teaching in Visual Art Department, or by art library staff following lists of images faculty provided. This collection is very interesting because it reflects the biases of art history as it was taught over a few decades when color slides were the main media for teaching and studying art.
I worried that what has been digitized, is only an “island,” and that a massive cultural “ocean” remains inaccessible for quantitative analysis. Luckily, such amplification of biases and focus on only “what is important” did not happen. Exploring the online libraries of digitized cultural artifacts seven years later, I am amazed by their richness and variety. The reason is that Europeana, DPLA, Library of Congress, NYPL, Internet Archive or Google Arts & Cultures do not just offer us images of high art like art museums. Instead, they are extensions of traditional libraries. And the libraries in modern times have an important function besides offering readers books and periodicals – they are places to which numerous people and organizations donate their archives. As these archives started to be digitized, an amazingly rich and varied historical cultural landscape started to emerge online.

For example, here are three examples among hundreds of digital image collections from the New York Public Library (NYPL):

“Photographs of The Catskill Water Supply System in Process of Construction.” 55 albumen print photographs created between 1906 and 1915.7
“Buttolph Collection of Menus” – A collection of Miss Frank E. Buttolph (1850–1924), a somewhat mysterious and passionate figure, whose mission in life was to collect menus donated to NYPL in 1899, 18,964 digitized items.8
“Catalog of the Chiroptera, by G. E. Dobson” – 31 digitized prints from a 1878 book.9

And here are examples listed in the blog post from europeana.eu referred to as “highlights of the new datasets ingested in the last months”:

Almost 100 objects (drawings, paintings, photographs) from Telegraph Museum in UK.
Over 3,000 photographs, XIX and XX century, mostly buildings from Culture Centre in Helsingborg.
Collection of 620 botanical drawings by Georg Schweinfurth from Botanic Garden and Botanical Museum Berlin-Dahlem.10

Comparing these collections with those of the digital image offerings of the largest art museums, we find that they are complete opposites of each other. Although modern art museums’ collections like that of libraries also developed through both their purchasing programs and donations, what was donated to them – or what museums chose to accept – was quite different. Libraries ended up housing millions of all kinds of heterogeneous items, few of them financially valuable. In contrast, modern art museums have traditionally focused on what has been recognized as very valuable. Indeed, original European “museums” included estates of very rich people, parts of royal palaces, or treasures of cathedrals and churches. For example, Vatican Museums originated in 1506 when Pope Julius II purchased

the ancient sculpture of Laocoön and his Sons and placed it on public display. (I should note that digitized collections of design and crafts museums such as Victoria and Albert in London or Cooper-Hewitt in New York are closer to that of libraries – their holdings are more varied and also organized in more categories than those of art museums.)

Libraries vs. Museums

However, there is also another aspect in museum’s history. Some of the original European museums contained not art but “curiosities.” One such famous museum is The Kunstkamera that was established in St. Petersburg in 1716 by Peter the Great to present “natural and human curiosities and rarities.” Another is the British Museum that opened in London in 1759, that initially showed a private collection of the physician and scientist Sir Hans Sloane.

Art history since the 20th century has created a highly controlled system that divides our visual heritage into “art” and everything else, and organizes the former by artists (their national origin, time period, and medium and style). The digital collections of art museums today also look ordered and systematic.

We are used to their ordered classifications. In comparison, the meta-collections of digitized visual artifacts by Europeana, DPLA and others may remind us of the cabinets of curiosities. Instead of a military-like “parades” of art history played in physical museums or on their sites, we find “trivia” and “ephemera.” (The latter word comes from Greek and New Latin where it referred to insects or flowers that were alive sometimes for less than a day.)

Browsing through page after page describing endless collections that often contain a few dozen or even only a few items – like the ones in the examples above – I often feel uncanny. In this view, the past looks un-periodic and un-systematized. Endless “deposits” of human material cultures have remained inside libraries, have then been digitized and are now connected by common metadata standards, web protocols, Javascript code, APIs and other computer machinery.

Labyrinth, kaleidoscope, Kunstkamera, Memex’ hypertext, random access memory, relational database – none of these models describe my experience of navigating digital cultural collections. For instance, consider Europeana with its 53 million items. The idea behind this massive multi-year project was to connect digitized artifacts from thousands of European museums and regional archives. So, rather than having to search all their individual sites, you can use the Europeana platform as a single point of access. The platform provides a common interface to all of the objects but it does not store them. They are stored at individual museums and archives. European Film Gateway, one of Europeana’s projects, does the same for dozens of European film archives.

Technically and conceptually, this works brilliantly. But experientially, the result has some unintended consequences. Instead of creating a kind of “united Europe” – a single pan-European space for cultural heritage – Europeana may be fragmenting it. As I browse through endless separate collections or individual items from these collections that fit my search terms, countries, geographic relations, and time periods are dissolved. Instead of a
“European” continent, it feels that I am looking at random survived files of many alien civilizations that got all mixed together.

This feeling is created by both very heterogeneous topics, and by equally heterogeneous styles. Photographs created in all kinds of techniques, engravings, etching, newspaper illustrations, covers of cigarette cases, early hand-colored photos, paintings … images are in rectangular formats, round frames, part of a text page, drawn in a corner of a hand written letter… texts typed, types set, hand written, printed on early dot matrix printers, carefully drawn with a brush … every possible subject and form of visual communication is here. (If Instagram platform during 2010–2015 can be thought as the extreme example of visual constraints, with all image being the same size and format and belonging to one medium, a digital historical collection is the other extreme).

But this heterogeneity, richness and variety is actually a good thing. It makes us aware of how rigid and limited our concepts of an “image” are today – a few clearly separated mediums, rectangular formats, and also separation between images and texts. So, while the abundance of communication “species” in digital libraries is on first sight disorientating – and it is certainly a challenge for large scale analysis using Computer Vision systems initially developed for contemporary photos – in the long run it is best for us. It forces us to face the human visual culture as it really exists historically – thousands of variations and their combinations, rather a net set of a small number of categories.

Cultural Sampling

The “islands” of digitized historical contents are constantly growing. But will they ever be big enough to let us understand the “ocean” – i.e., construct a sufficiently detailed map of the human visual history of the last few centuries? Richness and variety do not mean comprehensiveness. In other words: while digitization and organization of digitized items by Europeana, DPLA, and other projects continues, the most basic question for any quantitative study of cultural history remains unaddressed. This question is, how can we compile representative samples that systematically cover everything created in a particular period, geographic area and media – or in many such periods and areas together?\footnote{For an overview of different sampling methods, see Cook 2011 and Chambers / Skinner 2003.}

Anthropologists do use sampling methods in their research when they excavate sites or study groups of people (such as in urban anthropology that looks at contemporary cities). But there is a basic large question which is more difficult to address: Since the kinds and quantities of artifacts that remained from various ancient civilizations vary significantly, do they together add to a representative sample? (Of course, as excavations of sites and analysis of new artifacts continue, this sample is being continuously refined.)

Since I am a historian of modern visual culture and media of the last 200 years, I am confident that for this period we do not have any comprehensive sample of visual culture in this period before the arrival of social media. So, while the “islands” are increasing
in size and number, reconstructing the whole ocean maybe may become very difficult. I am using the term “sample” in the sense it is used in statistics: a smaller subset of the larger data. Constructing proper samples and determining the validity of predictions based on these samples is one of the main areas of statistics. In all social sciences including sociology, demographics, psychology, and political science these questions are particularly crucial, since these disciplines often use small human groups for surveys or observation. Construction of proper samples is also crucial for marketing research, human-computer interaction research and all other applied fields where researchers want to find people’s attitude about existing products, interest in new products and new features, their lifestyle aspirations, etc. And while the arrival of big social media data in the second part of the 2000s has changed the situation significantly, because now businesses can follow online millions of individuals tracking what pages they visit, what they click on, which ads they look at, and what they purchase, small groups continued to be widely used. (You can ask people who agreed to participate all kinds questions, or place them in situations and see what they chose – something which is not always possible online.)

We do not have systematic samples of modern visual and media culture. Instead we have numerous separate collections and archives that are being digitized. Therefore, the kind of question I asked in 2009 –What did people painted around the world in 1930? – is still unanswerable. And for many other questions, the situation is even worse. Consider for example the history of photography. While working on a book about Instagram aesthetics in the context of modern design, art and photography, I had a pretty big sample of Instagram: 16 million photos shared in 17 global cities between 2012 and 2016. It is important to note that these are not photos with particular tags. Instead, they are all geo-coded photos shared in larger city areas during a particular period. According to a few of computer science publications that analyzed large samples of Instagram posts in 2014, during that time Instagram users shared locations for 20% of their photos. This means that our datasets also represent approximately 20% of all Instagram photos shared in a given area and period. From a sampling point of view, these are very good samples. Not only are they quite substantial but we also know what part of a “population” is represented. (“Population” in statistics is a technical term that refers to the whole data that for practical reasons is not accessible to us. Instead, we can use small samples from which we can probabilistically infer characteristics of the whole data.)

I certainly did not expect to find anything like these samples for vernacular photography in the 20th century. But I assumed that after all digitization work of the last twenty years, I can easily find samples of at least few thousand digitized photographs for particular decades, and maybe even for particular countries. It turns out that nothing like this existed.

What has been digitized and made available online are various collections of vernacular photography from particular private collections. They added certain photos to their collec-

---

12 Manovich 2016.
13 Manikonda et al. 2014.
tions because each photo was interesting to them for some reason. Museum exhibitions of vernacular photography that I consulted were similarly “non-objective” – they were assembled by curators who had particular curatorial ideas. I also found some user groups on Flickr with “found photographs” contributed by group members. Every collection I consulted was the result of individual or groups’ taste and ideas of what should be included. Often people were only interested in more “artistic” and “avant-garde” examples of vernacular photography, rather than the typical.

To my knowledge, nobody has ever thought to create a representative sample that would contain characteristics of the field of vernacular photography as a whole in particular historical periods, types of cameras and printing, and so on (for example, photos made with Kodak Brownie cameras of 1900, or first portable 35-mm Leicas in 1925, or prints using Kodacolor after 1942, or Polaroid prints after 1972.) So now that we have learned from computer science studies of massive social media samples that we can look at any culture as a statistical population asking about distributions, averages, variance, clusters, and so on, we want similar historical samples. But they do not exist.

For example, the National Gallery of Art in Washington presented an exhibition in 2010 called The Art of the American Snapshot, 1888–1978: From the collection of Robert E. Jackson. According to the curators, “Organized chronologically, the exhibition focuses on the changes in culture and technology that enabled and determined the look of snapshots. It examines the influence of popular imagery, as well as the use of recurring poses, viewpoints, framing, camera tricks, and subject matter, noting how they shift over time.”

The online exhibition catalog shows that curators did an excellent job of capturing a number of aspects of vernacular photography and its evolution. However, since the exhibition only had 200 photographs for a 90-year period, that meant that the historical map exhibition constructs was very “low resolution” (to use the spatial metaphor) and also not complete. If we want to understand differences in snapshot photography between different countries, or find gradual changes in style or subjects that are not related only to the introduction of new photography technologies, or see if there may be some regional or demographic differences, we cannot accomplish this with 200 photos.

For a comparison, consider the Gallup U.S. Daily poll. For this poll, Gallop interviews (over the phone) 500 people across U.S. every day. For a country of 300 million people, this looks like a tiny sample. But because Gallup selects people at random and conducts these interviews every day, it accumulates 15,000 responses per month, and 175,000 per year. We also learn that “Gallup also weights its final samples to match the U.S. population

according to gender, age, race, Hispanic ethnicity, education, region, population density, and phone status. This weighting is done using data from a number of other surveys. For example, to weight by population density, Gallup uses U.S. Census reports. This systematic approach to sampling and analysis of the results is typical of all natural and social sciences, public administration, demographics, public polls, marketing research, and countless other areas. In fact, the only area where it is absent is humanities.

The question humanists have been asking is about canon, and how to make canons in their field more representative. There is a parallel here with the kind of weighting Gallup and other organizations that collect demographic data do. However, sometimes in the attempts to compensate for a lack of representation of older canons, the new canons are “weighted” more towards groups that were previously not represented. So as a result, we once again get something completely driven by ideologies, rather than a balanced sample.

A “balanced cultural sample” can be defined in multiple ways, all equally informative and complementary to each other. For example, we can include a proportion of all works produced in particular media, period, and place. Or we can focus instead not on what has been produced, but what audiences actually read, watched, or listened to. We may decide to select only works that achieved certain recognition (which would be equivalent of likes and favorites in contemporary social media), or disregard this information. But whatever we do, we need a systematic procedure, not simply a taste judgment. Statistics has developed a sophisticated theory of sampling which includes many methods, and since these methods are used today in all sciences, they should be adopted for analysis of historical cultural artifacts as well – if we are interested in understanding them as a kind of ecological or geological system, where all participants and artifacts are important – as opposed to only a set of “masterpieces.”

The idea of creating systematic and representative samples of culture is interesting by itself, because it leads to all kinds of follow up questions. And since our textbooks, museums, cultural portals, classes, and documentaries always represent human arts and cultures using only selected examples, the questions about cultural sampling are important in general, even if we are not conducting quantitative analysis. They relate to how we understand, represent and teach human cultural history – and also how we think about our cultural present, with its new scale of numbers of participants, their cultural interactions and experiences.

For example, imagine a hypothetical scenario where we can include any painting created in France in the 19th century in our sample. Now imagine that we want to create a representative sample, so we randomly select X number of paintings. Such a sample will include several academic salon paintings, realistic paintings, portraits and so on. And it would miss the 19th century art which we now recognize as most important – works by Impressionists and Post-impressionists. Why? It has been estimated that 13 key French Impressionists artists together created 13,000 paintings and pastels during their lifetimes. But this is a
very small number in comparison to all paintings created by artists living in France during the whole 19th century. So, a random sample would likely miss them all.

This is exactly the same problem, which accompanies a great deal of quantitative social media research in Computer Science. In many articles, authors explain how they carefully construct a random sample drawn from all users of Pinterest, Instagram or Twitter. Using such samples, they then develop statistical models that account for some characteristics of the behavior and posts of these users. This research is very interesting and important. But using a single global sample of a network with hundreds of millions of people from most countries in the world sharing billions of daily text posts, images and video has serious limitations. We can only see the “typical.” So we miss all kinds of regional variations, and presence and activity of endless users who don’t have the typical behaviors and posts. In other words, if any of these networks have their own “Impressionists,” they are not visible in the analysis that uses single random samples.

Sometimes, the sampling procedures used end up only including particular types of users. For example, in the paper “Analyzing User Activities, Demographics, Social Network Structure and User-Generated Content on Instagram” (2014), the researchers state: “To the best of our knowledge, we believe this is the first paper to conduct an extensive and deep analysis of Instagram’s social network, user activities, demographics, and the content posted by users on Instagram.”

This is how they describe the method they used to create a user sample for their study:

First, we retrieved the unique IDs of users who had pictures that appeared on Instagram’s public timeline by using Instagram API, which displays a subset of Instagram media that was most popular at the moment. This process resulted in a sample of unique users. However, after careful examination of each user in this sample, we found that these users were mostly celebrities (which explains why their posts were so popular). To avoid the sampling bias, for each user in this sample, we crawled the IDs of both their followers and friends, and later merged two lists to form one unified seed user list which contained 1 million unique users.

The final dataset has 5,659,795 images for 369,828 users (the rest had private accounts). Out of these images, 1,064,041 have geo-locations. But how well are these users representing the Instagram universe? Most people follow other people as opposed to celebrities. People who do follow celebrities and their friends are likely only one type of Instagram user. Additionally, given that the number of Instagram users in every country differs, with the biggest countries also often having larger number of users, such a “random” sample likely better represents some countries than others.

These considerations do not invalidate the results in this and all other papers that use a single large sample from massive global social networks. Their findings are valid. They just may not apply to every type of user or type of post on such networks. (Note that we are not talking about individual users but groupings, each with their own characteristics. In other words, these are like 19th century Impressionists who had common characteristics.)

18 Manikonda et al. 2014.
We also need to recall here perhaps the most fundamental “Achilles’ heel” of statistics. “The goal of statistics is to represent the facts in the most condensed way” (1833). But we pay a big price for such compression. The measures used in descriptive statistics summarize some population (i.e., a set of items) but they may not correspond to any concrete members of this population. For an example, let’s take a series of numbers: 1,1,2,3,2,9,9,10,11,11. The average (called “mean” in statistics) of this series is 6.36. But we don’t have any actual numbers close to this mean! No. 4, 5, or 6. Instead, we have two “clusters”: 1 to 3, and another one from 9 to 11. (This is called a bimodal distribution.)

In other words, the standard statistical measures of a large population can easily miss the presence of various groupings in this population. So, if we represent some “cultural population” – be it 19th century paintings or 20th century cinema, Instagram today, or global music videos – with a single random sample, we can miss all kinds of groupings (1960s New Wave or 1920s Soviet Montage school in cinema history; contemporary music videos from India, Korea, Vietnam, Thailand or Kazakhstan which have their own differences despite overall similarity; and so on.) And the characteristics which we will find may describe the “average” which never existed in reality. That is, it may not correspond to any actual group. And rather than capturing the presence of multiple distinct groups, it can hide them from view.

In fact, I would like to claim that in human societies and cultures there are no “averages.” Certainly, we can follow Adolphe Quetelet who in the early 1830s was the first to start to measure the physical characteristics of humans such as height and weight and found that their distributions followed “normal” curves. If we perform such measurements today, we will find similar distributions. And, in a sample of a million people, certainly many would have the exact height specified by the mean. In the same way, if we for example measure the length of tens of thousands of modern novels, we will find that some do have exactly the same length as the average novel.

But such results only hold if we limit the study of cultural artifacts, interactions, and experiences to one characteristic at a time. If we look at several selfies sampled from Instagram, we can calculate the average degree of smile, size of a face in a photo, and its position. And if the sample size is big enough, some actual selfies will have exactly the same numbers as the averages. But just as a face of every person is unique, like their fingerprints, their photos are also unique. So if we multiply the number of characteristics, eventually we will not find any real selfie that matches the sample averages on all of them. The same applies to any other type of cultural expression, past or present.

There is one field that does think about cultural sampling and it is using statistical methods to create and analyze these samples. This field is the sociology of culture. The most well-known book in this field remains famous Distinction: A Social Critique of the Judgement of Taste by French sociologist Pierre Bourdieu. Published in 1979, it has been recognized as one of the ten most important books of sociology in the 20th century. Bourdieu offered

19 Tyler 1872.
powerful intellectual ideas and theories that connected people’s cultural tastes and their socio-economic statuses. These theories were grounded in the statistical analysis of two large surveys of tastes of the French public conducted in the 1960s. Bourdieu collaborated with French “data scientists” (to use contemporary term) who developed a new analytical and visualization methods to represent relations between many elements, and he used this method in all of his later studies including Distinction.

Today sociologists of culture continue to use surveys of groups of people, but they also use samples from cultural publications. One example of the former is a study where the researchers “asked 1544 German-speaking research participants to list adjectives that they use to label aesthetic dimensions of literature in general and of individual literary forms and genres in particular (novels, short stories, poems, plays, comedies).” The example of the later is a study called “Institutional Recognition in the Transnational Literary Field, 1955–2005.” It uses “a sample of articles from 1955, 1975, 1995 and 2005 in French, German, Dutch and US elite papers (N=2,419).” Here is another example: an analysis of fashion discourse during 1949–2010 that uses 1301 fashion reviews from The New York Times and The International Herald Tribune. Although such samples are rather small in comparisons to social media scale, they are sufficient to answer particular questions the researchers asked in these studies.

When I first thought of cultural analytics in 2005, I imagined being able to construct detailed world-wide maps of particular fields – such as painting, cinema, graphic design or music video – for long historical periods. But as I realized that digitization efforts are not creating systematic samples such maps would require, I had to abandon these ideas for the time being. So instead, I focused on a different type of sampling that I could do given what has been digitized – by type of media. Starting in 2008, in our lab, we have worked on over 40 datasets that cover almost every major type of visual media today. We analyzed comics and Manga series, video games, feature films, documentaries, motion graphics, music video, political video ads, print magazines, historical photographs, born-digital photographs and other images, and interactive virtual worlds. We also deliberately included dataset that lie at the extremes of a high – low and professional – non-professional dimensions: from paintings of van Gogh, Mondrian and Rothko to 10 million Instagram photos shared in New York City by 5 million people. And we have also deliberately balanced Western and non-Western cultural sources. The latter include Japanese video games, music videos from across Korea, Instagram photos shared in seventeen global cities that cover four continents. We published analysis using Instagram photos shared in Tel Aviv, Israel during Fallen Soldiers and Victims of Terrorism Remembrance Day, and another analysis of Instagram photos shared during February 2014 Maidan revolution in Kiev, Ukraine.

20 Knoopa et al. 2016.
21 Verboord et al. 2015.
22 Van de Peer 2014.
In fact, the advantage of using social media data is that it is not “canonical” or “national.” Popular networks such as Facebook, Instagram, and others are used in every country except the few where they are/were blocked for periods of time (in the case of Facebook, Bangladesh, China, Iran, North Korea, Syria\(^\text{23}\)). As of May 2016, the messaging app WhatsApp that started in China was used in 109 countries, with one billion users sending 42 billion messages daily.\(^\text{24}\) And by the same time, 80% of Instagram 500M active users were outside U.S.\(^\text{25}\)

For example, when we were creating our Instagram samples datasets between 2012 and 2016, Instagram API allowed anyone to download all geo-tagged photos shared within a particular rectangular area defined by its latitude and longitude. Each area could be 5km × 5km in size, and collecting from a number of areas was not more complicated. So it was equally easy to download images from parts of Manhattan, or Moscow, or Bangkok, or Kiev, and so on. (To download all geotagged images shared during five months in Manhattan, we combined a number of areas to enclose the island in a large rectangle, and then filtered out the data outside of Manhattan boundaries).

This means that in practice, comparing many areas from around the world is as easy as comparing nearby areas from the same city – as long people share sufficient amounts of social media in these global areas. The global perspective is “built in” in social media. This of course also applies to the standard formats, constraints and affordances particular networks and apps provide for their users. Everyone who used Twitter between 2007 and 2017 had to fit their messages into the same 140 characters. Everyone who was using Instagram between 2010 and 2015 had to submit to its square image format and the same size: 640 × 640 (or 612 × 612). Everyone has access to exactly the same functions (adding hashtags, optional geo-tagging, etc.) and the same UI. This by itself raises an important question: does social media software lead to less diversity in user-generated content? This was one of the key questions for me during my eight years of research.

**Data Representation**

However, like every other type of data about society, social media data has its own limitations, and they are not insignificant. I will briefly discuss five issues which are all about representation – what gets represented (and available for research) and what is absent. While the use of social networks and the web continues to grow around the world, billions of people do not use them. Here is a concrete example from our own research of how this situation limits what we can “see” using their data. In 2014, Twitter agreed to provide selected researchers with access to any part of their data if they used it in new interesting ways. Thirteen hundred labs from around the world applied, and we were one of six labs

\(^\text{23}\) Kirkland 2014.
\(^\text{24}\) Smith 2016.
\(^\text{25}\) Facebook 2016.
that won. I asked Twitter to give us all tweets with geo-located images shared with them. Twitter added images functionality in 2011, and we were given access to all tweets with geo-located images shared worldwide between 2011 and 2014. When we plotted locations of a random sample of 100 million tweets from this data approximately half of the populated Earth surface had no coverage.

The second issue has to do with demographics of users who do use social networks. In “developed” countries and global megacities, people from all demographic groups use the networks. In a country like the USA, there is no significant differences in social network use between women and men, or different races, or people with different level of education – but there are still big differences between age groups. This is also true globally – although the differences are getting smaller with time. A report on social media use among people who were online in 34 countries in first quarter in 2016 found that 92% of those who are in 45–54 age group have social media accounts; for people in 55–65 age group the figure is 82%.²⁶

In many developing countries, the proportions of people using social networks among those using the web are higher than in developed countries. At first, this looks like good news because it could mean that we get data on cultural activities of larger proportion of populations in these countries. However, the reality is different. As the report explains, “As many as 98% of Internet users in countries like Malaysia, Brazil, Indonesia and Vietnam are on at least one network. In part, that’s a result of their lower Internet penetration levels, which means online adults in these regions are more likely than their counterparts in Europe or North America to come from young, urban and relatively affluent segments.”²⁷

The third issue is uneven spatial distribution of social networks activity and content even in big urban areas where we see very high use – until we zoom in. The amount of sharing and participation can vary dramatically between city areas, as we show in the Inequaligram project. We collected and analyzed 7,442,454 public geo-tagged Instagram images shared in Manhattan over five months. The inequality we found between the more populated and less populated parts of Manhattan was staggering. We found that the ratio between a square km area with most images and the area with least images was 250,000:1. According to our analysis, 50% of all images shared by local residents are within only 21% of Manhattan area. For visitors, this difference is almost twice as big: 50% of their images were shared in only 12% of the Manhattan area. In summary, even for such a densely populated urban area as Manhattan, its Instagram collective image only reflects part of it and not all.

The forth issue is what content people share, what comments they make, and what they are willing to say online. Social networks are not a mirror of society. Just as people in other areas of their lives play roles, follow norms, present particular identities and behave in ways expected from them (by “mainstream,” or their particular “subcultures,” or “tribes), they do this online. And because their posts and comments can be seen by all other network

²⁶ GWI 2016.
²⁷ Ibid.
users (unless they make posts or the whole account private), appear in Google search, and are saved by the networks, shared with marketers, etc., they are likely to be extra-careful. And, just as with professional cultural products, some of user-generated content is driven by conventions, stereotypes and models people see around them. For example, we find endless photos in “table top” genre on Instagram created by regular users, overwhelming proportions of selfies smile (see our selfiecity.net and selfiecity.net/London for more details), and travel photos follow their own conventions. All this means that the “culture” we can analyze using social media is its own universe, and not a simple sample of people’s cultural activities, taste and opinions outside the networks.

Finally, the fifth issue is access to social media data. In the middle part of 2000, all large social networks created APIs that allow people to freely download large data samples containing user posts and all public information about them visible online – date and time a post was shared, location (if user shared this information), username, tags, comments, and numbers of likes and re-shares. In the case of visual networks such as Instagram and Flickr, image and video along with their user descriptions and all other information was also available for downloads. Flickr launched its API in 2004, and Facebook and Twitter in 2006.28

While these APIs were intended for developers building apps that use data from the platforms, and for users to share contents between networks and also their blogs, computer science researchers, data visualization artists, and other creative technologists realized that they can also freely access this data, and numerous studies and projects were created. Hundreds of thousands of computer and social scientists and students used these APIs to download data, analyze it and publish papers.

However, there have always been limits on how much data can be downloaded. For example, during the period we were actively downloading Instagram data (2012–2016), it had a limit of 3000 images per hour, and only images from the last few days were available. Nevertheless, we were able to assemble 16 million Instagram photos shared in 17 global cities in different periods between 2012 and 2016. But given that in 2016 people are sharing 80 millions of images on Instagram per day, what we were able to assemble was a tiny portion.

However, because of the concerns with privacy and unauthorized use of posts, some of the biggest networks gradually limited or closed API access to bulk user data. Facebook limited the use of its API on April 30, 2015, and Instagram stopped allowing bulk downloads on June 1, 2016. At this moment (end of 2016), Twitter is still accessible, along with some networks popular in particular geographic areas such as Russian VK.

In summary, we know that social media and the web are not used by everyone; the proportions and demographics of those who use social media varies from place to place; and what people publish and share constitutes its own cultural reality as opposed to being a transparent window into the realities outside. We should always keep these limitations in mind. At the same time, using the web and social media data and contemporary

28 Lane 2012.
technologies for tracking and analyzing it questions the very idea of representation. This concerns the very foundation of modern research methods based on sampling.

These methods assume that for practical reasons we cannot have access to the complete “population” (i.e., full data). We can only access and analyze one or more samples of the population. Accordingly, modern statistics is divided into two areas. Inferential statistics is a set of methods for estimating characteristics of the population based on its sample(s). Descriptive statistics only describes the properties of whatever data we have, and it does not assume that this data came from a larger population.

However, when we analyze web and social media content and interactions, we often can have full data. Certainly, the companies that run social networks, media sharing sites or publishing platforms can record all interactions happening on their platforms. This is true for Facebook, YouTube, Twitter, Pinterest, Spotify, Amazon, Scribd, Shutterstock, Behance, academia.edu, and other social media and publication services. This does not mean that a company will be analyzing all their data, or keeping it forever, or even have its own researchers work on it – because companies don’t want to sued, have bad publicity or get in trouble with governments. So the data is anonymized, sampled when needed, and only particular parts of the data are made available to internal researchers depending on what lab they work for. However, the largest companies certainly take advantage of having massive data about user interactions on their platforms, using it to train systems that recommend other users to follow or other videos to watch and decide which posts from friends to show, select trending topics etc. Big data is also driving the main source of income for big social media companies – i.e. automatic advertising systems such as Google AdWords and Facebook Ads.

Although academic researchers do not have direct access to complete data from these companies, it is possible to use their APIs to download complete data that satisfies particular criteria, such as all activity on a particular platform within a particular time period. Many papers use such datasets. In our own work, we also followed this approach. We were using Instagram API to download all publically shared geo-coded images shared in a particular geographic area over a period of time. In fact, every Instagram dataset we used was generated in this way. For example, to create a dataset of 7,442,454 public Instagram images shared in Manhattan over five months, we used a single Mac to run our custom download program 24/7 during this whole period. As far as we know, the images we downloaded are all images people shared within this area and time with geo-location (which constitutes approximately 20% of everything shared).

Why may we want to use complete cultural data? If we are only interested in extracting general patterns, characteristics, and types – for example, the 10 most common types of images on Instagram – we certainly do not need all of the data. But such summarization and aggregation common to the use of statistical methods in 19th and 20th century is only one way to use cultural data. As I explained above, using small samples from diverse cultural “population” (such as trillions of Instagram images) may only reveal the “typical” and “most popular” and miss “regional variations” and “presence and activity of endless users who do not have the typical behaviors and posts.” Therefore, ideally Cultural Analytics should try
Cultural Data
to obtain and analyze complete data generated by some cultural process (be it career of a single photographer or all photos shared on Instagram).

Rather than only treating culture as “data points” that together create patterns that we want to discover, disregarding the individual points afterwards, Cultural Analytics should pay equal attention to both patterns and individual artifacts, experiences and interactions. As creators and audience members, we engage and enjoy concrete artifacts and experiences, and not “patterns.” A particularly successful artifact is often described as “unique” – i.e. it cannot be reduced to already existing patterns. As aesthetic subjects, we search and enjoy such uniqueness. One of the goals of Cultural Analytics is to help us find truly unique artifacts in the infinite universes of media now being created. And even if other artifacts are not unique in most ways, they may still have something unique in other ways, which can get lost if we reduce them to patterns. For instance, every human face is unique, and therefore even the most conventionally-driven photo of this face will be special for us. (In this aspect, Cultural Analytics should combine special perspective of sciences and of humanities – the former’s concern with general laws and regularities, and the latter’s concern with unique cultural objects.)

To conclude, I would like to note one techno-cultural development of the last 20 years that connects many issues I have discussed – the rise of search as a new dominant mode for interacting with information. This development is just one of many consequences of the dramatic and rapid expansion of information and content being produced which we have experienced since the middle of the 1990s. To serve the search results, Google, Bing, Baidu, Yandex, and other search engines analyze many different types of data – including both metadata of particular web pages (so-called “meta elements”) and their content. For example, according to Google, its search engine algorithm uses more than 200 input types.29

However, Google, Yandex or Bing do not reveal the measurements of web pages they analyze – they only serve their conclusions, i.e. which sites best fits the search string user entered determined by their propriety algorithms that combine these measures. In contrast, the goal of Cultural Analytics is to enable what we may call “deep cultural search” – give users the open-source tools so they themselves can analyze any type of cultural content in detail and use the results of this analysis in new ways.

References


Knoopa, Christine A., Valentin Wagnera, Thomas Jacobsen, Winfried Menninghaus: “Mapping the aesthetic space of literature ‘from below’.” In: Poetics 56 (June), pp. 35–49.


Abstract  Contemporary and future historians need to grapple with and confront the challenges posed by web archives. These large collections of material, accessed either through the Internet Archive’s Wayback Machine or through other computational methods, represent both a challenge and an opportunity to historians. Through these collections, we have the potential to access the voices of millions of non-elite individuals (recognizing of course the cleavages in both Web access as well as method of access). To put this in perspective, the Old Bailey Online currently describes its monumental holdings of 197,745 trials between 1674 and 1913 as the “largest body of texts detailing the lives of non-elite people ever published.” GeoCities.com, a platform for everyday web publishing in the mid-to-late 1990s and early 2000s, amounted to over thirty-eight million individual webpages. Historians will have access, in some form, to millions of pages: written by everyday people of various classes, genders, ethnicities, and ages. While the Web was not a perfect democracy by any means – it was and is unevenly accessed across each of those categories – this still represents a massive collection of non-elite speech.

Yet a figure like thirty-eight million webpages is both a blessing and a curse. We cannot read every website, and must instead rely upon discovery tools to find the information that we need. Yet these tools largely do not exist for web archives, or are in a very early state of development: what will they look like? What information do historians want to access? We cannot simply map over web tools optimized for discovering current information through online searches or metadata analysis. We need to find information that mattered at the time, to diverse and very large communities. Furthermore, web pages cannot be viewed in isolation, outside of the networks that they inhabited. In theory, amongst corpuses of millions of pages, researchers can find whatever they want.
to confirm. The trick is situating it into a larger social and cultural context: is it representative? Unique?

In this paper, “Lost in the Infinite Archive,” I explore what the future of digital methods for historians will be when they need to explore web archives. Historical research of periods beginning in the mid-1990s will need to use web archives, and right now we are not ready. This article draws on first-hand research with the Internet Archive and Archive-It web archiving teams. It draws upon three exhaustive datasets: the large Web ARChive (WARC) files that make up Wide Web Scrapes of the Web; the metadata-intensive WAT files that provide networked contextual information; and the lifted-straight-from-the-web guerilla archives generated by groups like Archive Team. Through these case studies, we can see – hands-on – what richness and potentials lie in these new cultural records, and what approaches we may need to adopt. It helps underscore the need to have humanists involved at this early, crucial stage.

**Keywords:** archive; world wide web; historical studies; webscraping; digital history

The Web is having a dramatic impact on how we research and understand the recent past. Historians, who have long laboured under conditions of source scarcity—we wish we had more information about the past, but it was not recorded or preserved—are now confronted with primary sources on a scale that defies both conventional methodologies and standard computational methods.¹ Web archives offer profound promise. Take a comparative example. The Old Bailey Online describes its holdings of 197,745 trials between 1674 and 1913 as the ‘largest body of texts detailing the lives of non-elite people ever published’.² The web archive of GeoCities, a platform for web publishing that operated from the mid-1990s to the early 2000s, amounts to over 38 million pages. Eventually, historians will have access to billions of such sources written by people of various classes, genders, ethnicities, and ages. While the World Wide Web is not a perfect democracy, by any means and any of the categories listed above, it still represents a massive shift. As a result, web archives exemplify this conundrum and represent challenge as well as opportunity.

What information do we want to access? How was the information collected? How do national boundaries intersect with the realm of the Internet? What are the implications of working with such large archives, collected without the informed consent or even knowledge of the overwhelming majority of contributors? These are pressing concerns. For the most part, historians cannot write histories of the 1990s unless they use web archives: with them, military historians will have access to the voices of rank-and-file soldiers on discussion boards; political historians, to blogs, the cut and thrust of websites, electoral commentary and
beyond; and of course, social and cultural historians, to the voices of the people on a scale never before possible.

The stakes are high. If we do not come to graps with web archives, the histories that we write will be fundamentally flawed. Imagine a history of the late 1990s or early 2000s that draws primarily on print newspapers, ignoring the revolution in communications technology that fundamentally affected how people share, interact, and leave historical traces behind. Yet even as we use web archives, we need to be cognizant of their functionalities, strengths, and weaknesses: we need to begin to theorize and educate ourselves about them, just as historians have been cognizant of analog archives since the cultural turn. As new discovery methods for finding information in web archives begin to appear, historians need to be ready to participate; otherwise we might not know why one particular response is number one, versus number one million.

The sheer amount of social, cultural, and political information generated and presented almost every day within the web archive since the Internet Archive began collecting in 1996 represents a complex data set that will fundamentally reshape the historical profession. We need to be ready.

ON COMPLEX DATA SETS: THREE DIFFERENT EXAMPLES

This is not an abstract concern: the history of the 1990s will be written soon. While there is no common rule for when a topic becomes ‘history,’ it took less than 30 years after the tumultuous year of 1968 for a varied, developed, and contentious North American historiography to appear on the topic of life in the 1960s. Carrying out ‘recent histories,’ be they of the 1970s or of events only a few years ago, brings with them a host of methodological issues from a lack of historiography, historical participants who can ‘talk back,’ and issues of copyright and privacy. The year 2021 will mark the 30th anniversary of the creation of the first publicly accessible website. Just as media, government, and business radically transformed their practices in the 1990s, historians must do so as well to analyze this information. ‘New media’ is not that new anymore.

Historians run very real risks if they are not prepared. Currently, the main way to access the archived Web is through the Wayback Machine, most notably associated with the Internet Archive. The Internet Archive emerged out of a concern around a ‘digital dark age’ in the mid-1990s, where rapid technological evolution led to fears around whether our heritage was being preserved. Responding to this, Internet entrepreneur Brewster Kahle founded the Internet Archive in June 1996, which began to rapidly grow their web archive collection. They did so by sending ‘web crawlers,’ automated software programs, out into the Web to download webpages that they found. This crawling process meant that depending on how the Web developed and the limits placed on a crawler, the crawler could indefinitely collect – generating an infinite archive.
While the Internet Archive was collecting data from 1996 onwards, the next step was to make it accessible to researchers. In 2001, they launched the still-dominant form of interacting with web archives: the Wayback Machine. You can try it yourself at http://archive.org/web. It is limited. A user needs to know the exact Uniform Resource Locator (URL) that they are looking for: a website like http://www.geocities.com/enchantedforest/1008/index.html, for example. The page is then retrieved from the web archive and displayed. If you know the URL of the page you are interested in, and only want to read a few, the Wayback Machine works by generating facsimiles of those pages. They are not perfect, as they may not collect embedded images, or might grab them at slightly different times (to avoid overloading any single server, the crawler might download the text of a website and then the image a few hours or even days later; this can lead to the storing of websites that never existed in the first place). Beyond technical issues, it is difficult to find documents with the Wayback Machine unless you know the URL that you want to view.

This latter shortcoming disqualifies it as a serious research tool unless it is paired with a search engine of some kind. Historians are used to full-text search interfaces. However, imagine conducting research through date-ordered keyword search results, carried out on billions of sites. It would produce an outcome similar to the current methods by which historians search digitized newspapers. In the absence of contextual information about the results found, they can be useless. It is possible to find almost anything you want within 38 million web pages. I can find evidence on any matter of topics that advances one particular argument or interpretation. Without the contextual information provided by the archive itself, we can be misled.

Three case studies can help us better understand the questions, possibilities, and challenges facing historians as we enter this archival territory. The first is the Wide Web Scrape, a compilation of billions of objects collected by the Internet Archive between 9 March and 23 December 2011. Next, I explore work that I have been doing with a collection of political websites created between 2005 and 2015. Finally, I explore the GeoCities end-of-life torrent, to get at the heart of ethical challenges.

Together, these studies suggest a path forward for historians. Those of us who use web archives do not need to become programmers, but do need to become aware of basic Web concepts: an understanding of what metadata is, how the Web works, what a hyperlink is, and basic definitional concepts such as URLs. Beyond this, however, is the crucial dimension of algorithmic awareness. When we query archives, we need to know why some results are coming to the top and others at the bottom. If we turn our research over to black boxes, the results that come from them can reaffirm biases: websites belonging to the powerful, for example, rather than the marginalized voices we might want to explore and
consider. The decisions that we as historians make now will have profound effects as tools begin to be developed to access web archives.

**DATA IS BIGGER THAN THE NATION: THE WIDE WEB SCRAPE**

As a data set, the Wide Web Scrape is exhaustive, transcending national borders. The 2,713,676,341 item captures—web sites, images, PDFs, Microsoft Word documents, and so forth—are stored across 85,570 WebARChive (WARC) files. The WARC file format, which is certified by the International Standards Organization, preserves web-archived information in a concatenated form. Generated by the Internet Archive, these files also serve as a good introduction to the geographic challenges of web archives: historians tend towards geographic boundaries, but these archives can transcend them. WARC files are an abundant resource, but that abundance is double edged.

As a Canadian historian looking for a relatively circumscribed corpus, I decided to focus on the Canadian Web, or websphere, as best I could. The ‘Canadian Web’, is however, intrinsically a misnomer. The Web does not work within national boundaries. It is a global network, transcending traditional geopolitical barriers (local fissures still appear, as seen in ‘this video is not available in your country’ messages). The Internet Archive exploits the Web’s borderless nature in their global crawling of material in a way national domain crawls by national institutions cannot. From Denmark to Britain, researchers collecting and studying national webspheres have taken different approaches. Some, such as the Danish NetLab, have confined their studies to national top-level domains (.dk). Others, such as the British Library’s born-digital legal deposit scheme, use algorithms and human intervention to find British sites outside of the .uk domain.

What does the data collected along the lines of a national websphere—a top-level domain such as .ca—look like? While all archival records are only as useful as the discovery tools that accompany them—a misfiled box in a conventional archive might as well not exist—the size of these collections elude traditional curation. From the holdings of the Wide Web Scrape, we examined the CDX files (akin to archival finding aids which contain information about the records found within archival boxes), and which can be measured in gigabytes rather than terabytes. They contain millions of lines of text like:

```
```
From this, we can learn a few things: in this case, we learn that the record is justlabour.yorku.ca, collected on 14 July 2011 at 7:37 GMT. It redirected (HTML code 302) to the table of contents for volume 16. If you visit justlabour.yorku.ca today, you’ll be redirected to a more recent issue. CDX files help us find specific records. Accordingly, I used them to download a sample of 622,365 .ca URLs.

Working with this data set was an interesting window into the choices historians need to make when they work with large data sets from the Web. Derived data—plain text, named entities (discussed later), extracted links, hyperlinks with anchor text—can be useful. Yet at every stage they present historians with questions. Some extracted hyperlinks will be relative—that is, /destination.html rather than http://www.history.ca/destination.html. Should they be reclassified if we want to make a chart of all the hyperlinks connecting different websites, and at what stage? To create plain text files, we use the warcbase platform. Working with this data set was an interesting window into the choices historians need to make when they work with large data sets from the Web. Derived data—plain text, named entities (discussed later), extracted links, hyperlinks with anchor text—can be useful. Yet at every stage they present historians with questions. Some extracted hyperlinks will be relative—that is, /destination.html rather than http://www.history.ca/destination.html. Should they be reclassified if we want to make a chart of all the hyperlinks connecting different websites, and at what stage? To create plain text files, we use the warcbase platform. Working with this data set was an interesting window into the choices historians need to make when they work with large data sets from the Web. Derived data—plain text, named entities (discussed later), extracted links, hyperlinks with anchor text—can be useful. Yet at every stage they present historians with questions. Some extracted hyperlinks will be relative—that is, /destination.html rather than http://www.history.ca/destination.html. Should they be reclassified if we want to make a chart of all the hyperlinks connecting different websites, and at what stage? To create plain text files, we use the warcbase platform.

There were three promising ways to query this data, each of which sheds light on various web archival challenges: keywords, named entity recognition (which finds entities like locations and names within text), and hyperlink structures. To search a large body of material with keywords, the Apache Solr search engine is ideal. It can index material and respond to queries from a number of front-ends that can run locally on a computer. The United Kingdom’s Web Archive, for example, uses a custom front-end Solr portal that provides full-text search access to their collections. One view prompts you to enter a query, and then subsequently see the relative frequency of that term rise and fall over time (how often was the word ‘nationalize’ used in 2006, for example, compared to 2012). With specific queries, this search approach works well. Yet on a broad scale, when looking for cultural trends, more context is necessary.

The most promising keyword approach to my data set was clustering, which takes a set of documents and groups them. If a web collection contained websites about cats, dogs, and pigs, the algorithm might cluster the cat sites together. Conversely, it might find another characteristic—the ages of the authors, perhaps—and cluster them that way. There are several different algorithms to choose from, although in my experience the Lingo clustering algorithm provides the best results (See Fig. 1). The free Carrot2 front end (http://project.carrot2.org/), which interfaces easily with a Solr database, is the most useful. From a query for ‘children’, we see that this sample of 622,365 websites contains pages relating to child health, health
Ian Milligan

Figure 1. Carrot2 clustering workbench results.

Figure 2. Example of connected clusters.

centres, service providers, public health, educational services, and consumer products such as Tylenol. Clicking on the graphical representation brings the user to a list of documents, and another click brings up an individual document. The image on the right is the graphical representation of overlapping clusters, such as the simplified Figure 2.

If a dot is connected to two clusters, it belongs to both. These connections can provide a rough sense of how representative things are: there are many websites about breastfeeding, for example, but not many about Christian early childhood education institutions. More importantly, it is possible to isolate a corpus to study. Used jointly, the Solr database and Carrot2 front end help transcend the Wayback Machine’s limitations.

The main drawback with this approach is the need to know what you are looking for. Extracting commonly mentioned locations can be fruitful, as in Figure 3.

Extracted using a combination of Stanford Named Entity Recognition (NER), Google Maps API, and verification by student research assistants, this
process found location names— for example, ‘Toronto’ or ‘Johannesburg’— and geolocated them by assigning coordinates. While longitudinal data will be more useful, allowing us to see how various locations changed over time, at this point we can see the attention paid towards Canadian trading partners and the complete absence of attention towards sub-Saharan Africa. Within Canada, Québec is overrepresented vis-à-vis the province of Ontario.

Web-wide scrapes represent the dream of social history: a massive documentary record of the lives of everyday people, their personal websites, small businesses, labour unions, community groups, and so forth. Yet the value of this information is balanced by the sheer size and complexity of these data sets. Web-wide scrapes are one extreme of what we can do with web archives: exploring a massive record of human activity, collected on a previously unimaginable scale.

ARCHIVE-IT POLITICAL COLLECTIONS: AN IDEAL SIZE?

Web-wide scrapes are time consuming and expensive to work with. Recognizing this, web archivists have begun to move towards more accessible data sets that bridge the gap between the lightweight CDX file and the heavy-duty WARC file (both of which we have seen in the preceding section). In this section, I argue that while our first inclination, as with the Wide Web Scrape, might be to go right to the content, more fruitful historical information can be found within the metadata.

Archive-It, a web archiving subscription service provided by the Internet Archive for universities and other institutions, recently piloted their research services portal. It provides access to Web Archive Transformation, or WAT, files: a happy medium between CDXs and WARCs. These provide rich metadata: everything that a CDX has, plus metatext about the website, the title, and the links and anchor text from each site. They are essentially the WARCs sans content, making them much smaller.
Beginning a decade ago, the University of Toronto Library (UTL) has put together thematic web collections with Archive-It. One of their major collections is about Canadian political parties and political interest groups, collected quarterly since 2005. Canada has seen pivotal changes within its political sphere over the last ten years, between 2005 and 2015: an arguable militarization of Canadian society, the transition from the ‘natural governing party’ of the centrist Liberal Party of Canada to the Conservative Party of Canada (and back in late 2015), as well as major policy changes on foreign policy, science policy, and climate change. Given these critical shifts, it is surprising on one level that UTL’s collection was not used more—the collection, for example, has never been cited before we began to work with it. On another level, however, it is unsurprising: the current portal to work with the collection at https://archive-it.org/collections/227 has only a very basic search function. It was only by reaching out to librarians at UTL and the Internet Archive that I was able to get the files and begin to explore what we could actually do with them. Ultimately, it became clear that metadata was just as—and in many cases more—useful than the content itself (we ended up providing access to the content through http://webarchives.ca, an implementation of the British Library’s Shine frontend).

By using either the Internet Archive’s web analysis workshop or warcbase, a web archiving platform, we can extract links the WAT files in this collection by domain. The results look similar to the example in Table 1.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Weight (number of links)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conservative.ca</td>
<td>Liberal.ca</td>
<td>10</td>
</tr>
<tr>
<td>Liberal.ca</td>
<td>NDP.ca</td>
<td>10</td>
</tr>
</tbody>
</table>

This data can be useful on a large scale. Consider Figure 4, which visualizes the external links stemming from and between the websites of Canada’s three main political parties. Each line, or edge, represents a hyperlink between domains (or nodes).
Above, we can see which pages only link to the left-leaning New Democratic Party (NDP or ndp.ca), those that link only to the centrist Liberals (liberal.ca) in the top, and those that only connect to and from the right-wing Conservative Party at right. In the middle are the websites that either link to all three parties or to just two of the three (to the left and right of the Liberal node, respectively). Even from this graph we can see that while many groups link to only the Liberals and the NDP, or to the Liberals and the Conservatives, few link just to the NDP and the Conservatives.

By taking quarterly slices of the data, we can also use metadata to identify the broad contours of a narrative as in Figure 5.

We can see that several entities link to all three parties, such as the environmentalist davidsuzuki.org or the Assembly of First Nations (afn.ca), and we can also see how all of the organizations linked to each other. The Liberal Party was then in power and was under attack by both the opposition parties. In particular, the left-leaning NDP linked hundreds of times to their ideologically close cousins, the centrist Liberals, as part of their electoral attacks, ignoring the
right-leaning Conservative Party in the process. Link metadata illuminates more than a close reading of an individual website would.

We can also find sections of this collection that link far more to themselves than to other parts. These divisions lend themselves well to specific extraction. Consulting the UTL’s entire collection via WARC files may be too difficult, but link analysis can tell us what to download. One experiment proved interesting. I took the two main political parties, the Liberals and Conservatives, over the period of study and (relying solely on links) found the communities that grew out of their party websites. The results were interesting: liberal.ca was in the same community as interest groups such as the National Association of Women and Law and media organizations such as *Maclean’s* magazine and the Canadian Broadcasting Corporation. Most interestingly, the left-wing New Democratic Party of Canada appeared in the same community. For the Conservatives, they were grouped with many cabinet ministers’ pages, but also with groups such as Consumers First, which fought for price parity between Canada and America.

By extracting some of these pages and topic modeling the results, we can confirm existing narratives and raise new questions. Topic modeling finds

---

**Figure 5.** Link structures during the lead-up to the 2006 federal election.
‘topics’ in text. For example, imagine that I am writing about women in a male-dominated labour movement. When I write about the women, I use words like ‘femininity’, ‘equity’, ‘differential’, and ‘women’. Men: masculinity’, ‘wildcat’, or ‘foremen’. In this thought experiment, imagine I am drawing these words from buckets full of slips of paper. Topic modeling reverses that process, putting those words back into the bucket and telling me what is in it. It is a quick way to get a sense of what might be happening in a large body of text.\textsuperscript{18}

Taking the link community that appeared around political parties, we were able to find topics most closely connected to them. In December 2014, the Liberals were highlighting cuts to social programs, issues of mental health, municipal issues, housing, and their new leader, Justin Trudeau (now, as of October 2015, the new Prime Minister of Canada). The Conservatives: Ukraine, the economy, family and senior issues, and the high-profile stimulus-based Economic Action Plan. For 2006, the results were surprising. The Liberals: community questions, electoral topics (given the federal election), universities, human rights, childcare support, and northern issues. The Conservatives: some education and governance topics, but notably, several relating to Canada’s aboriginal population. While the Liberals had advanced a comprehensive piece of legislation designed to improve the conditions of Canada’s aboriginal population, Conservative interest in the topic was surprising: perhaps it reflects the Conservative opposition to it? As one commenter on an earlier draft suggested, it may represent the influence of key advisors, one of whom was a leading Conservative scholar of native-newcomer relations. Questions are raised, suggesting promise in marrying content and metadata in such a manner.

A PLACE OF THEIR OWN: EXPLORING THE ETHICAL MINEFIELD OF GEOCITIES

In general, the sheer scale of distantly reading millions of websites or exploring the public record of political parties has kept us in the previous cases removed from everyday individuals. As the Web became mainstream in the mid-to-late 1990s, GeoCities played a critical role. For the first time, users could create their own web pages without learning HTML or FTP. On sites like GeoCities, they could become part of part of virtual communities, held together by volunteers, neighbourhood watches, web rings, and guestbooks. Even though in 1999 GeoCities was perhaps the third most popular website in existence, Yahoo! deleted it in 2009. Dedicated teams of Internet archivists, such as Archive Team (http://archiveteam.org), created the web archive that we can use today. It is large: at its peak, GeoCities had around 38 million pages.

GeoCities began in late 1994 as a service predicated on geospatial metaphors and giving voices to those who ‘had not had an equal voice in society’.\textsuperscript{19} Users could easily create new sites within an existing GeoCities community,
such as the Enchanted Forest for children or Area 51 for science fiction fans. They received an ‘address’ based on their neighbourhood: www.geocities.com/EnchantedForest/1005/index.html. In an era when the Web was understood as a new ‘frontier’, this claim to an actual address resonated. User numbers skyrocketed, from 1,400 in July 1995 to 100,000 by August 1996 and a million by October 1997.

I have been exploring the question of how community was created and enacted there. A significant minority of users threw themselves into the site. When a user arrived to create their site, they had to choose where to live: a small ‘cottage’ in the Enchanted Forest, perhaps, or a ‘tent’ in Pentagon. Reminders exhorted them to fit into the site’s theme, reach out to neighbours, and crucially – in a move reminiscent of the American 1862 Homestead Act – ‘move in’ and improve their property within a week. Some users became community leaders, welcoming new arrivals and teaching them the ropes. An awards economy boomed, with users creating their own awards and giving them to other sites. They visited each other’s guestbooks. Messages are disproportionately from GeoCities users rather than visitors from outside. This community structure persisted until 1999, when Yahoo! bought GeoCities and turned it into a conventional web host.

Like in the previous section, we can explore neighbourhoods with topic modelling. We can see topics in the Enchanted Forest about parties, friends, soldiers and children’s characters such as Pingu. In Heartland, topics relating to family, church, and genealogy appear, and in the LGBT-focused WestHollywood, the focus is on gender, transgender issues, and fighting against hate crimes. Over time, the topics discussed in some neighbourhoods changed. Pentagon moved beyond being a hub for deployed and constantly moving service people towards serving as a forum for political discussions and military history. Heartland came to advance a vision of family focused on Christianity and genealogy. These findings demonstrate that neighbourhoods both shaped and were shaped by user contributions.

How did this come to be? By extracting links, we can begin to find the central nodes that dozens or even hundreds of other websites linked to, as well as the web of connections that held everybody together. This gives us a few hundred websites per neighbourhood to investigate: the community leaders who received kudos from their members, sites that accumulated awards, those with active guestbooks. These factors produced many hyperlinks, both in and out, making these sites critical nodes.

Websites like GeoCities raise ethical questions. Unlike in our previous case studies, which dealt with institutional websites, in GeoCities we are dealing with largely personal websites from over a decade ago. The majority of these people almost certainly did not create these sites with a future historian in mind, nor are they likely to be aware that their sites live on within the Internet Archive or the Archive Team torrent. They did not give consent to the archiving
of their sites, nor did they have access to a robots.txt file that could have changed access parameters (see http://archive.org/about/exclude.php). Indeed, unless they remember their URL, users cannot see if their sites were archived in order to pursue their removal from the archive. Traditional archival collections often have restrictions: donor requests, privacy legislation, or the protection of personal information on medical, financial, or other grounds. While historians have ethical responsibilities at all times, in many cases the onus of making a collection available and accessible lies with institutions. Oral historians, on the other hand, operate outside traditional institutions, instead working in the personal spaces of their interviewees. Institutional review boards, committees that oversee how human subjects are used in research within most North American contexts, govern their work. While none of the above is simple, it is well-travelled ground. Where do web archives fall between these poles?

Strictly speaking, as we generally treat websites as ‘publications’, it is legal to quote from tweets, blogs, websites, and so forth. Legal does not equal ethical, though. As Aaron Bady notes, "The act of linking or quoting someone who does not regard their twitter as public is only ethically fine if we regard the law as trumping the ethics of consent." We need to consider user privacy expectations, which is at the heart of the distinction between a political candidate’s site and a GeoCities homestead. This is not to treat users as dupes but to recognize that somebody posting a website in an obscure corner of GeoCities might have an expectation of privacy: many of these sites would not have been discovered by regular users but are easily discovered by web crawlers methodically crawling a community structure.

We can find guidance from web scholars. danah boyd, a web scholar, notes that students with open Facebook profiles regarded a teacher visiting their page as a breach of privacy, social norms, and etiquette. The Association of Internet Researchers provides guidance that has researchers consider the public or private nature of the website and the differences between dealing with sources en masse versus individually. Stine Lomberg has emphasized the importance of distance but also, when exploring content, of considering user expectations of privacy.

Historians need to consider these factors when deciding how to appropriately use this material. Some GeoCities cases bring these questions into perspective. Memorial sites, by people who lost children or other loved ones, are both private and intimate but also have well-travelled guestbooks, often by people who lost loved ones of their own. Other searches bring up pages about suicide or depression. These can only be found thanks to today’s modern discovery tools. If a 15-year old wrote to the government with a rant, privacy legislation would excise her or his name; if you find the rant in GeoCities, the name—or their pseudonym (which can sometimes be connected to real names)—would be there. These are resources that would never make it into a traditional archive.
We have power because we can access the blogs, ruminations, and personal moments of literally millions of people that would never before have been accessed—but we need to use this power responsibly. With the Wayback Machine, the lack of full-text search provides some privacy, but as we undertake more computational inquiries historians can uncover things forgotten since their creation. My own take on this question is twofold, drawing on earlier literature: we need to consider the scale at play. Mining a few thousand sites and dealing with—and writing about—people in aggregate presents few privacy concerns, whereas zooming in on a handful of websites and closely reading them does. A website many other sites connect to, a proud prominent view counter in the corner (or other equivalent markers of popularity that have supplanted this now dated approach), a well-travelled guestbook, signals a website of an owner who wanted to be read and encountered, and who conceived of themselves as part of a broader Web of documents. A smaller website addressed to an internal audience, written by a teenager and full of revealing messages and pictures, is a different thing altogether.

GeoCities represents a new kind of primary source: the largely non-commercialized, unfettered thoughts of millions of everyday people in the mid-to-late 1990s, left for historians today. We can learn invaluable things, from the forms online community took on the Web to the opinions and thoughts on a host of social, political, or cultural issues or topics.

CONCLUSIONS

These three disparate web archiving case studies all demonstrate the critical questions that lie at the heart of these new complex data sets. The technical challenges are clear: not enough processing power or computer memory, the need to find access to a computing cluster, and the variety of file formats and types that underlie them. Rather than a narrow-lens pedagogical approach that stresses say the WARC file, historians who want to use these sources—arguably a necessity when undertaking topics in the 1990s and beyond—need to have a flexible understanding of software and standards.

While this article has focused on the research process, further issues will emerge when scholars attempt to publish this type of work. Images, already a sticking point with many publishers, are borrowed, altered, shared, throughout the Web: can one publish a notable image found in a 1996-era web archive if this has no contactable author or even real name? How can we share our research data with each other if we need to worry about digital rights? How do we balance global copyright regimes with the local contexts of journals and academics? At the least, pedagogical training in copyright is needed, as well as advocacy around orphan works and strengthening fair dealing/use.
Despite these challenges and cautions, which need to be heeded as we move forward, I want to return to the original promise articulated at the beginning of this paper. Each of these case studies, from the Wide Web Scrape to the political movements archive to GeoCities, presents promise. They provide more voices from a more diverse body of people, furthering the goals of social historians to write their histories from the bottom up, to move our stories away from the elites and dominant players of society to the everyday. Web archives are not going to have a slight impact on the practice of history: they are going to force a profound shift. We will have more sources than ever before, by people who never could have conceivably reached large audiences or had their words recorded. We should be optimistic, but we need to be prepared.

END NOTES

3. For examples from the Canadian context, see C. Levitt, Children of privilege: student revolt in the sixties: a study of student movements in Canada, the United States, and West Germany (Toronto, 1984) or D. Owram, Born at the right time: a history of the baby boom generation (Toronto, 1997).


20 F. Turner, From counterculture to cybersculture: Stewart Brand, the Whole Earth network, and the rise of digital utopianism (Chicago, 2008).


This first part of this book focuses on image core knowledge essential for image loading. It includes discussion on color theory, image types, formats, and the capabilities. Unfortunately there isn’t a single solution for digitally encoding images. Understanding these complexities and the many uses cases is important before addressing image loading. Depending on your familiarity with these subjects, it might be tempting to skip over some chapters and jump straight to Part II. Don’t feel bad. These chapters are intended to be used as reference and help you navigate the complexities of bringing high quality images to your users.
Images are an essential part of human history. Film-based photography has made the creation of images easy — it captures a moment in time by allowing light to go through a lens and hit film, where an array of minuscule grains of silver-based compound that change their brightness as a response to light intensity.

With the advent of computers, the digitization of photos soon followed, initially by scanning printed images to digital formats, then followed by digital cameras prototypes.

Eventually, commercial digital cameras started showing up alongside film-based ones, and ended up replacing them in the public’s eye (and hand). Camera phones also contributed to that, with most of us now walking around with high resolution digital cameras in our pockets.

The digital camera was very similar to the film-based one, only they had a matrix of light sensors replacing the silver grains in capturing light beams. These photosensors then send electronic signals representing the various colors captured to the camera’s processor, which stores the final image in memory as a bitmap — a matrix of pixels — before usually converting it to a more compact image format. This kind of image is usually referred to as a photographic image, or even more commonly, a photo.

But that’s not the only way to produce digital images. Humans wielding computers can create images without capturing any light by manipulating graphic creation software, capturing screenshots or many other means. We usually refer to such images as computer generated images or CGI.

This chapter will discuss digital images and the theoretical foundations behind them.
Digital image basics

In order to properly discuss digital images and the various formats throughout this book, some familiarity with the basic concepts and vocabulary is required.

We will discuss sampling, colors, entropy coding, and the different types of image compression and formats. If this sounds daunting, fear not. This is essential vocabulary that we need in order to dig deeper and understand how the different image formats work.

Sampling

We learned earlier that digital photographic images are created by capturing light and transforming it into a matrix of pixels. The size of the pixel matrix is what we refer to when discussing the image's dimensions — the number of different pixels that compose it.

If we look at light before it is captured, it is a continuous, analog signal. In contrast, a captured image of that light is a discrete, digital signal. The process of conversion of the analog signal to a digital one involves sampling, when the values of the analog signal are sampled in regular frequency, producing a discrete set of values.

Our sampling rate is a tradeoff between fidelity to the original analog signal and the amount of data we need to store and submit. Sampling plays a significant role in reducing the amount of data digital images contain, enabling their compression. We'll expand on that later on.

Image Data Representation

The simplest way to represent an image is by using a bitmap — a matrix as large as the image's width and height, where each cell in the matrix represents a single pixel and can contain its color for a color image or just its brightness for a grayscale image. Images that are represented using a bitmap (or a variant of a bitmap) are often referred to as raster images.

Figure 2-1. To the left, a continuous signal. To the right, a sampled discrete signal.
But how do we digitally represent a color? To answer that we would need to get familiar with…

**Color spaces**

We’ve seen above that a bitmap is a matrix of pixels, and each pixel represents a color. But how do we represent a color using a numeric value?

In order to dive into that, we’ll need to take a short detour to review color theory basics. Our eyes are built similarly to the digital camera we discussed earlier, where the role of photosensitive electronic cells is performed by light sensitive pigmented biological cells called rods and cones. Rods operate in very low light volumes and are essential for vision in very dim lighting, but play almost no part in color vision. Cones on the other hand, operate only when light volumes are sufficient, and are responsible for color vision.

Humans have three different types of cones, each one responsible for detecting a different light spectrum, and therefore, for seeing a different color. These three different colors are considered primary colors: red, green and blue. Our eyes use the colors the cones detect (and the colors they don’t detect) to create the rest of the color spectrum that we see.
One more interesting characteristic of human vision is that its sensitivity to light changes is not linear across the range of various colors. Our eyes are significantly more sensitive when light intensity is low (so in darker environments) than they are when light intensity is high. That means that humans notice changes in darker colors far more than they notice changes in light colors.

Cameras capture light differently. The intensity of light that they capture is linear to the amount of photons they get in the color range that they capture. So, light intensity changes will result in corresponding brightness changes, regardless of the initial brightness.

That means that if we represent all color data as captured by our cameras using the same number of bits per pixel, our representation is likely to have too many bits per pixel for the brighter colors and too few for the darker ones.

A process called **Gamma Correction** is destined to bridge that gap between linear color spaces and “perceptually linear” ones, making sure that light changes of the same magnitude would be equally noticeable by humans, regardless of initial brightness.

![Figure 2-3. A view of winter-time French countryside, Gamma corrected on the left and without Gamma correction on the right.](image)

**Additive vs. Subtractive**

There are two types of color creation: additive and subtractive. Additive colors are colors that are created by a light source, such as a screen. When a computer need a screen’s pixel to represent a different color, it **adds** the primary color required to the colors emitted by that pixel. So, the “starting” color is black (absence of light) and other colors are added until we reach the full spectrum of light, which is white.

Conversely, printed material, paintings and non-light-emitting physical objects get their colors using a subtractive process. When light from an external source hits these materials, and only some light wavelengths are reflected back from the material and
hit our eyes, creating colors. Therefore, for physical materials, we often use other primary subtractive colors, which are then mixed to create the full range of colors. In that model, the “starting” color is white (the printed page), and each color we add subtracts light from that, until we reach black when all color is subtracted.

As we can see from the above, there are multiple ways to recreate a sufficient color range from the values of multiple colors. These various ways are called color spaces. Let’s describe some of the common ones.

Figure 2-4. Additive colors created by light vs. subtractive colors created by pigments.

**RGB (Red, Green & Blue)**

RGB is one of the most popular color spaces (or color space families). The main reason for that is that screens, which are additive by nature (they emit light, rather than reflect light from an external light source), use these three primary pixel colors to create the range of visible colors.

The most commonly used RGB color space is sRGB, which is the standard color space for the W3C, among others. In many cases, it is assumed to be the color space used for RGB unless specified otherwise. Its gamut (the range of colors that it can represent) is more limited than other RGB color spaces, but it is considered a baseline that all current color screens can produce.
CMYK (Cyan, Magenta, Yellow & Key)

CMYK is a subtractive color space which is most commonly used for printing. The “Key” component is simply black. It has a wider gamut than sRGB, so it can show more colors, especially in the green-blue hues. Instead of having three components for each pixel as RGB color spaces do, it has four components. The reasons for that are print-related practicalities. While in theory the black color could be achieved in the subtractive model by combining cyan, magenta and yellow together, in practice the outcome black is not “black enough”, long to dry, and too expensive. Since black printing is quite common, that resulted in a black component being added to the color space.

YCbCr

YCbCr is actually not a color space on its own, but more of a model that can be used to represent gamma corrected RGB color spaces. The “Y” stands for gamma corrected luminance (the brightness of the sum of all colors), “Cb” stands for the chroma component of the blue color and “Cr” stands for the Chroma component of the red color.

RGB color spaces can be converted to YCbCr using a fairly simple mathematical formula.
One advantage of the YCbCr model over RGB is that enables us to easily separate the brightness parts of the image data from the color ones. The human eye is more sensitive to brightness changes than it is to color ones, and the YCbCr color model enables us to harness that to our advantage when compressing images. We will touch on that in depth later in the book.

YCgCo

YCgCo is conceptually very similar to YCbCr, only with different colors. Y still stands for gamma corrected luminance, but Cg stands for the green chroma components and Co stands for the orange chroma component.
YCgCo has a couple of advantages over YCbCr. The RGB⇔YCgCo transformations are mathematically (and computationally) simpler than RGB⇔YCbCr. On top of that, YCbCr transformation tends to lose some data due to rounding errors, whereas the YCgCo transformations do not, since they are “friendlier” to floating point fractional arithmetic.

\[
\begin{bmatrix}
Y \\
C'g \\
C'o
\end{bmatrix} = \begin{bmatrix}
1/4 & 1/2 & 1/4 \\
-1/4 & 1/2 & -1/4 \\
1/2 & 0 & -1/2
\end{bmatrix} \cdot \begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

Figure 2-8. Formula to convert from RGB to YCgCo. Note the use of powers of 1/2, which makes this transformation easy to compute and float-friendly.

There are many other color spaces and models, but going over all of them is out of the scope of this book. The color models above are all we need to know in order to further discuss images on the web.

Figure 2-9. Winter-time French countryside. Top to bottom, left to right: Full image, Y component, Cg component and Co component.
Bit Depth

Now that we've reviewed different color spaces, which can have a different number of components (three for RGB, four for CMYK) let's address how precise each of the components should be.

Color spaces are a continuous space, but in practice, we want to be able to define coordinates in that space. The unit measuring the precision of these coordinates for each component is called **bit depth** — it's the number of bits that you dedicate to each one of your color components.

What should that bit depth be? Like everything in computer science, the correct answer is “it depends”.

For most applications, 8 bits per component are enough to represent the colors in a precise enough manner. In other cases, especially for high fidelity photography, more bits per components may be used in order to maintain color fidelity as close to the original as possible.

Color profiles

How does the encoder know which color space we referred to when we wrote down our pixels? That's where something called color or ICC (International Color Consortium) profiles come in.

These profiles can be added to our images as meta data and help the decoder accurately convert the colors of each pixel in our image to the equivalent colors in the local display's “coordinate system”.

In case the color profile is missing, the decoder cannot perform such conversion, and as a result, its reaction varies. Some browsers will assume that an image with no color profile is in the sRGB color space and will automatically convert it from that space to the local display's color space. At the same time, other browsers will send the image's pixels to the screen as they are, effectively assuming that the color profile the images were encoded in matches the screen's. That can result in some color distortion, so in case color fidelity is important, color profiles are essential for cross-browser color correctness.

On the other hand, adding a color profile can add a non-negligible number of bytes to your image. A good tradeoff is probably to make sure your images are in the sRGB color space and add a **fairly small sRGB** color profile to them.

We will discuss how you can manage and control your images’ color profiles more in the Operationalizing Your Image Workflow chapter.
**Alpha**

We discussed all the possible options we have to represent colors, but we left something out. What about the possibility to represent lack of color?

In some cases we want parts of our image to be transparent or translucent, so that our users will see a non-rectangular image, or otherwise will be able to see through the image onto its background.

![Figure 2-10. An image with an alpha channel over different backgrounds. Note the different colors of the dice edges.](image)

The representation of the absence of color is called an **alpha channel**. It can be considered as a fourth color, where the zero value means that the other three colors are fully transparent, and a maximal value means that the other three colors are fully visible.

**Frequency domain**

As we now know, we can break our images into three components: one brightness component and two color ones. We can think of each one of these components as a two dimensional function that represents the value of each pixel in the spatial domain, where the X and Y axis are the height and width of the image, and the function's value is the brightness/color value of each pixel.
As such, we can apply certain mathematical transforms on these functions, in order to convert them from the spatial domain into the frequency domain. A frequency domain based representation gives us the frequency in which each pixel value is changing rather than its value. Conversion to the frequency domain can be interesting, since it enables us to separate high frequency brightness changes from low frequency changes.

It turns out that another characteristic of human vision is that we notice high frequency brightness and color changes significantly less than we notice low frequency ones. If brightness or color is changing significantly from one pixel to the next, and then back again, our eye will tend to “mush” these neighbouring pixels into a single area with an overall brightness value that is somewhere in between.

We will expand on how this is done and used when we talk about JPEGs in the JPEG chapter.

**Image Formats**

In the following chapters we will discuss the various image formats that are in common use today. But before we can dive into the details of each one of the formats, let’s explore the slightly philosophical question: What is image compression and why it is needed?

**Why Image-Specific Compression?**

As you may have guessed, image-compression is a compression technique targeted specifically at images. While many generic compression techniques exist, such as gzip, LZW, LZMA, Bzip2 and others, when it comes to raster images, we can often do better. These generic compression algorithms work by looking for repetitions and finding better (read: shorter) ways to represent them.
While that works remarkably well for text and some other types of documents, for most images, that’s not enough. That kind of compression can reduce the number of used bytes for bitmap images that have a lot of pixels of exactly the same color right next to one another. While that’s great, most images — especially those representing real-life photography — don’t exhibit these characteristics.

So, pretty early on, various image compression techniques and related formats began to form and eventually a few formats were standardized upon. Many of these image compression techniques use generic compression techniques internally, but do so as part of a larger scheme that maximizes their benefits.

**Raster vs. vector**

Raster images vs. vector-based ones present the first fundamental divide in regards to image formats and compression techniques. The first fundamental divide we would discuss with regard to image formats and compression techniques is that of raster images vs. vectorial based images.

As previously mentioned, a raster image is comprised from a rectangular matrix called a bitmap. Each value in that matrix is representing the color of a certain pixel that the computer can then copy onto its graphics memory in order for it to be painted to the screen.

Unlike raster, vector images don’t contain the colors of individual pixels. Instead, they contain mathematical instructions that enable the computer to calculate and draw the image on its own.

While vector images can have many advantages over raster images in various scenarios, raster images are more widely applicable. They can be used for both computer generated graphics as well as real life photos, whereas vector images can only be efficiently used for the former.

Therefore, throughout the book, unless specified otherwise, we will mostly be referring to raster images, with the main exception being the SVG and Vector Images chapter.

**Lossy vs. Lossless Formats**

Another characteristic that separates the various formats is whether or not they incur a loss of image information as part of the compression process. Many formats perform various “calculated information loss” in order to reduce the eventual file size.

Quite often that loss in image information (and therefore image precision and fidelity to the origin) is aiming to reduce information that is hardly noticed by the human eye, and is based on studies of human vision and its characteristics. Despite that, it’s
not unheard of for precision loss to be noticeable, which may be more critical for some applications than others.

Therefore, there are both lossy and lossless image formats, which can answer those two different use-cases: image compression while maintaining 100% fidelity to the original vs. compression that can endure some information loss while gaining compression ratio.

**Lossy vs. Lossless Compression**

While the formats themselves can be lossy or lossless, there are various examples where images can undergo lossy as well as lossless compression, regardless of the target format. Metadata that is not relevant to the image’s display (e.g. where the image was taken, camera type, etc.) can be removed from images resulting in arguably lossless compression even if the target format is lossy. Similarly, image information can be removed from the image before it is saved as the target format, resulting in lossy compression of a lossless image format.

One exception to that is that you cannot save an image losslessly in a format that only has a lossy variant. This is because these formats usually apply some degree of loss as part of their encoding process, and that cannot be circumvented.

We will further discuss lossless and lossy compression in the Operationalizing Image Compression chapter.

**Prediction**

Often, the encoding and decoding processes both include some guess of what a pixel value is likely to be, based on surrounding pixel values, and then the actual pixel value is calculated as the offset from the “expected” color. That way we can often represent the pixel using smaller, better compressible values.

**Entropy encoding**

Entropy encoding is very common generic compression technique and is used in order to give the most frequent symbols the shortest representation, so that the entire message would be as compact as possible. Entropy coding is often used in image compression to further compress the data, after the main image specific parts are performed.

Since entropy encoding requires us to know what the most frequent symbols are, there are typically two steps to entropy encoding. The first pass on the data gathers statistics regarding the frequency of words in the data, and a dictionary translating those words into symbols is created from the frequency data. Then the second pass on the data is used to translate the words into shorter symbols using the previously created dictionary.
In some domains, where word frequency is known in advance with a good enough approximation, the first step is skipped and a ready made frequency-based dictionary is used instead. The result is potentially slightly larger data stream, but with the advantage of a single-pass algorithm that is faster and possible to perform on-the-fly.

When compressing content using entropy encoding, the dictionary that was used for the encoding has to be present in the decoder as well. Sending the dictionary data adds a “cost” to entropy encoding that somewhat reduces its benefits.

Other types of entropy encoding permit adaptive encoding, where a single pass over the data is enough. Such encodings count the frequency and assign codes to symbols as they go, but change the code assigned to each symbol as its frequency changes.

**Relationship with Video Formats**

One important thing to keep in mind when discussing image formats is that they share many aspects with video formats. In a way video formats are image formats with extra capabilities, that enable them to represent intermediary images based upon previous full images, with relatively low cost. That means that inside every video format, there’s also an image format that is used to compress those full images. Many of the new efforts in the image compression field come from adopting compression techniques from the video compression world, or by adopting the still image encoding parts (called I-frame encoding) from video formats and building an image format based on that (e.g. WebP and BPG, which we will discuss later on).

**Comparing Images**

Comparing the quality of an image compressed using different settings, different encoders or different formats is not a trivial task when it comes to lossy compression. Since the goal of lossy image compression is achieving quality loss, but one that, to some extent, flies under the radar of most people, any comparison has to take both the visual quality of the image and its eventual byte size into account.

If you’re trying to compare the quality and size of a single image, you can probably do that by looking at the image output of different encoding processes and trying to “rank” the various variants in your head, but that is hardly scalable when you have many images to compare, and it is impossible to automate.

Turns out, there are multiple algorithms that try to estimate just that. They give various “scores” when comparing the compressed images to their originals, giving you the opportunity to tune your compression to the visual impact compression would have on the image, rather than to arbitrary “quality” settings.

**PSNR and MSE**
The Peak Singal-to-Noise Ratio (PSNR) is a metric that estimates the ratio of error introduced by the compression algorithm. It often uses Mean-Square-Error (MSE) in order to do that. In a nutshell, MSE is the average mathematical distance of the pixels in the compressed image from the original one. PSNR calculates that and uses the ratio between the maximum possible pixel value to the MSE in order to estimate the impact of compression on the image.

That method works to estimate divergence from the original, but it's not necessarily tied to the impact of that divergence on the user's perception of the compressed image. As we'll see later on, some formats rely on further compressing parts of the image that are less noticeable by the human eye in order to achieve better compression ratios with little perceived quality loss. Unfortunately, PSNR and MSE don't take that into account, and therefore may be skewed against such formats and techniques.

**SSIM**

Structural Similarity (SSIM) is a metric that tries to take the image's structure into account when calculating the errors in the image. It operates under the assumption that human visual perception is adapted to extract structural information, and therefore deterioration in the structural contents of an image would mean that it would be perceived as a lower quality one.

The algorithm estimates structural changes by comparing the intensity and contrast changes between pixel blocks in both the original and compressed image. The larger the intensity and contrast differences are, the more “structural damage” the compressed image's pixel blocks have sustained.

The result of the algorithm is an average of those differences, providing a score in the range of 0 to 1.

When the result is 1 the compressed image is a perfect replica of the original image, and when it is close to 0, very little structural data have remained.

So when using SSIM for compression tuning, you want to aim at close to 1 values for “barely noticeable” compression, and lower values, if you're willing to compromise image quality for smaller files.

SSIM also has a multi-scale variant (so MS-SSIM), which takes multiple scales of both images into account when calculating the final score.

There's also the Structural Dissimilarity metric (or DSSIM) which is very similar to SSIM, but has an inverse range, where 0 is the perfect score and 1 means that the compressed image has no resemblance to the original.

**Butteraugli**

Butteraugli is a recent visual comparison metric from Google, which aims to be even more accurate than SSIM in predicting perceived image quality. The metric is based
on various anatomic and physiological observations related to the human eye structure.

As a result, the algorithm “supresses” the importance of some colors based on the differences in location and density of different color receptors, calculates frequency domain image errors (while putting more weight on low frequency errors as they are more visible than high frequency ones), and then clusters the errors, as multiple errors in the same area of the image are likely to be more visible than a single one.

It is still early days for that metric, but initial results look promising.

Summary

In this chapter we went through the basic terms and concepts we use when discussing digital images and the various image formats. In the following chapters we will make good use of this knowledge by diving in to the details of what each format does and how it does it.
Earlier in the book, you learned about the difference between lossy and lossless image formats. Lossy image formats lose image information during their compression process—typically taking advantage of the way we perceive images to shave away unnecessary bytes. Lossless image formats, however, do not have that benefit. Lossless image formats incur no loss of image information as part of their compression process.

**GIF (It’s pronounced GIF)**

When it comes to image formats on the web, the Graphic Interchange Format (GIF) may no longer be the king of castle, but it certainly is its oldest resident. Originally created in 1987 by CompuServe, the GIF image format was one of the first portable, non-proprietary image formats. This gave it a distinct advantage over the many proprietary, platform-specific image formats when it came to gaining support and adoption on first Usenet, then the World Wide Web.

The GIF format was established at a time of very limited networks and computing power and many of the decisions on how to structure the format reflect this. Unfortu-nately as we’ll see, this does limit both its ability to portray rich imagery as well as its ability to compress.

**Block by block**

The building blocks of the GIF format are….well, they’re blocks. A GIF file is composed of a sequence of data blocks, each communicating different types of information. These blocks can be either optional or required.
The first two blocks of every GIF file are required, and have a fixed length and format.

**Header block**

First up is the *header* block. The header takes up 6 bytes and communicates both an identifier of the format and a version number. If you were to look at the header block for any given GIF, you would almost certainly see one of the following sequences:

- 47 49 46 38 39 61
- 47 49 46 38 37 61

The first three bytes (47, 49, 46) are the GIF’s signature and will always equate to “GIF”. The last three bytes specify the version of the GIF specification used—either “89a” (38, 39, 61) or “87a” (38, 37, 61).

The first three bytes of the header block translate to “GIF”. The second three bytes either translate to “89a” or “87a” depending on the version of the GIF standard the image is taking advantage of. Generally speaking, image encoders will use the older “87a” for compatibility reasons unless the image is specifically taking advantage of some features from the 89a specification (such as animation).

**Logical Screen Descriptor**

Immediately following the header block is the *logical screen descriptor*. The logical screen descriptor is 7 bytes long and tells the decoding application how much room the image will occupy.

The first values communicate the *canvas width* and *canvas height* and can be found in the first two pairs of bytes. These are legacy values that stem from an apparently belief that these image viewers may render multiple images in a single GIF, on the same canvas. Since the only time in practice that a GIF contains multiple images is if it is animated, most viewers today ignore these values altogether.

By converting the next to a binary number, you get a series of boolean switches to indicate four distinct pieces of data.

The first bit is the *global color table flag*. If the bit is 0, there is no global color table being used in the image. If the bit is one, then a global color table will be included right after the logical screen descriptor.

GIF’s employ *color tables* to help index the color for each pixel in an image. The color table contains the colors in the image, as well as a corresponding index value starting at zero. So if the first pixel of an image is the color green, then in the color table, the color green will have a corresponding index value of 0. Now, whenever the image is being processed and encoded, anytime that color is discovered, it can be represented by the number zero.
GIF’s can feature both a global color table as well as a number of local color tables if multiple images are being used (typically in animation). While the global color table is not required, it is almost always included in the image.

The next three bits are the color resolution. The color resolution is used to help determine the size of the global color table. The formula for the number of entries in the global color table is “$2^{(N+1)}$” where N is equal to the number indicated by in the color resolution bits.

**Understanding palettes**

GIF is a **palette-based** image format; that is, the colors that the image uses have their RGB values stored in a palette table. In the case of the GIF format, each table can hold up to 256 entries. This 256 color limit made a great deal of sense when the GIF format was established—hardware was far less capable than it is today—however it severely limits GIF’s ability to display images that contain much detail.

**Hacking GIF’s color limit**

While GIF’s are restricted to a 256 color palette, it is actually technically possible for you to save a true color GIF. Because the GIF format allows for multiple image blocks, and each of those image blocks can have its own 256-color palette, you can technically layer these blocks on top of each other creating a true color image.

However, keep in mind that sometimes things that sound like a good idea really aren’t. Creating a true color GIF is one of those things. Because of the layering and all those color palettes, the resulting file will be gigantic. In addition, not all image editors even handle multiple image blocks correctly. Put it all together and creating true color GIFs is a better answer to a really technical trivia question than it is an actual approach.

**LZW or the rise and fall of the GIF**

The GIF format boasted a powerful lossless compression algorithm known as Lempel-Ziv-Welch, or more commonly, LZW. This algorithm allowed GIF to improve compression significantly over other lossless formats of the time, while maintaining similar compression and decompression times. This file savings, paired with GIF’s interlace option that allowed a rough version of an image to be displayed before the full image has been transmitted, made GIF a perfect fit for the limited networks and hardware of the web’s early days.

Unfortunately, the same compression algorithm that made it such a great format for the web also directly led to GIF’s fall from grace. As it turns out, the algorithm had
been patented by Unisys. In December of 1994, Unisys and Compuserve announced that developers of GIF-based software (compression tools, etc) would be required to pay licensing fees. As you might imagine, this didn’t sit well with developers and the community at large.

There were many repercussions of this announcement, but none more notable than that it lead to the creation of the PNG image format in early 1995.

The PNG file format

Depending on who you ask, PNG either stands for Portable Network Graphics or, displaying a little bit of recursive humor, PNG not GIF (we programmers have a very finely tuned sense of humor). The PNG format was the communities response to the licensing issues that arose around GIF.

The early goal of creating the format was pretty straightforward: create an open alternative to GIF to avoid all the licensing issues. It didn’t take long for everyone involved to realize that they wouldn’t be able to do this and maintain backwards compatibility in anyway. While everyone loves a seamless fallback, the advantage was that this meant the folks creating the PNG format could be more ambitious in their aims—if they weren’t going to be able to maintain backward compatibility, why not make PNG better in every possible way. For the most part, it would seem, they succeeded.

Understanding the mechanics of the PNG format

PNG’s are comprised of a PNG signature followed by some number of chunks.

PNG Signature

The PNG signature is an 8 byte identifier that remains identical for every single PNG image. This identifier also works as a clever way to verify that the PNG file was not corrupted during transfer (whether over the network or from operating system to operating system). If the signature is altered in anyway, then the file has been corrupted somewhere along the line.

For example, the first value in the PNG signature is “137”—a non-ASCII, 8-bit character. Because it is a non-ASCII character, it helps to reduce the risk of a PNG file being mistakenly identified as a textfile, and vice versa. Since it is 8-bits, it also provides verification that the file was not passed over a 7-bit channel. If it was, the 8th bit would be dropped and the PNG signature would be altered.

The full list of bytes of the PNG signature can be found below:
Table 3-1. PNG Signature Bytes

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>8-bit, non-ASCII character</td>
</tr>
<tr>
<td>80</td>
<td>P</td>
</tr>
<tr>
<td>78</td>
<td>N</td>
</tr>
<tr>
<td>71</td>
<td>G</td>
</tr>
<tr>
<td>13</td>
<td>Carriage-return (CR) character</td>
</tr>
<tr>
<td>10</td>
<td>Line-feed (LF) character</td>
</tr>
<tr>
<td>27</td>
<td>CTRL-Z</td>
</tr>
<tr>
<td>10</td>
<td>Line-feed (LF) character</td>
</tr>
</tbody>
</table>

Chunks

Other than the first 8 bytes that the PNG signature occupies, a PNG file is made entirely of chunks—the building blocks of the PNG format.

Each chunk is comprised of the same set of four components:

1. *Length field:* The length field takes up 4 bytes and refers to the length of the chunk's data field.
2. *Type field:* The type field takes up 4 bytes and indicates to the decoder what type of data the chunk contains.
3. *Chunk data:* The chunk data contains the bytes of data that the chunk is trying to pass along. This can range anywhere from 0 bytes to 2GB in size.
4. *Cyclic Redundancy Code (CRC):* The CRC is a 4 byte check value. The decoder calculates the CRC based on the chunk data and chunk type—the length field is not used in the calculation. If the calculated CRC value matches the 4-byte CRC field included in the chunk, the data has not been corrupted.

**Cyclic Redundancy Code Algorithm**

The actual algorithm used to calculate the CRC makes for pretty dry reading (says the guy writing about the nuances of PNG compression) but if that's your cup of tea, you can find the exact algorithm online.

Ancillary and Critical Chunks

The type field communicates a decent amount of information about the chunk within its four little bytes. Each byte has a designated purpose. In addition, each byte has a
simple boolean value of information that is turned on and off by the capitalization of the character occupying that byte.

The first byte is the **ancillary bit**. Just as with blocks in the GIF format, not all chunks are essential to successfully display an image. Each chunk can either be **critical** (uppercase) or **ancillary** (lowercase). A **critical** chunk is one that is necessary to successfully display the PNG file. An **ancillary** chunk is one that is not—instead it’s purpose is to provide supporting information.

The second byte is the **private bit**. The private bit informs the decoder if the chunk is public (uppercase) or private (lowercase). Typically private chunks are used for application-specific information a company may wish to encode.

The third byte is a reserved bit. Currently this bit doesn't inform the coder of anything other than conformance to the current version of PNG which require an uppercase value here.

The fourth byte is the **safe-to-copy bit**. This bit is intended for image editors and tells the editor whether it can safely copy an unknown ancillary chunk into a new file (lowercase) or not (uppercase). For example, an ancillary chunk may depend on the image data in some way. If this is the case, it couldn't be copied over to a new file in case any of the critical chunks had been modified, reordered, or new critical chunks had been added.

The capitalization means that two chunk types that look nearly identical can be very different. Consider iDATA and IDATA. While they appear similar, the first byte makes them distinct chunk types. iDATA is an ancillary chunk type—it's not essential to properly display the image. IDATA, on the other hand, starts with the first character capitalized indicating that it is a critical chunk type and, therefore, any decoder should throw an error since it will not be able to display the image.

The PNG specification defines four critical chunk types, three of which are required for a PNG file to be valid.

<table>
<thead>
<tr>
<th>Chunk type</th>
<th>Name</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHDR</td>
<td>Image header</td>
<td>Yes</td>
</tr>
<tr>
<td>PLTE</td>
<td>Palette</td>
<td>No</td>
</tr>
<tr>
<td>IDAT</td>
<td>Image data</td>
<td>Yes</td>
</tr>
<tr>
<td>IEND</td>
<td>Image trailer</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The IHDR chunk is the first chunk in any PNG image and provides details about the type of image (more on that in a bit); the height and width of the image; the pixel depth; the compression and filtering methods; the interlacing method; whether the...
image has an alpha channel (transparency) as well as whether the image is truecolor, grayscale or colormapped.

The IDAT chunk contains the compressed pixel data for the given image. Technically, the IDAT chunk can contain up to 2GB of compressed data. In practice, however, IDAT chunks rarely reach that size. Instead, they are broken up into several IDAT chunks. Having smaller IDAT chunks allows the viewer to find the image trailer earlier. This in turn allows them to know the image is valid so that they can make decisions about how to handle the display of the image in question.

Imagine an IDAT chunk that is 2GB of data. As we learned early, each chunk has a CRC that allows the viewer to verify that the data within that chunk is valid and not corrupted. If the IDAT chunk is a full 2GB, the viewer must wait until that entire 2GB has been downloaded before it can find the CRC and verify that the image is in tact. If, instead, that IDAT chunk is split into several smaller chunks, then each chunk can be verified quickly using its CRC. Not only can this speed things up, but it also helps to prevent from the awkwardness that arises when a viewer attempts to display an image only to find too late that the image data is corrupted. As a result, you will more typically find IDAT chunks ranging from 8 to 32 kilobytes.

The final require chunk is the IEND chunk. IEND is as simple as you can possible get when it comes to chunks—it contains no data at all. It's entire purpose is to indicate that there are no more chunks ni the image.

Pairing these three required chunks—IHDR, IDAT, IEND—with a PNG signature gives you the simplest PNG file possible. In fact, these three chunks are all you need to build a truecolor or grayscale PNG file.

However, like its predecessor GIF, PNG's can also take advantage of color palettes. If a color-pallette is being used, then the PNG file also needs to include the PLTE (palette) chunk. The PLTE chunk houses a series of RGB values that may be included in the image.

**Interlacing**

Both GIF's and PNG's have an *interlacing* feature that, similar to the progressive JPEG feature you’ll learn about in the next chapter, enables an image to be rendered quickly as a low-resolution version, and then with each successive pass after that, be progressively filled in. This interlacing approach allows the browser to give the user some sense of the makeup of the image earlier than the typical top-down approach to image rendering.

The GIF approach to interlacing is a one-dimensional scheme; that is, the interlacing is based on horizontal values only, choosing to focus on a single row at a time. GIF's approach to interlacing has four passes. First, every eighth row is displayed. Then, every eigth row is displayed again—this time offset by four rows from the first pass.
For example in an image comprised of eight rows of pixels, pass one would display row one and pass two would display row five.

The third pass displays every fourth row and the fifth and final pass displays every other row. You can see how each row of an image is displayed using GIF interlacing in the diagram below.

In contrast, PNG’s interlacing method is a two-dimensional scheme. Instead of analyzing a single row at a time, PNG’s interlacing method involves looking at the pixels within a row.

The first pass involves filling in every eighth pixel—both horizontally and vertically. The second pass fills in every eighth pixel (again horizontally and vertically) but with an offset of four pixels to the right. So given an image 8 pixels wide and 8 pixels high, pass one would fill in the first pixel in the first row, and pass two would fill in the fifth pixel on the first row.

The third pass fills in the pixels that are four rows below the pixels filled in by the first two passes. Using the same 8px by 8px image, pass three would fill in the first pixel on row five as well as the fifth pixel on row five.

The fourth pass displays the pixels that are offset by two columns to the right of the first four pixels, and the fifth pass fills in the pixels that fall two rows below each of the prior displayed pixels.

Pass six fills in all remaining pixels on the odd rows, and the seventh and final pass fills in all remaining pixels on the even row.

That’s a lot of numbers, and is quite possibly as clear as mud at this point. For those more visually minded, the following diagram shows which pixels are filled in for each pass.

While the PNG method of interlacing involves more passes, if you were to assume the same network conditions and compression levels, an interlaced PNG image would be on pass four by the time the GIF image had completed its first pass. Why? Because the first pass of GIF interlacing involves 1/8th of the data of the GIF image itself—1 in every 8 rows—whereas the first pass of PNG interlacing involves only 1/64th of the data—1 pixel in every 64 pixels (8 horizontally multiplied by 8 pixels vertically). The impact is particularly noticeable on any images with text as the text becomes readable much more quickly using the PNG approach to interlacing.

Progressive loading, higher-fidelity much earlier than the GIF counterpart—PNG interlacing sounds great right? Unfortunately it’s not all sunshine and roses. The consequence of PNG’s approach to interlacing is that it can dramatically increase the file size because of its negative impact on compression.

Remember all those filters we talked about? Because each pass in the PNG interlacing process has different widths, it’s far simpler to treat each pass as a completely separate
image for filtering. The consequence is that the filtering process has less data to work with, making compression less effective. On top of that, the benefits of progressively loading images has been debated with no definitive conclusion. When you combine the severe reduction in compression with the questionable value of interlacing in the first place, PNG interlacing starts to make a lot less sense. Typical, you’re better off ignoring interlacing on both PNG’s and GIF’s altogether.

**There can be only one!**

So given all the information above, here's the ultimate question: when do you use a GIF and when do you use a PNG? The answer is to favor PNG’s for all except the smallest of images. Likewise, if you want to use animation at all, GIF will be the way to go (though as we've seen above, you could argue MP4’s are even better).

Basically, while the GIF format helped pave the way for formats like PNG, it's time has come and gone. If you are ever considering putting a GIF in a page, take a step back and consider if another alternative would work better.

**Summary**

In this chapter we looked at the two most popular and widely supported lossless image formats on the web, GIF’s and PNG’s. We looked at how each format is encoded and compressed, as well as what tweaks we can make to maximize those savings. Now that you know all about lossless formats, not only can you impress your friends with your in-depth knowledge of filtering and compression algorithms, but you can also start to save precious bytes with every image you produce.

In the next chapter, we’ll dig into JPEG’s—the web’s favorite lossy image format—and learn how to optimize them as much as possible.
JPEGs are the web’s most abundant image file format. According to the HTTP archive, at the time of this writing, they make up 45% of all image requests, and about 65% of image traffic. They are good candidates for full color images and digital photos, making them the go-to image format whenever people want to share important moment in their lives (e.g. what they are having for brunch) over the Internet. JPEG’s capability of lossily compressing images to save bandwidth (without losing too much quality in the process) has gained the format worldwide adoption.

**History**

The need for photographic image compression was clear from the early days of personal computing. Multiple proprietary formats were devised in order to achieve that, but eventually, the need to share these images between users made the case for a standard format clear.

Even before the Internet was widespread, corporations shared images with their users over CD-ROMs with limited storage capacity, and wanted the users to be able to view these images without installing proprietary software. In the early days of the Internet (then mostly at 9600 baud speeds) it was apparent that a standard format could not come soon enough.

A few years earlier, back in 1986, the Joint Photographic Experts Group was formed, and after 6 years of long debates, published the ITU T.81 standard in 1992. The group’s acronym was adopted as the popular name of this new format: JPEG.
The JPEG Format

The bytestream of files that we call JPEG nowadays (often with extensions such as .jpg and .jpeg) is not a direct result of a single standard. They are composed of a container and payload. The payload corresponds to the original T.81 standard (or, to be more accurate, to a subset of that standard that is supported by browsers), while the container is defined by other standards entirely, and is used to, well, “contain” the payload and important metadata about the image that the decoder needs in order to decode it.

Containers

The T.81 standard actually defined a standard JPEG container called JIF, for JPEG Interchange Format. But JIF failed to gain traction, mostly because it was overly strict and failed to provide some information that was required for the decoding process. Luckily JIF was built with forward compatibility in mind, so it was soon succeeded by other, backwards compatible, container formats.

There are two commonly used types of JPEG containers today: JFIF and EXIF.

JFIF stands for JPEG File Interchange Format, and is the older and more basic of the two containers. EXIF stands for Exchangeable Image File Format, and can contain far more metadata than JFIF, such as the location the image was taken, the camera’s settings, copyright, and other metadata that might be relevant for humans editing and manipulating the image, but is not required to display the image in a browser.

Later on we will see how lossless optimization often trims that data in order to reduce it’s size. What is common to all these container types is their internal structure, which is somewhat similar.

They are all composed of…

Markers

Each JPEG file, regardless of container, is composed of markers. These markers all start with the binary character 0xff, where the following character determines the marker’s type. The JFIF and EXIF parts are contained in “application markers” that contain segments that are used to contain these container-specific information. Decoders that weren’t created to interpret or use JFIF or EXIF specific markers, just ignore them and move on to the next one.

A few markers that are fairly important in the JPEG world:

- **SOI** - The “Start Of Image” marker represents the start of the JPEG image. It is **always** the first marker in the file.
• SOF - “Start Of Frame” represents the start of the frame. With one non-practical exception, a JPEG file will contain a single frame.

• DHT - “Define Huffman Table” contains the Huffman tables. We'll discuss them in detail in the “Entropy Encoding” section.

• DQT - “Define Quantization Table” contains the quantizations tables which we'd discuss in the “DCT” section.

• SOS - “Start Of Scan” contains the actual image data. We'll discuss its content below.

• EOI - “End Of Image” represents the end of the JPEG image, and should always be the last marker of the file.

• APP - Application markers that enable extensions to the basic JIF format, such as JFIF and EXIF.

The terms “image”, “frame”, “scan” and “component” can be confusing so let's clarify them. Each JPEG is a single “image”, which contains (in all practical cases) a single “frame”, and a frame can contain one or many “scans”, depending on the encoding mode, which we'll discuss below. On top of that, each scan can contain multiple components. Quite the Russian doll.

One thing that is often surprising is that the JPEG’s pixel dimensions can be buried rather deep inside the bytestream, as part of the Start Of Frame (SOF) marker's header. That means that for JPEGs with a lot of data before that marker (notably EXIF-based JPEGs with a lot of metadata) the information regarding the JPEG's dimensions may come in pretty late. That can be a problem if you're processing the JPEG on-the-fly, and particularly, large chunks of EXIF data can often mean that the browser knows the image dimensions significantly later than it could have if the (irrelevant) EXIF data wasn't there.
Since browsers use the presence of image dimensions for layout changes in certain cases, as well as for triggering various internal processing events, the presence of EXIF metadata in your images can have a significant negative impact on your site’s performance.

**Color transformations**

Another key concept about JPEGs is that they convert the input image’s from its origin RGB color model to the YCbCr color model, breaking the image into brightness, blue chroma and red chroma components.

As we discussed in the Digital Image theory chapter, the human eye is more sensitive to luminance details than it is to details in color components. That means that we can generally get away with relatively high color component detail loss, while the same is not always true for the luminance component.

JPEG takes advantage of that and applies different (and often harsher) compression on the color components of the image.

As we’ve seen, one of the disadvantage of YCbCr vs other, more modern color models (e.g. YCgCo) is that YCbCr is not binary fraction friendly. Those mathematical operations, when carried out by a computer, are bound to lose some precision, and therefore an RGB to YCbCr to RGB conversion is somewhat lossy in practice. That adds to the lossy aspect of the format.
Subsampling

One of the major ways that compression of the color components is performed is called subsampling. Sampling, which we’ve learned about in the Digital Images Theory chapter, is about fitting an analog signal (e.g. a real-life image of continuous color) into an inherently discrete medium, such as a pixel bitmap, a process which by definition losses detail and precision.

Subsampling is about losing even more precision during the sampling (or re-sampling) process, resulting in less detail, entropy, and eventually bytes to send to the decoder.

When we discuss subsampling in JPEG, we are most often talking about chroma subsampling: subsampling of the color components. Doing this reduces the color components sampling precision, which is OK since as we said, the human eye tends to be more forgiving for lost color precision details.

How is subsampling done in JPEG? There are multiple patterns for possible subsampling in the JPEG standard. In order to understand what these subsampling patterns mean, let’s start by drawing a 4x2 pixels row of the Cb (blue chroma) component.

![Figure 4-2. A 4x2 pixel block](image)

Now as you can see in the 4x2 pixels above, each has a different value. Subsampling means that we coalesce the colors of some of them into a single intensity value.

The notation given to the various subsampling patterns is J:a:b, where:

- J is the number of pixels in each row. For JPEG that number is often 4. There are always 2 rows.
- a represents the number of colors used from the first row.
- b represents the number of colors used in the second row.

But just in case you’re dozing off, let’s look at a few examples. Here are a few subsampling patterns with that notation.
If you were paying attention, you may have noticed that the 4:4:4 example is exactly the same as the original. In fact, 4:4:4 means that for each row of 4 pixels, 4 colors are picked, so no subsampling is taking place.

Let’s take a look at what other subsampling patterns are doing.

4:4:0 means that color intensity is averaged between every two vertical pixels in the 4x2 block. In 4:2:2 intensity is averaged between two horizontally neighbouring pixels. 4:2:0 averages intensity between the pixels in each 2x2 block inside the 4x2 block. And finally, 4:1:1 means that intensity is averaged between four vertically neighbouring pixels.

The above example is tainted to make it clear that we’re talking about chroma subsampling, but you should note that each pixel in the example only represents the intensity of one of the color components. That makes it significantly easier to average the pixel color intensity without losing too much color precision.

Also, as you can notice from the examples above, not all subsampling method are created equal, and some are more likely to be noticeable than others. In practice, most JPEGs “in the wild” exhibit either 4:4:4 subsampling (so no subsampling at all) or 4:2:0.

We have seen that we lose precision by subsampling, but what do we gain from it?

By getting rid of pixels in the chroma components we effectively reduce the size of the color component bitmap by half for 4:2:2 and 4:4:0 subsampling and by three quarters (!) for 4:2:0 and 4:1:1. That drop in pixel count equates to significant bytesize savings as well as significant memory savings when dealing with the decoded image in memory. We’ll further discuss these advantages in the “Image processing” chapter.
Entropy coding

As we discussed in the Digital Images Theory chapter, entropy coding is a technique that replaces datastream symbols with codes, such that common symbols get shorter codes.

The JPEG standard includes two different options for entropy encoders: Huffman encoding and arithmetic encoding.

Huffman encoding has been around since 1952, and is based on the idea that once the frequency of the symbols in the data stream is known, the symbols are sorted by their frequency using a binary tree. Then each symbol gets assigned with a code that represents it, and which cannot be confused with the other codes as part of the decoding process. That is, no two or more codes, when concatenated, comprise another, longer code. That fact avoids the need to add length signals for each code, and makes the decoding process straightforward.

Huffman encoding is widely used and has lots of advantages, but suffers from one downside: the codes assigned to each symbol are always comprised of an integer
number of bits. That means that they cannot reflect with complete accuracy the symbol frequency, and therefore, leave some compression performance on the table.

**Huffman encoding in detail**

Let’s sink our teeth into a specific case in order to better understand what that means. Let’s say we have an alphabet comprised of the letter A, B and C, and their probability to appear in the stream is the same for all symbols: 1/3.

With Huffman encoding, we would use a tree structure to arrange them so that the symbols with highest probability are closest to the tree’s root, and then assign symbols accordingly. Since all symbols have the same probability, we’ll end up with the following tree:

*Figure 4-5. A Huffman tree used to code said alphabet.*

As we can see from the tree above, A would be assigned the symbol 0, B would be assigned the symbol 10 and C would be assigned the symbol 11. That means we’re “spending” more bits than needed on B and C, while “spending” less than required on A. B and C are subsidizing A, if you will. Huffman encoding is still a huge win, but if we compare the number of bits it takes us to encode a symbol, we’re not reaching this
theoretical ideal due to this difference between each symbol’s probability and the number of bits we encode it with.

Arithmetic encoding to the rescue!

Arithmetic encoding is able to encode a symbol using fractions of a bit, solving that problem and achieving the theoretical encoding ideal. How does arithmetic coding do that “fractions of a bit” magic? It uses an (extremely long) binary fraction as the code representing the entire message, where the combination of the fraction’s digits and the symbols probability enable decoding the message back.

Arithmetic encoding in detail

To illustrate the way that works, the encoding process starts with the current interval being set to the range between 0 and 1, and with the output binary fraction set to 0.

Each symbol is then assigned a range on the current interval that corresponds with the probability that it would appear in the data stream. For the current symbol in the data stream, its lower limit is added to the output, and the current interval is set to the range of the current symbol. The process then repeats itself until all symbols are encoded.

Figure 4-6. The process of encoding the message “CAB” in an alphabet comprised of A, B and C using arithmetic encoding.
Unfortunately, when it comes to JPEGs, Huffman encoding is the one that is most often used, for the simple fact that arithmetic encoding is not supported by most JPEG decoders, and specifically not supported in any browser. The reason for that lack of support is that decoding of arithmetic encoding is more expensive than Huffman (and was considered prohibitively expensive in the early days of JPEGs), and that it was encumbered by patents at the time that JPEG was standardized. Those patents are long expired, and computers are way better at floating point arithmetic than they used to be in 1992, yet support in decoders is still rare, and it would also be practically impossible to introduce arithmetic encoding support to browsers without calling these JPEGs a brand new file format (with its own MIME type).

But even if arithmetic encoding is rarely used in JPEGs, it is widely used in other formats, as we’ll see in the Browser Specific Formats chapter.

While entropy codings can be adaptive, meaning that they don’t need to know the probabilities of each symbol in advance and can calculate them as they pass the input data, Huffman in JPEG is not the adaptive variant. That means that often the choice is between an optimized, customized Huffman table for the JPEG, that has to be calculated in two passes over the data, and a standard Huffman table, which only requires a single pass, but often produces compression results which are not as good as its custom, optimized counterpart.

Huffman tables are defined in the DHT marker, and each component of each scan can use a different Huffman table, which can potentially lead to better entropy encoding savings.

**DCT**

In the Digital Images Theory chapter we touched upon conversion of images from the spatial domain to the frequency domain. The purpose of such a conversion is to facilitate filtering out high frequency brightness changes that are less visible to the human eye.

In JPEG, the conversion from the spatial domain to frequency domain and back is done by mathematical functions called Forward Discrete-Cosine Transform (FDCT) and Inverse Discrete-Cosine Transform (IDCT). We often refer to both as DCT.

**How does DCT work?**

DCT takes as its input a mathematical function and figures out a way to represent it as a sum of known cosine functions. For JPEGs, DCT takes as input the brightness function of one of the image components.
Figure 4-7. The Y component of an image, plotted as a 2D function

**How does DCT do its magic?**

DCT defines a set of basis functions: special cosine functions which are orthogonal to each other.

That means that:

- There's no way to represent any of the waveforms that these functions create as a sum of the other functions.
- There's only one way to represent an arbitrary 1D function (like sound waves or electrical currents) as the sum of the basis functions, multiplied by scalar coefficients.

This allows us to replace any $n$ value vector by the list of $n$ coefficients that can be applied to the basis functions to recreate the function's values.

The DCT basis functions are ordered from the lowest frequency one to the left and up to the highest frequency on to the right.
Each one of the DCT values is a scalar multiplier of one of the basis functions. The first value, which correlates to the constant function we’ve seen earlier, is called the DC component, since when discussing electrical currents, that constant function represents the Direct Current part. All other values are called the AC components, since they represent the Alternating Current component.

The reason we’re using electrical terms such as DC and AC is that one dimensional DCT is often used to represent electrical signals, such as analog audio signal.

Since we’re talking about images, 1D DCT is not very interesting in and of itself, but we can extend the same concept to two dimensional functions (such as the brightness function of an image). As our basis functions we can take the 1D DCT 8 basis functions we’ve seen earlier and multiply them with each other to get 8x8 basis functions. These functions can then be used in a very similar way to represent any arbitrary set of 8x8 values as a matrix of 8x8 coefficients of those basis functions.

One small difference image data with regard to audio waves or electrical currents is that our function’s possible output range is from 0 to 255, rather than having both positive and negative values. We can compensate for that difference by subtracting 128 from our function’s values.

Figure 4-8. The basis functions of 1 dimensional DCT
Figure 4-9. The multiplication of 1D DCT basis functions creates the following 8x8 matrix of 2D basis functions.

As you can see in the upper left corner, the first basis function is of constant value. That's the two dimensional equivalent of the DC component we've seen in 1D DCT. The other basis functions, due to the fact they result from multiplying our 1D basis functions, are of higher frequency the further they are from that top left corner. That is visualized above by the fact that their brightness values change more frequently.

Let's take a look at the brightness values of the following 8x8 pixel block:
Figure 4-10. A random 8x8 pixel block

Since we want to convert it to DCT coefficients, the first step would be to center these values around 0, by subtracting 128 from them. The result is

Applying DCT on the above matrix results in

50 | Chapter 4: JPEG
Now every cell in the above matrix is the scalar coefficient of the basis function of the corresponding cell. That means that the coefficient in the upper left corner is the scalar of the constant basis function, and therefore it is our DC component. We can regard the DC component as the overall brightness/color of all the pixels in the 8x8 block. In fact one of the fastest ways to produce a JPEG thumbnail that’s 1/8 of the original JPEG is to gather all DC components of that JPEG.

We’ve seen that the coefficient order corresponds with the basis function order, and also that the basis function frequency gets higher the further we are in the right and downwards directions of the matrix. That means that if we look at the frequency of the coefficients in that matrix, we would see that it increases as we get further away from the top left corner.

Now, when serializing the coefficient matrix it’s a good idea to write the coefficients from the lowest frequency to the highest. We’ll further talk about the reasons in the Quantization section, but suffice to say that it would be helpful for compression. We achieve that kind of serialization by following a zig-zag pattern, which makes sure that coefficients are added from lowest frequency to the highest one.

![Zig-zag pattern](image)

*Figure 4-11. The zig zag pattern used by JPEG to properly order lower frequency components ahead of higher frequency ones.*

**Minimal Compression Units**

So, we can apply DCT to any 8x8 block of pixel values. How does that apply to JPEG images that can be of arbitrary dimensions?
As part of the DCT process each image is broken up into 8x8 pixel blocks called MCUs, which stands for Minimal Compression Units. Each MCU undergoes DCT in an independent manner.

What happens when an image width or height doesn't perfectly divide by eight? In such cases (which are quite common) the encoder adds a few extra pixels for padding. These pixels are not really visible when the image is decoded, but are present as part of the image data to make sure that DCT has an 8x8 block.

One of the visible effects of the independent 8x8 block compression is the “blocking” effect that JPEG images get when being compressed using harsh settings. Since each MCU gets its own “overall color” the visual switch between MCUs can be jarring and mark the MCU barriers.

![Figure 4-12. Same image with rough compression settings. Note the visible MCU blockiness.](image)

**Quantization**

Up until now, we’ve performed DCT, but we didn’t save much info. We replaced representing sixty-four 1 byte integer values with sixty-four 1 byte coefficients. Nothing to write home about when it comes to data savings.

So, where do the savings come from? They come from a stage called **quantization**. This stage takes the above coefficients and divides them by a **quantization matrix** in order to reduce their value. That is the lossy part of the JPEG compression, the part where we discard some image data in order to reduce the overall size.
Let's take a look at the quantization matrix of the above image:

\[
\begin{array}{cccccccc}
3 & 2 & 8 & 8 & 8 & 8 & 8 & 8 \\
2 & 10 & 8 & 8 & 8 & 8 & 8 & 8 \\
10 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\
8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\
8 & 8 & 8 & 8 & 10 & 9 & 8 & 8 \\
8 & 8 & 8 & 8 & 8 & 9 & 12 & 7 \\
8 & 8 & 8 & 8 & 9 & 12 & 12 & 8 \\
8 & 8 & 8 & 8 & 10 & 8 & 9 & 8 \\
\end{array}
\]

But that image is the original that was produced by the digital camera, and is quite large from a bytesize perspective (roughly 380KB). What would happen if we'd compress that JPEG with quality settings of 70 to be 256KB, or roughly 32% smaller?
And its quantization matrix?

\[
\begin{array}{cccccccc}
10 & 7 & 6 & 10 & 14 & 24 & 31 & 37 \\
7 & 7 & 8 & 11 & 16 & 35 & 36 & 33 \\
8 & 8 & 10 & 14 & 24 & 34 & 41 & 34 \\
8 & 10 & 13 & 17 & 31 & 52 & 48 & 37 \\
11 & 13 & 22 & 34 & 41 & 65 & 62 & 46 \\
14 & 21 & 33 & 38 & 49 & 62 & 68 & 55 \\
29 & 38 & 47 & 52 & 62 & 73 & 72 & 61 \\
43 & 55 & 57 & 59 & 67 & 60 & 62 & 59 \\
\end{array}
\]

As you can see from the above quantization matrices, they have slightly larger values at the bottom right corner than at the upper left one. As we’ve seen, the bottom right coefficients represent the higher frequency coefficients. Dividing those by larger values means that more high frequency coefficients will finish the quantization phase as a zero value coefficient. Also, in the q=70 version, since the dividers are almost eight times larger, a large chunk of the higher frequency coefficients will end up discarded.

But, if we look the two images, the difference between them is not obvious. That’s part of the magic of quantization. It gets rid of data that we’re not likely to notice anyway. Well, up to a point at least.

**Compression Levels**

Earlier we saw the same image, but compressed to a pulp. Wonder what the quantization matrix on that image looks like?

\[
\begin{array}{cccccccc}
160 & 110 & 100 & 160 & 240 & 255 & 255 & 255 \\
120 & 120 & 140 & 190 & 255 & 255 & 255 & 255 \\
140 & 130 & 160 & 240 & 255 & 255 & 255 & 255 \\
140 & 170 & 220 & 255 & 255 & 255 & 255 & 255 \\
\end{array}
\]
We can see that almost all frequencies beyond the first 20 are guaranteed to be quantified to zero (as their corresponding quantization value is 255). And it’s even harsher in the quantization matrix used for the chroma components:

| 170 180 240 255 255 255 255 255 |
| 180 210 255 255 255 255 255 255 |
| 240 255 255 255 255 255 255 255 |
| 255 255 255 255 255 255 255 255 |
| 255 255 255 255 255 255 255 255 |
| 255 255 255 255 255 255 255 255 |
| 255 255 255 255 255 255 255 255 |
| 255 255 255 255 255 255 255 255 |

It is not surprising then that the image showed such blockiness. But what we got in return to that quality loss is an image that is 27KB or 93% (!!!) smaller than the original. And you could well argue that the result is still recognizable.

Note that the compression level and quality settings of the different JPEG encoders mean that they pick different quantization matrices to compress the images. Also worth noting that there’s no standard for what quantization matrices should be picked and what quality levels actually mean in practice. So a certain quality setting in one encoder can mean something completely different (and of higher/lower visible quality) when using a different encoder.

One more thing of note is that encoders can (and often do) define a different quantization matrix for different components, so it can apply harsher quantization on the chroma components (which are less noticeable) than it would apply on the luma component.

**Dropping Zeroes**

How does zeroing out the coefficients help us better compress the image data? Since we are using a zigzag pattern in order to sort the coefficients from lower frequency to high frequency, having multiple zero values at the end of our coefficient list is very easy to discard, resulting in great compression. JPEG further takes advantage of the fact that in many cases zero values tend to gather together, and adds limited form of Run-Length-Encoding, which discards zeros and simply writes down the number of preceding zeros before non-zero values. The remaining values after quantization are also smaller numbers which are more amenable to entropy encoding, since there’s higher probability that these values are seen multiple times than a random 0-255 brightness value.

**Dequantization**
At the decoder, the reverse process happens. The quantified coefficients are multiplied by the values of the quantization matrix (which is sent to the decoder as part of the DQT marker) in a process called **dequantization**, which reconstructs an array of coefficients. The accuracy of these coefficients vs. the coefficients encoded varies based on the values of the quantization matrix. As we’ve seen, the larger these values are, the harsher the compression and therefore the further are the coefficients that the decoder sees from the original ones.

**Lossy by nature**

It is important to note that the quantization process as well as the YCbCr color transformations are lossy processes. That means that if we’d take a JPEG and compress it to the same quality (so, using the same quantization tables) over and over, we will see a significant quality loss after a while. Each time we encode a JPEG, we lose some quality comparing to the original images. That’s something worth bearing in mind when constructing your image compression workflow.

**Progressive JPEGs**

Sequential JPEGs are JPEG in which each one of the MCUs is sent in its entirety in a single scan. Such JPEGs can be decoded as they come, creating a partial image.

Progressive JPEGs on the other hand are JPEGs which MCU data is sent over in multiple scans, enabling the decoder to start decoding an approximative image of the entire JPEG after receiving just one of the scans. Future scans further refine the image details. That enables (in supporting browsers) to optimize for a first impression of the user, without compromising on the eventual image quality.
We can see that the image is not perfect, but it is fairly complete.

There are two forms of sending JPEG data progressively: spectral-selection and successive-approximation. Spectral-selection means that the parts of the MCU data that are sent first are the low frequency coefficients in their entirety, where higher frequency coefficients are sent as part of a later scan. Successive-approximation means that for each coefficient, its first few bits are sent as part of an early scan, while the rest of its bits are sent at a later scan.

These two methods are not mutually exclusive, and can be combined for ideal progressive results. Some coefficients are sent in their entirety, while other are sent over multiple scans.

One significant advantage of progressive JPEGs is that each scan can have its own dedicated Huffman table, which means that progressive JPEGs can have higher compression ratio, as each part of the JPEG can have a highly optimal Huffman table.

It is worth noting that, as the popular saying goes, there's more than one way to scan a JPEG. There's a very large number of combinations for possible scans, differing from one another in the coefficients that get sent in their entirety, the coefficients that get sent progressively using successive approximation, as well as which components get sent first.

This allows us to squeeze some extra compression from JPEGs. Finding the ideal combination of progressive scans and their relative Huffman compression performance is a non-trivial problem. Fortunately, the search space is not huge, so smart encoders just brute-force their way to find it. That is the secret of the lossless optimizations performed by tools like jpegrescan, which are now integrated as part of MozJPEG (which we'll soon discuss).
Unsupported modes

The JPEG standard includes two more modes, but those are rarely supported by encoders and decoders, meaning they are rarely of practical use.

Hierarchical mode

Hierarchical operation mode is similar to progressive encoding, but with a significant difference. Instead of progressively increasing the quality of each MCU with each scan being decoded, the hierarchical mode enables progressively increasing the spatial resolution of the image with each scan.

That means that we can provide a low-resolution image and then add data to it to create a high-resolution image! Here’s how it works — the first scan is a low-resolution baseline image, while each following scan upsamples the previous scan to create a prediction basis upon which it builds. This way, each scan other than the first only sends only the difference required to complete the image to be of full resolution.

Unfortunately, it is not very efficient compared to other JPEG modes. It is also limited in its utility, since upsampling can only be done by a factor of two.

Lossless mode

The lossless operation mode in JPEG is another rarely supported operation mode. It is quite different from the other operation modes in the fact that it doesn’t use DCT to perform its compression, but instead uses neighbouring pixels based prediction (called Differential Pulse Code Modulation or DPCM) in order to anticipate the value of each pixel, and encode only the difference between prediction and reality. Since the difference tends to be a smaller number, it is then more susceptible to entropy coding, resulting in smaller images compared to the original bitmap (but still significantly larger than lossy, DCT based JPEGs).

JPEG Optimizations

As we’ve seen in the Digital Images chapter, lossy image formats such as JPEG (ignoring its irrelevant lossless mode of operation) can undergo both lossy and lossless types of compression. In this section we’ll explore various optimization techniques that are often used to reduce the size of JPEG images.

Lossy

As far as lossy optimization, JPEG images can be optimized by undergoing the regular DCT based high-frequency reduction, only with more aggressive quantization tables. Quantization tables with higher numeric values lead to higher loss of high-
frequency brightness changes, resulting in smaller files but with more visible quality loss.

Therefore a common way to optimize JPEGs is to decompress them and then recompress them with lower “quality” values (which translate into higher numeric values quantization tables).

**Lossless**

There are multiple ways to losslessly optimize a JPEG:

- Optimize its Huffman tables for current scans.
- Rescanning it, in order to achieve the ideal combination of progressive JPEG scans and Huffman tables.
- Remove non-photographic data such as EXIF metadata.

We already discussed the first two when we talked about Huffman tables and progressive JPEGs, so we’ll expand on the third here.

EXIF metadata is added to JPEGs by most if not all modern digital cameras and by some photo editing software. It contains information regarding when and where the image was taken, what were the camera settings, copyright info and more. It may also contain a thumbnail of the image, so that a preview image can be easily displayed.

However, when delivering images on the web, all that info (perhaps besides copyright information) is not really relevant. The browser doesn’t need that information and can display the image just fine without it. Furthermore, the user cannot access that information unless they explicitly download the image to look for it (and disregarding very specific and niche use-cases, they would not care about it).

Also, as we saw earlier, that metadata may appear in the JPEG before the information regarding the JPEG dimensions, which can lead to delays in the time the browser knows what the image dimension are, and can result in a “bouncy” (or “bouncier”) layout process.

So, it makes good sense to remove this metadata from web served images. There are many software utilities that enable you to do that, and we’ll further discuss them in the Operationalizing Image Compression chapter.

You should note that EXIF data may also contain orientation information which in some cases can alter the image orientation when displayed in the browser. At least today, most browsers (with the notable exception of mobile Safari) ignore orientation information for images that are embedded in the document (either content images or background images), but they are respecting it when the user navigates directly to the image. Firefox also respects orientation information when an (experimental) CSS property called `image-orientation` indicates that it should.
Therefore, dropping orientation info can cause user confusion or content breakage in various scenarios. It is advisable to maintain it intact when processing JPEGs.

**MozJPEG**

We already mentioned that JPEG has been around for a long while, and JPEG encoders existed for just as long. As a result, many of them have not been updated with new features and improvements in recent years. At the same time, various browser-specific image formats (which we’ll discuss in the next chapter) were sparking interest in image compression and since their encoders were being written from scratch, they included more recent algorithms, which presented a non-negligable part of the reason these formats performed better than JPEG.

Mozilla, reluctant to introduce support for these newer formats, decided to start improving JPEG’s encoding and bring it up to the current state-of-the-art, so that we can at least compare the different formats on a level playing field.

Hence they started the MozJPEG project, with the goal of increasing JPEG’s compression performance and create smaller, similar quality files compared to other encoders, without hurting JPEG’s compatibility with all existing browsers. In order to reduce unnecessary development, and increase compatibility with existing image compression workflow, the project is a fork of the libjpeg-turbo project and a drop-in replacement of it in terms of binary interface.

The project uses various encoding optimizations to achieve improved compression rates:

- Lossless compression based on ideal progressive scan patterns which produce smaller files.
- Trellis quantization - An algorithm that enables the encoder to pick better adapted quantization tables, in order to minimize image distortion for the current image.
- Quality tuning based on visual metrics, such as SSIM.
- Deringing of black text over white background.
- And more.

**Summary**

In this chapter we looked into how JPEGs are constructed, which methods they use in order to achieve their impressive compression ratios, and how can JPEGs be optimized further.

Practical takeaways of this chapter include:
• Progressive JPEGs can show the full image in lower quality sooner, providing a better user experience than sequential JPEGs.

• Progressive JPEG can have smaller byte size than sequential ones.

• JPEG encoders’ quality metric is often only an indication of the quantization table used and its impact on various images may vary greatly.

• Lossless optimization such as EXIF removal can have significant implications on byte size as well as the browser’s ability to calculate the image’s layout as early as possible.

• Chroma subsampling can significantly reduce the size of JPEG’s color components.

• JPEG’s compression is a lossy process, and each consecutive reencoding results in some quality loss.

• If you have an image compression workflow that’s producing JPEGs, MozJPEG should probably be a part of it.

In the next chapter we will see how other, newer image formats are taking similar methods further (by incorporating algorithmic knowledge that the compression industry have accumulated since 1992), to accomplish even better compression ratios.
While the traditional image formats used on the web, GIF, JPEG, and PNG, have served us well and will continue to be useful well into the future, there are a number of new formats that have been developed that can be particularly useful on the web today. The most notable and useful of these formats are Google’s WebP, Microsoft’s JPEG XR, and JPEG 2000. All three of these formats improve on the features of GIF, JPEG, and PNG while often also improving compression and fidelity.

The biggest improvement these formats all provide to the web is that they all support lossy compression with full transparency. Traditionally, to have an image on the web with full transparency, the only option was to use PNG. While this enabled full transparency it came at the cost of dramatically heavier images because PNG’s compression is lossless. Now, with these new formats, it’s possible to get the best of both worlds: full transparency at a fraction of the byte size.

The second improvement WebP, JPEG XR, and JPEG 2000 provide is smarter and fancier image compression. We’ve learned a lot about image compression since JPEG was first introduced in 1992 and these three formats have capitalized on that. While each of these formats uses a different approach to compression, they often outperform JPEG at comparable fidelity levels for byte savings.

There’s one drawback to these formats though, at least on today’s web. Not all browsers support these formats. Actually, for the most part, any of today’s major browsers will support only one, if any, of these formats. This means that, if you want to use any of these formats and get their benefits, you’ll need to be smart about how the images get delivered. If you serve the wrong format to the wrong browser you’ll end up with a broken image at the added expense of transferring all of those image bytes to the end user for nothing. Bummer!
When these formats are used properly there are substantial byte savings to be had. Let’s discuss these three new formats in more detail.

**WebP**

WebP, developed and promoted by Google, was the first browser-specific image format to gain any substantial adoption and mindshare from web developers. It’s based on Google’s VP8 video codec; specifically it wraps VP8’s intra-frame image coding in a RIFF image container.

Today, there are effectively three different variations of WebP: Basic, Extended, and Animated. The “basic” variation is very simple. It supports encoding a single lossy opaque image, much like JPEG. The “extended” variation added support for lossless compression and, more importantly, full transparency. Finally, “animated” WebP images are built on top of the “extended” variation and add animation support; this makes animated WebP images a good replacement for animated GIFs if the browser has support.

These three variations show that WebP is happy to evolve to improve and add features but it also shows a tricky compatibility landscape. Different versions of different browsers have varying support for the different variations of WebP.

**WebP Browser Support**

Browser support for WebP extends primarily to Google / Blink based browsers: Chrome, Android Browser, and Opera. The support matrix looks like this today:

Table 5-1. WebP browser version support

<table>
<thead>
<tr>
<th></th>
<th>Basic</th>
<th>Extended</th>
<th>Animated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome (desktop)</td>
<td>&gt;= 17</td>
<td>&gt;= 23</td>
<td>&gt;= 32</td>
</tr>
<tr>
<td>Chrome (Android)</td>
<td>&gt;= 25</td>
<td>&gt;= 25</td>
<td>&gt;= 32</td>
</tr>
<tr>
<td>Chrome (iOS)</td>
<td>&gt;= 29 and &lt; 48</td>
<td>&gt;= 29 and &lt; 48</td>
<td>No</td>
</tr>
<tr>
<td>Android</td>
<td>&gt;= 4.0</td>
<td>&gt;= 4.2</td>
<td>No</td>
</tr>
<tr>
<td>Opera (desktop)</td>
<td>&gt;= 11.10</td>
<td>&gt;= 12.10</td>
<td>&gt;= 19</td>
</tr>
<tr>
<td>Safari</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Firefox</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Chrome for iOS brought dropped WebP support in the transition from UIWebView to iOS 8's WKWebView. Using WKWebView brought performance and stability. Unfortunately, WKWebView uses the native Safari rendering engine and does not allow much in the way of customization. The result is that WebP support in iOS was dropped in version 48. All versions of Chrome for iOS, however do support JPEG 2000.

Google suggests using the Accept HTTP request header and checking if it contains `image/webp` to determine if a server should optionally serve a WebP image to a client. While this works for many cases, it has problems relating to the evolving nature of WebP. If a client sends an "Accept: `image/webp`" header, you can assume it supports up to the "extended" variation of WebP but it is impossible to know (with the Accept header alone) if the client supports WebP animation. If new features are added to WebP (like improved VP9 coding) then this problem compounds and it will be impossible to determine support by Accept header alone.

Because of this deficiency with the Accept header and because most other browser-specific formats don't use the Accept header, this author suggests that the most robust solution generally is to, unfortunately, parse User-Agent strings to determine image support in addition to the Accept header. The Accept header is discussed in more detail in Chapter 13.

**WebP Details**

The most interesting variation of WebP when talking about optimizing for the web is the "extended" variation. This variation is important because it supports both lossy encoding and full transparency. With these two features, WebP becomes a great format to replace both JPEG and PNG. You get the byte savings of JPEG (and then some) and the transparency support previously only available in the byte-heavy PNG format. The lossless compression modes are useful in many contexts but web performance isn't particularly one of them. WebP offers good byte savings for it's lossless encoding when compared against other lossless encodings but the image weight is usually impractical for normal web use. The lossless encoding features of WebP are more interesting and relevant for image archiving purposes.

At it's core, lossy WebP is encoded very similarly to how JPEG is encoded with some major important differences. Like in JPEG encoding, the Discreet Cosine Transform
(DCT) is also used in WebP encoding. Instead of JPEG’s 8x8 pixel blocks, WebP uses 4x4 pixel blocks for performing the DCT. WebP also allows for a variety of zig-zag patterns to traverse the pixels in a block compared to JPEG’s single zig-zag pattern. The biggest improvement over JPEG is that WebP will try to predict the pixels in a block using pixels above and to the left of the block and a number of predetermined prediction algorithms. Having made a prediction of a particular block, this block can now be precisely described as a difference from this prediction. While JPEG applies the DCT to the raw pixels, WebP applies the DCT to this prediction difference. WebP’s approach means that the coefficients produced by the DCT are generally much smaller and contain more zeros than JPEG’s approach. This is one of the primary compression improvements of WebP over JPEG.

The second major difference WebP has against JPEG is the compression algorithm used to encode all of these DCT coefficients. JPEG uses Huffman encoding whereas WebP uses the superior Arithmetic encoding. The JPEG specification allows for JPEGs to be encoded using Arithmetic encoding but this was never implemented by anything other than very specialized encoders and decoders. The reason Arithmetic encoding never caught on with JPEG is because, at the time, there were a number of patents protecting the algorithm and licensing the technology would have been costly. Because of this, virtually all JPEGs are encoded using Huffman encoding and changing that would involve an almost impossible shift in JPEG compatibility and legacy JPEG code. By the time WebP hit the scene, patents surrounding Arithmetic encoding had expired allowing for a fresh start.

WebP isn’t perfect though, there are two important features of JPEG that are missing with (lossy) WebP. The first missing feature is configurable chroma subsampling. VP8 encoding specifies that a chroma subsampling of 4:2:0 will always be used; this means that all WebP images are also encoded using 4:2:0 chroma subsampling. For the vast majority of images this is a great choice and, among other benefits, provides very sizeable byte savings with minimal visual degradation. There are a number of image types, though, that don’t lend well to this aggressive chroma subsampling. Images with hard edges between black or white and solid color often have noticeable artifacts along these edges. With this chroma subsampling, there’s often a dark ring in the colored edge that is unacceptable to many people. This is most commonly seen with solid colored text in images. The inability to configure chroma subsampling in WebP means that either you have to live with this degradation in these types of images or you have to use another image format for these images. Thankfully, there’s been recent work towards improving WebP’s chroma subsampling. The latest version of the cwebp tool offers a "-pre 4" option that uses a new chroma subsampling algorithm that dramatically reduces this degradation at the expense of longer image encoding time.

The second important feature that JPEG has that is missing from WebP is progressive loading. Instead of loading the image top to bottom, JPEG has the ability to load pro-
gressively starting with an entire low quality image which then progressively improves in quality as data is received. This ability to show an early full low quality image is great for the perception of fast loading; it makes people think the image has loaded much sooner than it really does. This feature is entirely absent from WebP. It can be argued that WebP images load faster than a comparable JPEG simply because WebP images are much lighter weight byte-wise. This argument doesn’t necessarily hold up for large images, though, where it is more important to get a quicker sense of completeness at the expense of lower fidelity (which will later be improved) than it is to display the full fidelity final image line by line as the data comes in. The absence of progressive loading also makes some more interesting optimizations impossible. For example, with HTTP/2 it is possible to be clever about how image resources are prioritized and multiplexed. A smart HTTP/2 server might give a higher priority to the beginning of an image that can be progressively loaded and a lower priority to the remaining bytes. This allows the low quality portion of the image to load quickly while also reducing bandwidth contention for other resources. This is, unfortunately, impossible with WebP.

**WebP Tools**

The tooling for working with WebP images is pretty good; better than all of the other tools for working with browser-specific image formats. The two main tools are *libwebp* and *ImageMagick*. *libwebp* is, itself, a C library for encoding and decoding WebP images but has useful standalone tools bundled with it. These tools are *cwebp* and *dwebp* for encoding and decoding WebP images respectively. If you’re familiar with cjpeg for creating JPEG images then *cwebp* will feel very familiar. *ImageMagick* actually uses *libwebp* internally to provide WebP support. If you are already using *ImageMagick* for some of your image processing then using it to take advantage of WebP becomes very convenient.

**JPEG XR**

JPEG XR is Microsoft’s take on a new image format. The XR stands for extended range, which was one of the primary goals of the format. JPEG XR allows for higher bit depths per color channel than JPEG which leads to an extended range of possible colors that can be represented. While this extended range is the feature prominent in the format’s name, it isn’t the feature that’s most interesting from a web performance perspective. Like WebP, the important features of JPEG XR above and beyond JPEG are improved lossy encoding and transparency support making it a good replacement for both JPEG and PNG images.
JPEG XR Browser Support

The only browsers that support JPEG XR today are Microsoft’s browsers, specifically Internet Explorer 10 and higher and the new Edge browser. While Internet Explorer 9 does support JPEG XR partially, there were rendering bugs that made the format unusable for most purposes. Internet Explorer 9 would display an unsightly grey border around all JPEG XR images; this was fixed in Internet Explorer 10. The support matrix looks like this today:

<table>
<thead>
<tr>
<th>Browser</th>
<th>Support Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer</td>
<td>&gt;= 10</td>
</tr>
<tr>
<td>Edge</td>
<td>Yes</td>
</tr>
<tr>
<td>Chrome</td>
<td>No</td>
</tr>
<tr>
<td>Android</td>
<td>No</td>
</tr>
<tr>
<td>Opera</td>
<td>No</td>
</tr>
<tr>
<td>Safari</td>
<td>No</td>
</tr>
<tr>
<td>Firefox</td>
<td>No</td>
</tr>
</tbody>
</table>

Internet Explorer and the Edge browser will send an "Accept: image/jxr" header with HTTP requests for images. This header could be used by a server to decide if a JPEG XR image should be served to a client. Unfortunately Internet Explorer 10 and earlier doesn’t send this header so, in general, it’s more practical to parse User-Agent strings if you want to cover the widest supported user base.

JPEG XR Details

JPEG XR supports all of the important features of JPEG while improving lossy encoding byte savings and adding support for full transparency. Unlike WebP, JPEG XR does support a full range of chroma subsampling options as well as support for progressive loading.

A number of new approaches are taken to compress images using JPEG XR, many of which are designed to enable — but not force — lossless encoding. Firstly, while JPEG uses YCbCr to describe pixel data, JPEG XR uses a similar but slightly different color-space: YCgCo. Just as Cb is blueness and Cr is redness, Cg is greenness and Co is orangeness. YCgCo accomplishes a lot of the same goals as YCbCr but is able to do so in a completely lossless way. Secondly, instead of using the Discreet Cosine Transform like JPEG, JPEG XR uses a modified version called Photo Core Transform (PCT). PCT is similar to DCT except for the process is entirely lossless as well. All lossiness in JPEG XR is entirely due to PCT coefficient quantization. A lossless JPEG XR image is the special case where all quantizations are set to 1 — no quantization. JPEG XR improves on JPEG by allowing a certain amount of overlapping when working with
blocks of pixels. This overlapping helps reduce the blocking effect infamous in low quality JPEG images.

To improve compression, JPEG XR allows for different PCT coefficient ordering patterns instead of JPEG’s single zig-zag pattern. JPEG XR also has a certain amount of block prediction to help reduce the magnitude of the PCT coefficients. Both of these techniques, if even at just a conceptual level, are mirrored in WebP. JPEG XR does not mirror WebP with the final entropy encoding though. JPEG XR, like JPEG, still uses Huffman coding to compress the final PCT coefficient data instead of using the superior Arithmetic encoding.

**JPEG XR Tools**

JPEG XR’s tools are its biggest downfall. They are definitely the most difficult tools to use of all the browser-specific formats. Microsoft provides software called *jxrlib* with bundled tools called *JxrEncApp* and *JxrDecApp* to encode and decode JPEG XR images. The software is very rarely updated and is provided as source code only. Anyone who wants to use these tools will have to go through the process of building the software themselves for their own system.

*ImageMagick* advertises JPEG XR support but it isn’t actually particularly useful. *ImageMagick* only supports lossless encoding so it isn’t useful for web performance. *ImageMagick* actually just delegates all encoding and decoding work to the *JxrEncApp* and *JxrDecApp* tools if it’s able to find them. This delegation works sometimes but seems to work inconsistently. It’s often worth the effort to use the *JxrEncApp* and *JxrDecApp* tools directly even though they are rather difficult to use.

**JPEG 2000**

JPEG 2000 was developed by the Joint Photographic Experts Group as their follow up to JPEG. In addition to a completely new way of encoding images, a number of new features were added to JPEG 2000 that weren’t available in JPEG like lossless encoding, higher channel bit depths, and full transparency support.

**JPEG 2000 Browser Support**

Support for JPEG 2000 is available in all of Apple’s recent browsers. Support has been available in desktop and mobile Safari since version 5. An interesting side effect of this Safari support is that Chrome for iOS also supports JPEG 2000. This is because Chrome for iOS is built on top of Safari instead of Blink and means it’s the only browser that supports more than one browser specific format: JPEG 2000 and WebP. The support matrix looks like this today:
Table 5-3. JPEG 2000 browser version support

<table>
<thead>
<tr>
<th>Browser</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safari &gt;= 5</td>
<td>Yes</td>
</tr>
<tr>
<td>Chrome (iOS)</td>
<td>Yes</td>
</tr>
<tr>
<td>Chrome (non-iOS)</td>
<td>No</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>No</td>
</tr>
<tr>
<td>Edge</td>
<td>No</td>
</tr>
<tr>
<td>Android</td>
<td>No</td>
</tr>
<tr>
<td>Opera</td>
<td>No</td>
</tr>
<tr>
<td>Firefox</td>
<td>No</td>
</tr>
</tbody>
</table>

Safari doesn't send any hints in HTTP headers about what image formats it will accept. Unlike recent versions of Chrome and Edge, Safari doesn't send any Accept header with image requests. This means that the most practical way for a server to determine whether or not it should send a JPEG 2000 image is by parsing the User-Agent string.

**JPEG 2000 Details**

JPEG 2000 maintains all of the important features of JPEG including configuration options for chroma subsampling and progressive loading which are absent from WebP. Support for full transparency has been added which, like WebP and JPEG XR, makes JPEG 2000 another great alternative to JPEG and PNG.

While the feature set of JPEG 2000 is similar to the other browser specific formats, under the hood it is the most different format as far as encoding of the actual image is concerned. JPEG 2000 is different because it doesn't use DCT or any variation of DCT. Instead, JPEG 2000 uses a Discreet Wavelet Transform (DWT) at the core of its encoding. Its best to think of DWT as a transform that takes an image and divides it in to four parts. The first part is the original image at one half the width and one half the height. The other three parts are all also individually one half the height and one half the width of the original image but, combined, contain the details necessary to exactly construct the full size image from the first part. One part has horizontal details, one part has vertical details, and the last part has diagonal details.
Figure 5-1. Original image before Discrete Wavelet Transform
You can see in Figure 5-2 that the three detail parts are mostly empty and black. This emptiness allows for a lot of opportunities for compression. To extract even more sparse details, we can repeat this DWT process recursively on the first newly scaled image part. After we've recursively applied DWT a number of times, the detail parts are quantized much like DCT coefficients are quantized in JPEG. After quantization, Arithmetic encoding is used for final compression.

**JPEG 2000 Tools**

The tools for encoding JPEG 2000 are in the middle of the road as far as ease of use and features go. The OpenJPEG project provides a C library and the `opj_compress` and `opj_decompress` tools for encoding and decoding images. These tools don't abstract the concept of “quality” to a simple 1 to 100 scale like most image encoders, instead quality is described using compression ratios or PSNR values. The current release is also missing important features like transparency and chroma subsampling support although transparency support is available if you build the latest unreleased version from the project's source control repository.
ImageMagick has decent JPEG 2000 support and, in fact, uses the OpenJPEG C library behind the scenes. This means that ImageMagick has the same limitations as OpenJPEG when working with JPEG 2000 images but provides a simpler interface if you're already familiar with ImageMagick.

Finally, Kakadu Software makes a popular full featured JPEG 2000 encoder that people and businesses are able to license for a fee. While features like chroma subsampling are available, learning how to use the features is difficult. This encoder is also much faster for encoding.
CHAPTER 6
SVG and Vector Images

outline

• trouble with raster; what are vector images
• advantages / disadvantages
• pro: scalability
• con: pixl perfect caused by scaling
• different vector formats
• AI
• SVG basics:
  • viewBox / group / def / etc
  • css styling
  • basic optimizations (svgo, etc)
• browser support
• broad:
  • narrow support: specific attributes
• tools & resources
4. The Transmission of Digital Audio: Data Formats

Now that you understand the fundamental concepts, terms, and principles behind analog audio and how it is digitized into digital audio, it is time to explore how digital audio is compressed and stored using popular open source digital audio file formats.

You’ll learn about advanced digital audio concepts, such as compression, codecs, bit rates, streaming audio, captive digital audio, and HD audio. Finally, you’ll look at a number of powerful digital audio formats that are supported by open source content development platforms, such as HTML5, Java, JavaFX, and Android Studio. You can use any of the digital audio formats to deliver digital audio content for podcasts, music publishing, web site design, audio broadcasts, or multimedia applications.

Audio Compression and Data Formats

Once you sample your audio, you compress it into a digital audio file format for streaming over the Web or for captive audio file playback within an application. In this chapter, you’ll look at encoding audio using bit rates, and learn about streaming and the new 24-bit HD audio standard, which is now utilized in broadcast and satellite radio. I’ll also cover audio codecs and the audio file formats that they support across open platforms, such as HTML5, Java, and Android. (I cover digital audio data footprint optimization in Chapter 12, after you learn more about Audacity and digital audio editing.)

I want to make sure that you have a deep understanding of these digital audio new media assets so that you can eventually “render” them inside of your target application and attain a professional product that offers an impeccable end-user experience.

Digital Audio Codecs: Bit Rates, Streaming, and HD
Digital audio assets are compressed using something called a codec, which stands for “code decode.” The codec is an algorithm that applies data compression to digital audio samples and determines which playback rate, called a bit rate, it will use, as well as if it will support streaming or playback during network data transfer. First, let’s take a look at how you use digital audio assets in your applications: Do you store audio inside an application or do you stream it from remote servers over the Internet? After that, you’ll consider the audio playback rate or data-streaming bit rate that you’ll want to use. Finally, you’ll learn about HD audio and see if it’s appropriate for your digital audio applications. Only then will you be ready to look at the different audio file formats, which are actually codecs!

**Digital Audio Transmission:**
**Streaming Audio or Captive Audio?**

Just as with digital video, which you view on the Internet every day, digital audio assets can either be captive, or contained within an application (for example, in an Android APK file), or they can be streamed using a remote data server. Similar to digital video, the upside to streaming digital audio data is that it can reduce the data footprint (size) of your application’s files. The downside is reliability.

Streaming audio saves the data footprint, because you don’t have to include all that data-heavy new media digital audio in your app file. Thus, if you are planning on coding a Jukebox application, you want to consider streaming digital audio data, as you would not want to pack your song library into your app’s file because it would be 10 gigabytes (in a large library).

Otherwise, for application audio, such as user interface feedback sounds, game play audio, and so forth, try to optimize your digital audio data so that you can include it inside your app file as a captive asset. In this way, it is available to your application users when needed.

As you know, I’ll go over optimization in Chapter 12, after digital audio editing has been covered. The reason that I want to cover this topic toward the end of the book is that the last step in the asset creation process is exporting your digital audio data using one of the formats discussed in the next section.

The downside to streaming digital audio is that if your user’s connection (or your audio server) goes down, your audio file won’t be present for your end users to play and listen to! The
reliability and availability of a digital audio data stream is a key factor to consider on the other side of the streaming audio vs. captive digital audio decision.

**Streaming Digital Audio Data: Setting Your Bit Rates Optimally**

One of the primary concepts in streaming your digital audio is the bit rate of that digital audio data. Again, this is very similar to digital video, which also uses the concept of bit rates to determine the size of the data pipe that the audio data streams through. The digital audio bit rate is defined during digital audio file compression by the settings that you give to the codec.

Digital audio files that need to support a lower bit rate to accommodate slower bandwidth networks have more compression applied to the digital audio data. This results in a lower audio-quality level. However, lower playback quality isn’t as noticeable in digital audio as it is in digital video.

Low bit-rate digital audio can always play back smoothly across a greater number of hardware devices. This is because if there are fewer bytes of audio data to transfer over any given data network, then there are fewer digital audio data bytes to be processed by the CPU inside that hardware device.

As a processor gets faster, it can process more bytes per second. As a data bandwidth connection gets faster, it can more comfortably send or receive more bytes per second.

Therefore, it is important to remember that you are not only optimizing your audio file size for network transfers, but you are also optimizing your digital audio assets for the amount of system memory that asset requires, as well as the amount of processing cycles that the CPU uses to process the digital audio asset sample data.

**High-Definition HD Digital Audio: 24-Bit 48 kHz Sampling Data**

As I mentioned in Chapter 3, the industry baseline for superior standard definition (SD) audio quality is known as CD quality audio, which is defined as a 16-bit data sample resolution and the 44.1 kHz data sampling frequency. It was used to produce audio CD products way back in the 20th century and it is still used as a minimum digital audio quality standard.
There is also a more recent HD audio standard that uses a 24-bit data sample at a 48 kHz or a 96 kHz sample frequency. It is used today in HD radio and HD satellites, as well as in HD-compatible Android devices, such as the new Droid X HD “high-fidelity” Android smartphones. These provide the user with an extremely high-fidelity digital audio experience. HD audio is supported by several of the open source codecs.

Digital Audio Storage and Playback: File Formats

There are considerably more digital audio codecs supported in the open platforms (HTML5, Java, or Android) than digital imaging codecs, as there are only four image codecs: PNG, JPEG, GIF, and WebP. Android Studio audio support, for instance, includes MP3 (MPEG-3) files, WAV or AIFF (PCM) files, MP4 or M4A (MPEG4) files, OGG files, FLAC files, and MID, MXMF, and XMF MIDI files, which as you know from Chapter 2 are not really digital audio data. Let’s cover all the digital audio formats that support sampled (digitized waveform) data.

MIDI: Musical Instrument Data Interface’s MID, XMF, and MXMF

Since MIDI was covered in Chapter 2, I will just go over the different file formats supported in open platforms, such as HTML5 (browsers and operating systems), Java (using JavaFX), and Android Studio. There are several MIDI file formats, including MID, XMF, and MXMF MIDI formats. They are exceptionally compact because there is zero waveform data; there is only performance data, such as note on, note off, aftertouch, and so on. You opened and scored a MIDI file named fidelio.mid using Rosegarden in Chapter 2!

MPEG-3: The Popular MP3 Digital Audio Player Data Format

The most popular digital audio format in history is the MP3 digital audio file format, which is short for MPEG-3. Most of you are familiar with the MP3 digital audio files found on music download web sites such as Napster. Most of us collected songs in this format to use on popular MP3 players or in CD-ROM music collections. The reason this MP3 digital audio file format is popular is because it has a relatively good compression-to-quality ratio, and because the codec needed to play MP3 audio files is found everywhere, including Android, iOS, Blackberry, Windows, Java, JavaFX, and HTML5.
MP3 is an acceptable format to use in your web site or application as long as you can get the highest quality level possible out of it by using the optimal encoding work process (again, this will be covered in Chapter 12).

Because of software patents, Audacity 2 can’t include MP3 encoding software or distribute any MP3 software from its own web site, which is why I showed you how to download and install the free LAME and FFMPEG encoders for Audacity.

It’s important to note that the MP3 codec outputs a lossy compression audio file format. Lossy compression is where some of the audio data, and thus quality, is discarded during a data compression process; it cannot be recovered. This is similar to the JPEG compression algorithm for digital images, which can cause visual artifacts (purple, green, or yellow pixel smudges).

Open platforms do support the open source lossless audio codec called FLAC, which stands for Free Lossless Audio Codec. Support for FLAC is now as widespread as MP3, due to the free nature of the software decoder.

**FLAC: The 24-Bit HD Audio Capable Free Lossless Audio Codec**

FLAC uses a fast algorithm, so the decoder is highly optimized for speed. FLAC supports 24-bit audio, and there are no patent concerns for using it. It is a great audio codec to use in Android or HTML5 if you need high-quality audio with a reasonable data footprint (file size). FLAC supports a range of sample resolutions, from 4-bit data per sample, up to 32-bit data sampling. It also supports a wide range of sample frequencies, from 1 Hz to 65,535 Hz (or 65 kHz), using 1 Hz increments; it is extremely flexible. From an audio playback hardware standpoint, I suggest using a 16-bit sample resolution and either a 44.1 kHz or a 48 kHz sample frequency, unless you’re targeting HD audio, in which case you should use 24-bit with 48 kHz for HD audio.

FLAC is supported in Android 3.1 and Java. Therefore, if your end users are using current Android devices, you should be able to safely utilize the FLAC codec. It is possible to use completely lossless new media assets in Android application development by using PNG8, PNG24, PNG32, and FLAC, as long as your application is targeting Android 3.1 or later hardware devices. Next, let’s take a look at another impressive open source codec.
Ogg Vorbis: A Lossy High-Performance Open Source Codec

Another open source digital audio codec supported by Android is the Ogg Vorbis format. This lossy audio codec is brought to you by the Xiph.Org Foundation. The Vorbis codec data is usually held inside an OGG audio data file extension, and thus, Vorbis is commonly called the Ogg Vorbis digital audio data format.

Ogg Vorbis supports sampling rates from 8 kHz to 192 kHz, and supports 255 discrete channels of digital audio. As you now know, this represents 8-bits worth of audio channels. Ogg Vorbis is supported across all Android versions or API-level releases.

Vorbis is quickly approaching the quality of MPEG HE-AAC and Windows Media Audio (WMA) Professional, and it is superior in quality to MP3, AAC-LC, and WMA. It's a lossy format, so the FLAC codec still has superior reproduction quality over Ogg Vorbis, as FLAC contains all the original digital audio sample data. Ogg Vorbis audio and Ogg Theora video are supported in HTML5.

MPEG-4: Advanced Audio Coding AAC-LC, AAC-ELD, or HE-AAC

Android, with a market share in the 90% range across all hardware devices, supports all the MPEG-4 AAC (Advanced Audio Coding) codecs, including AAC-LC, HE-AAC, and AAC-ELD. Java, which is the development and publishing platform that is nearing 90% market share among developers, also supports these codecs. AAC audio data samples are contained using MPEG-4 file “containers” (.3gp, .mp4, or .m4a file extensions). AAC-LC and HE-AAC can be decoded with all versions of Android. The AAC-ELD is only supported after Android OS 4.1. ELD stands for Enhanced Low Delay; this codec is intended for use in real-time, two-way communications applications, such as a digital walkie-talkies, or Dick Tracy–style smartwatch apps.

The simplest AAC codec is the AAC-LC (Low Complexity) codec, which is the most widely used. This is sufficient for most digital audio encoding applications. AAC-LC yields a higher quality result at a lower data footprint than the MP3 codec.

The most complicated AAC codec, HE-AAC (High Efficiency) codec, supports sampling rates from 8 kHz to 48 kHz, and both stereo and Dolby 5.1 channel encoding. Android decodes
both V1 and V2 levels of HE-AAC. Android can also encode audio using the HE-AAC-V1 codec in Android devices later than version 4.1.

Because of software patents, Audacity doesn’t include an MPEG-4 encoder. Be sure to download and install the free FFmpeg 2.2.2 encoder for Audacity, from http://lame.buanzo.org before you start Chapter 5, where you’ll use Audacity 2.1.1. You should have done this in Chapter 1, so just make sure that you have the libraries installed to maximize Audacity’s features!

**AMR: The MPEG-4 Adaptive Multi-Rate Audio Codecs for Voice**

For encoding speech, which usually features a different type of sound wave than music does, there are also two other AMR (Adaptive Multi-Rate) audio codecs, which are extremely efficient for encoding things like speech or “short-burst” sound effects.

There is an AMR-WB (Adaptive Multi-Rate Wideband) codec in Android that supports nine discrete settings, from a 6.6 kbps bit rate up to 23.85 kbps, sampled at 16 kHz. This is a pretty high sampling rate where voice is concerned! This is the codec to use on Narrator tracks, if you’re creating interactive e-book Android Studio applications, for example.

There’s also an AMR-NB (Adaptive Multi-Rate Narrowband) codec in Android that supports eight discrete settings, from 4.75 kbps to 12.2 kbps audio bit rates sampled at 8 kHz. This is an adequate sample rate if the data going into the codec is high quality or if resulting audio samples do not require high quality due to the noisy nature of the content (for example, a bomb blast).

**Pulse-Code Modulation : Windows WAV or Mac AIFF PCM Codecs**

Finally, almost all operating systems, including Windows, Mac OS, and Linux-based ones, such as Android, Tizen, Ubuntu, openSUSE, Blackberry, Firefox OS, Opera OS, and Chrome OS, support the PCM (pulse-code modulation) codecs, commonly known as the Windows WAV (WAV) audio format or the Apple AIFF (Audio Interchange File Format). Most of you are familiar with this lossless digital audio format from one of these two popular operating systems. It is lossless because there is zero compression applied. PCM audio is commonly used for CD-ROM content, as well as telephony applications. This is because PCM Wave audio is an uncompressed digital audio format. It has no CPU-intensive compression
algorithms applied to the data stream, and thus decoding (CPU overhead) is not an issue for telephony equipment or for CD players.

For this reason, when we start compressing digital audio assets into various file formats in Chapter 12, which covers digital audio data footprint optimization, you will use PCM as the “baseline” file format.

You probably won’t put PCM into Kindle (MOBI), Java (JAR), or Android (APK) distributable files, however, because there are other formats, such as FLAC and MPEG-4 AAC, which give you the same quality, and do it using an order of magnitude less data.

Ultimately, the only way to find out which audio formats supported by Android have the best digital audio result for any given audio data is to actually encode digital audio in all the primary codecs that you know are well supported and efficient. I show you how this is accomplished in Chapter 12.

Summary

In this chapter, you looked at the digital audio data encoding concepts, principles, and file formats used to compress and decompress digital audio assets, as well as to publish and distribute to end users. You also learned how sample resolution, sample frequency, bit rate, streaming, and HD audio can contribute to your digital audio sample’s quality and to its data footprint.

In the next chapter, you learn about digital audio data footprint optimization concepts, terms, and principles.
This chapter is a summary of the fundamental concepts of digital video.

If you are unfamiliar with video, this chapter will introduce the major issues, to acquaint you with the framework and nomenclature that you will need to address the rest of the book. If you are already knowledgeable about video, this chapter will provide a quick refresher, and will direct you to specific topics about which you’d like to learn more.

Imaging

The three-dimensional world is imaged by the lens of the human eye onto the retina, which is populated with photoreceptor cells that respond to light having wavelengths in the range of about 400 nm to 700 nm. In an imaging system, we build a camera having a lens and a photosensitive device, to mimic how the world is perceived by vision.

Although the shape of the retina is roughly a section of a sphere, it is topologically two-dimensional. In a camera, for practical reasons, we employ a flat image plane, sketched in Figure 1.1 overleaf, instead of a spherical image surface. Image system theory concerns analyzing the continuous distribution of power that is incident on the image plane.

A photographic camera has, in the image plane, film that is subject to chemical change when irradiated by
light. The active ingredient of photographic film is contained in a thin layer of particles having carefully controlled size and shape, in a pattern with no coherent structure. If the particles are sufficiently dense, an image can be reproduced that has sufficient information for a human observer to get a strong sense of the original scene. The finer the particles and the more densely they are arranged in the film medium, the higher will be the capability of the film to record spatial detail.

**Digitization**

Signals captured from the physical world are translated into digital form by *digitization*, which involves two processes. A signal is digitized when it is subjected to both *sampling* and *quantization*, in either order. When an audio signal is sampled, the single dimension of time is carved into discrete intervals. When an image is sampled, two-dimensional space is partitioned into small, discrete regions. Quantization assigns an integer to the amplitude of the signal in each interval or region.

1-D sampling

A signal that is a continuous one-dimensional function of time, such as an audio signal, is sampled through forming a series of discrete values, each of which represents the signal at an instant of time. *Uniform sampling*, where the time intervals are of equal duration, is ubiquitous.

2-D sampling

A continuous two-dimensional function of space is sampled by assigning, to each element of a sampling...
grid, a value that is a function of the distribution of intensity over a small region of space. In digital video and in conventional image processing, the samples lie on a regular, rectangular grid.

Samples need not be digital: A CCD camera is inherently sampled, but it is not inherently quantized. Analog video is not sampled horizontally, but is sampled vertically by scanning, and sampled temporally at the frame rate.

**Pixel array**

A digital image is represented by a matrix of values, where each value is a function of the information surrounding the corresponding point in the image. A single element in an image matrix is a *picture element*, or *pixel*. In a color system, a pixel includes information for all color components. Several common formats are sketched in Figure 1.2 below.

In computing it is conventional to use a sampling grid having equal horizontal and vertical sample pitch – *square pixels*. The term *square* refers to the sample pitch; it should not be taken to imply that image information associated with the pixel is distributed uniformly throughout a square region. Many video systems use sampling grids where the horizontal and vertical sample pitch are not equal.

![Figure 1.2 Pixel array.](image-url)
In computing it is usual to represent a grayscale or pseudocolor pixel as a single 8-bit byte. It is common to represent a truecolor pixel as three 8-bit red, green, and blue \( (R'G'B') \) components totaling three bytes – 24 bits – per pixel.

### Spatiotemporal domains

A digital video image is sampled in the horizontal, vertical, and temporal axes, as indicated in Figure 1.3 above. One-dimensional sampling theory applies along each of these axes. At the right is a portion of the two-dimensional spatial domain of a single image. Some spatial processing operations cannot be separated into horizontal and vertical facets.

### Scanning notation

In computing, a display is described by the count of pixels across the width and height of the image. Conventional television would be denoted \( 644 \times 483 \), which indicates 483 picture lines. But any display system involves some scanning overhead, so the total number of lines in the raster of conventional video is necessarily greater than 483.
Video scanning systems have traditionally been denoted by their total number of lines including sync and blanking overhead, the frame rate in hertz, and an indication of interlace (2:1) or progressive (1:1) scan, to be introduced on page 11.

525/59.94/2:1 scanning is used in North America and Japan, with an analog bandwidth for studio video of about 5.5 MHz.

625/50/2:1 scanning is used in Europe and Asia, with an analog bandwidth for studio video of about 6.5 MHz. For both 525/59.94 and 625/50 component digital video according to ITU-R Rec. BT.601-4 (“Rec. 601”), the basic sampling rate is exactly 13.5 MHz. Bandwidth and sampling rate will be explained in later sections.

1125/60/2:1 scanning is in use for high-definition television (HDTV), with an analog bandwidth of about 30 MHz. The basic sampling rate for 1125/60 is 74.25 MHz. A variant 1125/59.94/2:1 is in use. This scanning system was originally standardized with a 1920×1035 image having pixels about 4 percent taller than square.

1920×1080 The square-pixel version of 1125/60 is now commonly referred to as 1920×1080.

1280×720 A progressive-scan one megapixel image format is proposed for advanced television in the United States.

**Viewing distance and angle**

A viewer tends to position himself or herself relative to a scene so that the smallest detail of interest in the scene subtends an angle of about one minute of arc (1/60°), approximately the limit of angular discrimination for normal vision. For the 483 picture lines of conventional television, the corresponding viewing distance is about seven times picture height (PH); the horizontal viewing angle is about 11°. For the 1080 picture lines of HDTV, the optimum viewing distance is 3.3 screen heights, and the horizontal viewing angle is almost tripled to 28°. The situation is sketched in Figure 1.4 overleaf.
To achieve a viewing situation where a pixel subtends $1'/60^\circ$, viewing distance expressed in units of picture height should be about $3400$ divided by the number of picture lines. A computer user tends to position himself or herself closer than this – about 50 to 60 percent of this distance – but at this closer distance individual pixels are discernible. Consumer projection television is viewed closer than $7\times PH$, but at this distance scan lines become objectionable.

**Aspect ratio**

Variants of conventional 525/59.94 systems having 16:9 aspect ratio have recently been standardized, but few are deployed as I write this.

**Frame rate, refresh rate**

A succession of flashed still pictures, captured and displayed at a sufficiently high rate, can create the illusion of motion. The quality of the motion portrayal depends on many factors.
Most displays for moving images involve a period of time when the reproduced image is absent from the display, that is, a fraction of the frame time during which the display is black. In order to avoid objectionable flicker, it is necessary to flash the image at a rate higher than the rate necessary to portray motion. Refresh rate is highly dependent on the ambient illumination in the viewing environment: The brighter the environment, the higher the flash rate must be in order to avoid flicker. To some extent the brightness of the image itself influences the flicker threshold, so the brighter the image, the higher the refresh rate must be. Since peripheral vision has higher temporal sensitivity than central (foveal) vision, the flicker threshold of vision is also a function of the viewing angle of the image.

Refresh rate is generally engineered into a system. Once chosen, it cannot easily be changed. Different applications have adopted different refresh rates, depending on the image quality requirements and viewing conditions of the application.

In the darkness of a cinema, a flash rate of 48 Hz is adequate. In the early days of motion pictures, a frame rate of 48 Hz was thought to involve excessive expenditure for film stock, and 24 frames per second were found to be sufficient to portray motion. So, a conventional film projector flashes each frame twice. Higher realism can be obtained with specialized cameras and projectors that operate at higher frame rates, up to 60 frames per second or more.

In a dim viewing environment typical of television viewing, such as a living room, a flash rate of 60 Hz is sufficient. Originally, television refresh rates were chosen to match the local AC power line frequency.

In a bright environment such as an office, a refresh rate above 70 Hz might be required.
Motion portrayal

It is conventional in video for each element of an image sensor device to integrate light from the scene for the entire frame time. This captures as much of the light from the scene as possible, in order to maximize sensitivity and/or signal-to-noise ratio. In an interlaced camera, the exposure time is usually effectively the duration of the field, not the duration of the frame. This is necessary in order to achieve good motion portrayal.

If the image has elements that move an appreciable distance during the exposure time, then the sampled image information will exhibit smear. Smear can be minimized by using an exposure time that is a fraction of the frame time; however, the method involves discarding light from the scene and a sensitivity penalty is incurred.

When the effect of image information incident during a single frame time persists into succeeding frames, the sensor exhibits lag. Lag is a practical problem for tube-type cameras, but generally not a problem for CCD cameras.

Flicker is absent in any image display device that produces steady, unflashing light for the duration of the frame time. You might think that a nonflashing display would be more suitable than a device that flashes, and many contemporary devices do not flash. However, if the viewer's gaze is tracking an element that moves across the display, a display with an on-time approaching the frame time will exhibit smearing of elements that move. This problem becomes more severe as eye tracking rates increase; for example, with the wide viewing angle of high-definition television.

Raster scanning

In cameras and displays, some time is required to advance the scanning operation – to retrace – from one line to the next and from one picture to the next. These intervals are called blanking intervals, because in a

conventional CRT display the electron beam must be extinguished (blanked) during these time intervals. The horizontal blanking time lies between scan lines, and vertical blanking lies between frames (or fields). Figure 1.5 above shows the raster structure of 525/59.94 and 625/50 digital video systems, including these blanking intervals. In analog video, sync information is conveyed during the blanking intervals.

The horizontal and vertical blanking intervals required for a CRT display are quite large fractions of the line time and frame time: in 525/59.94, 625/50, and 1920×1035 systems, vertical blanking occupies 8 percent of each frame period. Although in principle a digital video interface could omit the blanking intervals and use a clock having a lower frequency than the sampling clock, this would be impractical. Digital video standards use interface clock frequencies chosen to

525/59.94 is colloquially referred to as NTSC, and 625/50 as PAL, but the terms NTSC and PAL properly apply to color encoding standards and not to scanning standards.
match the large blanking intervals of typical display equipment. Good use is made in digital systems of what would otherwise be excess data capacity: A digital video interface may convey audio signals during blanking; a digital video tape recorder might record error correction information in these intervals.

In analog video, information in the image plane is scanned uniformly left to right during a fixed, short interval of time – the active line time – and conveyed as an analog electrical signal. There is a uniform mapping from horizontal position in the image to time instant in the electrical signal. Successive lines are scanned uniformly from the top of the image to the bottom, so there is also a uniform mapping from vertical position in the image to time instant in the electrical signal. The fixed pattern of parallel scanning lines disposed across the image is the raster. The word is derived from the Greek rake, from the resemblance of a raster to the pattern left on a newly raked field.

Figure 1.6 above shows the waveform of a single scan line, showing voltage from 0 V to 700 mV in a component analog system (with sync at -300 mV), and code-
Interlace

At the outset of television, the requirement to minimize information rate for transmission – and later, recording – led to interlaced scanning. Each frame is scanned in two successive vertical passes, first the odd field, then the even field, whose scan lines interlace as illustrated Figure 1.7 below. Total information rate is reduced because the flicker susceptibility of vision is due to a wide-area effect. As long as the complete height of the picture is scanned rapidly enough to overcome wide-area flicker, small-scale picture information – such as that in the alternate lines – can be transmitted at a lower rate.

If the information in an image changes vertically at a scale comparable to the scanning line pitch – if a fine pattern of black-and-white horizontal line pairs is scanned, for example – then interlace can cause the content of the odd and the even fields to differ markedly. This causes twitter, a small-scale phenomenon that is perceived as extremely rapid up-and-down motion. Twitter can be produced not only from degenerate images such as fine horizontal black-and-white lines, but also from high-amplitude brightness detail in an ordinary image. In computer generated imagery (CGI), twitter can be reduced by vertical filtering.

If image information differs greatly from one field to the next, then instead of twitter, large-scale flicker will
result. A video camera is designed to avoid introduction of so much vertical detail that flicker could be produced. In synthetic image generation, vertical detail may have to be explicitly filtered in order to avoid flicker.

Scanning standards

Conventional broadcast television scans a picture whose aspect ratio is 4:3, in left-to-right, top-to-bottom order using interlaced scanning.

A scanning system is denoted by its total line count and its field rate in hertz, separated by a solidus (slash). Two scanning standards are established for conventional television: 525/59.94, used primarily in North America and Japan; and 625/50, used elsewhere. It is obvious from the scanning nomenclature that the line counts and frame rates are different. There are other important differences:

<table>
<thead>
<tr>
<th>System</th>
<th>525/59.94</th>
<th>625/50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture:Sync ratio</td>
<td>10:4</td>
<td>7:3</td>
</tr>
<tr>
<td>Setup, percent</td>
<td>7.5</td>
<td>0</td>
</tr>
<tr>
<td>Count of equalization, broad pulses</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Line number 1, and 0, defined at first equalization pulse</td>
<td>First broad pulse</td>
<td>First broad pulse</td>
</tr>
</tbody>
</table>

The two systems have gratuitous differences in other parameters unrelated to scanning.

Systems with 525/59.94 scanning usually employ NTSC color coding, and systems with 625/50 scanning usually use PAL, so 525/59.94 and 625/50 systems are loosely referred to as NTSC and PAL. But NTSC and PAL properly refer to color encoding. Although 525/59.94/NTSC and 625/50/PAL systems dominate worldwide broadcasting, other combinations of scanning and color coding are in use in large and important regions of the world, such as France, Russia, and South America.

525/59.94 video in Japan uses 10:4 picture to sync ratio and zero setup.

Monochrome systems having 405/50/2:1 and 819/50/2:1 scanning were once used in Britain and France, respectively, but transmitters for these standards have now been decommissioned.
The frame rate of 525/59.94 video is exactly \( \frac{60}{1.001} \) Hz. In 625/50 the frame rate is exactly 50 Hz. Computer graphics systems have various frame rates with few standards and poor tolerances.

An 1125/60/2:1 high-definition television production system has been adopted as SMPTE Standard 240M and has been proposed to the ITU-R. At the time of writing, the system is in use for broadcasting in Japan but no international broadcasting standards have been agreed upon.

All of these scanning systems are interlaced 2:1, and interlace is implicit in the scanning nomenclature. Noninterlaced scanning is common in desktop computers and universal in computer workstations. Emerging high-definition television standards have interlaced and noninterlaced variants.

Standards conversion refers to conversion among scanning standards. Standards conversion, done well, is difficult and expensive. Standards conversion between scanning systems having different frame rates, even done poorly, requires a fieldstore or framestore. The complexity of standards conversion between 525/59.94 scanning and 625/50 scanning is the reason that it is difficult for consumers – and broadcasters – to convert European material for use in North America or Japan, or vice versa.

Transcoding refers to changing the color encoding of a signal, without altering its scanning system.

Sync structure

At a video interface, synchronization (sync) is achieved by associating, with every scan line, a line sync datum denoted \( 0_H \) (pronounced zero-\( H \)). In component digital video, sync is conveyed using digital codes 0 and 255 outside the range of picture information. In analog video, sync is conveyed by voltage levels “blacker than black.” \( 0_H \) is defined by the 50-percent point of the leading (falling) edge of sync.

In both 525/59.94 and 625/50 video the normal sync pulse has a duration of 4.7 µs. Vertical sync is identified by broad pulses, which are serrated in order for a receiver to maintain horizontal sync even during the vertical interval. Narrow equalization pulses, half the sync pulse duration at twice the line rate, are present during intervals immediately before and immediately following the broad pulses.

When analog sync separators comprised just a few resistors and capacitors, to achieve stable interlacing required halving the duration of the line syncs and introducing additional pulses halfway between them. Originally the equalization pulses were the ones interposed between the line syncs, but the term now refers to all of the narrow pulses. The absence of sync level between the end of a broad pulse and the start of the following sync was called serration. If you think of field sync as a single pulse asserted for several lines, serration is the negation of this pulse at twice the line rate.

An equalization pulse has half the duration of a normal sync. The duration of a vertical (broad) pulse is half the line time, less a full sync width. A 525/59.94 system has three lines of preequalization pulses, three lines of vertical sync, and three lines of postequalization pulses. A 625/50 system has two and one-half lines (five pulses) of each of preequalization, broad, and postequalization pulses. Figure 1.8 above sketches the vertical sync component of 525/59.94 analog video.

Monochrome 525-line broadcasting originated with a line rate of exactly 15.750 kHz. When color was intro-
duced to NTSC in 1953, the monochrome horizontal frequency was multiplied by exactly \( \frac{1000}{1001} \) to obtain the NTSC color line rate of approximately 15.734 kHz. Details are in *Field, frame, line, and sample rates*, on page 199. All 525-line broadcast signals – even monochrome signals – now employ this rate. The line rate of 625/50 systems has always been exactly 15.625 kHz, corresponding to a line time of exactly 64 \( \mu \)s.

**Data rate**

- \( b = \text{bit} \)
- \( B = \text{Byte} \)

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Value</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>10³</td>
<td>1000</td>
</tr>
<tr>
<td>( K )</td>
<td>2¹⁰</td>
<td>1024</td>
</tr>
</tbody>
</table>

**SI, datacom:**
- \( M \) 10⁶ 1 000 000

**Disk:**
- \( M \) 10³ · 2¹⁰ 1 024 000

**RAM:**
- \( M \) 2²⁰ 1 048 576

**Data rate of digital video**

*Data rate* of a digital system is measured in bits per second (b/s) or bytes per second (B/s), where a byte is eight bits. The formal, international designation of the metric system is *Système International d’Unités*, SI. The SI prefix \( k \) denotes 10³ (1000); it is often used in data communications. The \( K \) prefix used in computing denotes 2¹⁰ (1024). The SI prefix \( M \) denotes 10⁶ (1 000 000). Disk storage is generally allocated in units integrally related to 1024 bytes; the prefix \( M \) applied to disk storage denotes 1 024 000. RAM memory generally has capacity based on powers of two; the prefix \( M \) applied to RAM denotes 2²⁰ or 1024 K (1 048 576).

*Line rate* is an important parameter of a video system: Line rate is simply the frame rate multiplied by the number of lines per total frame.

The aggregate *data rate* is the number of bits per pixel, times the number of pixels per line, times the number of lines per frame, times the frame rate.

In both analog and digital video it is necessary to convey not only the raw image information, but also information about which time instants (or which samples) are associated with the start of frame, or the start of line. This information is conveyed by signal synchronization or *sync* elements. In analog video and composite digital video, sync is combined with video by being coded at a level *blacker than black*. 
All computer graphics systems and almost all digital video systems have the same integer number of sample clock periods in every raster line. In these cases, sampling frequency is simply the line rate times the number of samples per total line (S/TL).

In 625/50 PAL there is not an exact integer number of samples per line: Samples in successive lines are offset to the left a small fraction, $\frac{4}{625}$ of the horizontal sample pitch. The sampling structure is not precisely orthogonal, although digital acquisition, processing, and display equipment treat it so.

The data capacity required for the active pixels of a frame is computed by simply multiplying the number of bits per pixel by the number of active pixels per line, then by the number of active lines per frame. To compute the data rate for the active pixels, simply multiply by the frame rate.

Standards are not well established in display systems used in desktop computers, workstations, and industrial equipment. The absence of published data makes it difficult to determine raster scanning parameters.

**Linearity**

A video system should ideally satisfy the principle of superposition; in other words, it should exhibit linearity. A function $f$ is linear if and only if (iff):

$$f(a + b) = f(a) + f(b)$$

The function $f$ can encompass an entire system: A system is linear iff the sum of the individual responses of the system to any two signals is identical to its response to the sum of the two. Linearity can pertain to steady-state response, or to the system’s temporal response to a changing signal.

Linearity is a very important property in mathematics, in signal processing, and in video. But linearity in one domain cannot be carried across to another domain if
a nonlinear function separates the two. An image signal usually originates in a sensor that has linear response to physical intensity. And video signals are usually processed through analog circuits that have linear response to voltage or digital systems that are linear with respect to the arithmetic performed on the code-words. But a video camera applies a nonlinear transfer function – *gamma correction* – to the image signal. So the image signal is in a linear optical domain, and the video signal is in a linear electrical domain, but the two domains are not the same.

**Perceptual uniformity**

A system is *perceptually uniform* if a small perturbation to a component value is approximately equally perceptible across the range of that value. The volume control on your radio is designed to be perceptually uniform: Rotating the knob 10 degrees produces approximately the same perceptual increment in volume anywhere across the range of the control. If the control were physically linear, the logarithmic nature of loudness perception would place all of the perceptual “action” of the control at the bottom of its range. Figure 1.9, in the margin, shows the transfer function of a potentiometer with standard *audio taper*.

The CIE $L^*$ system, to be described on page 88, assigns a perceptually uniform scale to lightness. Video signals are coded according to perceptual principles, as will be explained in Chapter 6, *Gamma*, on page 91.

**Noise, signal, sensitivity**

Any analog electronic system is inevitably subject to noise that is unrelated to the signal to be processed by the system. As signal amplitude decreases, the noise makes a larger and larger relative contribution. In analog electronics, noise is inevitably introduced from thermal sources, and perhaps also from nonthermal sources of interference.

In addition to random noise, processing of a signal may introduce distortion that is correlated to the signal
itself. For the purposes of objective measurement of the performance of a system, distortion is treated as noise. Depending on its nature, distortion may be more or less perceptible than random noise.

*Signal-to-Noise Ratio* (SNR) is the ratio of a specified signal, often the reference amplitude or largest amplitude signal that can be carried by a system, to the amplitude of undesired components including noise and distortion. SNR is expressed in units of *decibels* (dB), a logarithmic measure.

*Sensitivity* refers to the minimum signal power that achieves acceptable (or specified) SNR performance.

### Quantization

A signal whose amplitude takes a range of continuous values is *quantized* by assigning to each of a finite set of intervals of amplitude a discrete, numbered level. In *uniform quantization* the steps between levels have equal amplitude. The degree of visual impairment caused by noise in a video signal is a function of the properties of vision. In video, it is ubiquitous to digitize a signal that is a nonlinear function, usually a 0.45-power function, of physical (linear-light) intensity. The function chosen minimizes the visibility of noise.

The effect of quantizing to a finite number of discrete amplitude levels is equivalent to adding *quantization noise* to the ideal levels of a quantized signal. Quantization has the effect of introducing noise, and thereby diminishes the SNR of a digital system. Eight-bit quantization has a theoretical SNR limit of about 56 dB (peak signal to rms noise).

If an input signal has very little noise, then situations can arise when the quantized value is quite predictable at some points, but when the signal is near the edge of a quantizer step, uncertainty in the quantizer is reflected as noise. This situation can cause the reproduced image to exhibit *noise modulation*. It is beneficial to introduce roughly a quantizer step’s worth of

\[
20 \log_{10}\left( k \sqrt{12} \right)
\]
noise (peak to peak) prior to quantization, to avoid this effect. This introduces a very small amount of noise in the picture, but guarantees avoidance of “patterning” of the quantization.

Quantization can be applied to a unipolar signal such as luma. For a bipolar signal such as a color difference it is standard to use a mid-tread quantizer, such as the one sketched in Figure 1.10 in the margin, so that no systematic error affects the zero value.

**Frequency response, bandwidth**

Figure 1.11 below shows a test signal starting at zero frequency and sweeping up to some high frequency. The response of a typical electronic system is shown in the middle graph; the response diminishes at high frequency. The envelope of that waveform – the system’s frequency response – is shown at the bottom.

Figure 1.11 *Frequency response* of any electronic or optical system falls as frequency increases. Bandwidth is measured at the half-power point (-3 dB), where response has fallen to 0.707. Television displays are often specified at limiting resolution, where response has fallen to 0.1.
Loosely speaking, *bandwidth* is the rate at which information in a signal can change from one state to another. The response of an electronic system deteriorates above a certain information rate. Bandwidth is specified or measured at the frequency where amplitude has fallen 3 dB from its value at zero frequency (called *DC*) – that is, to the fraction 0.707 of its value at DC.

The rate at which an analog video signal can change from one state to another, say from white to black, is limited by the bandwidth of the video system. This places an upper bound on *horizontal resolution*. Consumer video generally refers to horizontal resolution, measured as the number of black and white elements (*TV lines*) that can be discerned over a horizontal distance equal to the picture height.

### Bandwidth and data rate

Data rate does not apply directly to an analog system, and the term *bandwidth* does not properly apply to a digital system. When a digital system conveys a sampled representation of a continuous signal, as in digital video or digital audio, the bandwidth represented by the digitized signal is necessarily less than half – typically about 0.45 – of the sampling rate.

When arbitrary digital information is conveyed through an analog channel, as by a modem, the data rate that can be achieved depends on bandwidth, noise, and other properties of the channel. Figure 1.12, in the margin, shows a simple scheme that transmits two bits per second per hertz of bandwidth, or 2400 b/s for a channel having 1200 Hz analog bandwidth. The bottom sketch shows that if each half-cycle conveys one of sixteen amplitude levels, providing the channel has sufficiently low noise, four bits can be coded per half-cycle. The rate at which the signal in the channel can change state – the *symbol rate* or *baud rate* – is constant at 2400 baud, but this modulation method has a *data rate* or *bit rate* of 9600 b/s.
Resolution

As picture detail increases in frequency, the response of an imaging system will eventually deteriorate. In image science and in television, resolution refers to the capability of an imaging system to reproduce fine detail in the picture.

The absolute upper limit to resolution in a digital image system is the number of pixels over the width and height of a frame, and is the way the term resolution is used in computing.

In conventional North American television, 483 scan lines cover the height of the image. High-definition television systems use up to 1080 picture lines. The amount of information that can be captured in a video signal is bounded by the number of picture lines. But other factors impose limits more severe than the number of lines per picture height.

In an interlaced system, vertical resolution must be reduced substantially from the scan-line limit, in order to avoid producing a signal that will exhibit objectionable twitter upon display.

Resolution in film

In film, resolution is measured as the finest pattern of straight, parallel lines that can be reproduced, expressed in line pairs per millimeter (lp/mm). A line pair contains a black region and a white region.

Motion picture film is conveyed vertically through the camera and projector, so the width – not the height – of the film is 35 mm. Cinema usually has an aspect ratio of 1.85:1, so the projected film area is about 21 mm × 11 mm, only three-tenths of the 36 mm × 24 mm projected area of 35 mm still film.

The limit to the resolution of motion picture film is not the static response of the film, but judder and weave in the camera and the projector.
Resolution in television

In video, resolution refers to the number of line pairs (cycles) resolved on the face of the display screen, expressed in cycles per picture height (C/PH) or cycles per picture width (C/PW). A cycle is equivalent to a line pair of film. In a digital system, it takes at least two samples – pixels, scanning lines, or TV lines – to represent a line pair. However, resolution may be substantially less than the number of pixel pairs due to optical, electro-optical, and electrical filtering effects. Limiting resolution is defined as the frequency where detail is recorded with just 10 percent of the system’s low-frequency response.

In consumer television, the number of scanning lines is fixed by the raster standard, but the electronics of transmission, recording, and display systems tend to limit bandwidth and reduce horizontal resolution. Consequently, in consumer electronics the term resolution generally refers to horizontal resolution. Confusingly, horizontal resolution is expressed in units of lines per picture height, so once the number of resolvable lines is measured, it must be corrected for the aspect ratio of the picture. Resolution in TV lines per picture height is twice the resolution in cycles per picture width, divided by the aspect ratio of the picture.

Resolution in computer graphics

In computer graphics, resolution is simply the number of discrete vertical and horizontal pixels required to store the digitized image. For example, a 1152×900 system has a total of about one million pixels (one megapixel, or 1 Mpx). Computer graphics is not generally very concerned about whether individual pixels can be discerned on the face of the display. In most color computer systems, an image comprising a one-pixel black-and-white checkerboard actually displays as a uniform gray, due to poor high-frequency response in the cable and video amplifiers, and due to rather large spot size at the CRT.
Computer graphics often treats each pixel as representing an idealized rectangular area independent of all other pixels. This notion discounts the correlation among pixels that is an inherent and necessary aspect of image acquisition, processing, compression, display, and perception. In fact the rather large spot produced by the electron beam of a CRT and the arrangement of phosphor triads on the screen, suggested by Figure 1.13, produces an image of a pixel on the screen that bears little resemblance to a rectangle. If pixels are viewed at a sufficient distance, these artifacts are of little importance. However, imaging systems are forced by economic pressures to make maximum perceptual use of the delivered pixels, consequently we tend to view CRTs at close viewing distances.

**Luma**

As you will see in *Luma and color differences*, on page 155, a video system conveys image data in the form of a component that represents brightness, and two other components that represent color. It is important to convey the brightness component in such a way that noise (or quantization) introduced in transmission, processing, and storage has a perceptually similar effect across the entire tone scale from black to white. Ideally, these goals would be accomplished by forming a true CIE luminance signal as a weighted sum of linear-light red, green, and blue; then subjecting that luminance to a nonlinear transfer function similar to the CIE $L^*$ function that will be described on page 88.

There are practical reasons in video to perform these operations in the opposite order. First a nonlinear transfer function – *gamma correction* – is applied to each of the linear $R$, $G$, and $B$. Then a weighted sum of the nonlinear components is computed to form a luma signal, $Y'$, representative of brightness.

In effect, video systems approximate the lightness response of vision using RGB intensity signals, each raised to the 0.45 power. This is comparable to the $\frac{1}{3}$ power function defined by $L^*$.
The coefficients that correspond to the so-called NTSC red, green, and blue CRT phosphors of 1953 are standardized in Recommendation ITU-R BT. 601-4 of the ITU Radiocommunication Sector (formerly CCIR). I call it Rec. 601. To compute nonlinear video luma from nonlinear red, green, and blue:

\[ 601'Y' = 0.299 R' + 0.587 G' + 0.114 B' \]

The prime symbols in this equation, and in those to follow, denote nonlinear components.

The unfortunate term “video luminance”

Unfortunately, in video practice, the term luminance has come to mean the video signal representative of luminance even though the components of this signal have been subjected to a nonlinear transfer function. At the dawn of video, the nonlinear signal was denoted \( Y' \), where the prime symbol indicated the nonlinear treatment. But over the last 40 years the prime has been elided and now both the term luminance and the symbol \( Y \) collide with the CIE, making both ambiguous! This has led to great confusion, such as the incorrect statement commonly found in computer graphics and color textbooks that in the YIQ or YUV color spaces, the Y component is CIE luminance! I use the term luminance according to its standardized CIE definition and use the term luma to refer to the video signal, and I am careful to designate the nonlinear quantity with a prime symbol. But my convention is not yet widespread, and in the meantime you must be careful to determine whether a linear or nonlinear interpretation is being applied to the word and the symbol.

Color difference coding

In component video, the three components necessary to convey color information are transmitted separately.

The data capacity accorded to the color information in a video signal can be reduced by taking advantage of the relatively poor color acuity of vision, providing full
luma bandwidth is maintained. It is ubiquitous to base color difference signals on blue minus luma and red minus luma \((B' - Y', R' - Y')\). Luma and \((B' - Y', R' - Y')\) can be computed from \(R', G',\) and \(B'\) through a \(3 \times 3\) matrix multiplication. Once luma and color difference – or chroma – components have been formed, the chroma components can be subsampled (filtered).

In component digital video, \(C_B\) and \(C_R\) components scaled from \((B' - Y', R' - Y')\) are formed.

In component analog video, \(P_B\) and \(P_R\) color difference signals scaled from \((B' - Y', R' - Y')\) are lowpass filtered to about half the bandwidth of luma.

In Figure 1.14 above, the left-hand column sketches a \(2 \times 2\) array of \(R'G'B'\) pixels that, with 8 bits per sample, would occupy a total of 12 bytes. This is denoted \(4:4:4\) \(R'G'B'\). \(Y' C_B C_R\) components can be formed from \(R'G'B'\), as shown in the second column; without subsampling, this is denoted \(4:4:4 Y' C_B C_R\).

The use of 4 as the numerical basis for subsampling notation is a historical reference to a sample rate of about four times the color subcarrier frequency.
4:2:2 Y’C_B C_R digital video according to Rec. 601 uses 4:2:2 sampling: Chroma components are subsampled by a factor of 2 along the horizontal axis. Chroma samples are coincident (cosited) with alternate luma samples.

In an 8-bit system using 4:2:2 coding, the $2 \times 2$ array occupies 8 bytes, and the aggregate data capacity is 16 bits per pixel. For studio digital video, the raw data rate is 27 MB/s.

4:1:1 A few digital video systems have used 4:1:1 sampling, where the chroma components are subsampled by a factor of 4 horizontally.

4:2:0 JPEG, H.261, MPEG-1, and MPEG-2 usually use 4:2:0 sampling. $C_B$ and $C_R$ are each subsampled by a factor of 2 both horizontally and vertically; $C_B$ and $C_R$ are cosited vertically halfway between scan lines. Horizontal subsampling is inconsistent. In MPEG-2, $C_B$ and $C_R$ are cosited horizontally. In JPEG, H.261, and MPEG-1, $C_B$ and $C_R$ are not cosited horizontally; instead, they are sited halfway between alternate luma samples.

MAC A transmission system for analog components – *Multiplexed Analog Components*, or MAC – has been adopted in Europe for direct broadcast from satellite (DBS). In MAC, the color difference components are not combined with each other or with luma, but are time-compressed and transmitted serially. MAC is not standardized by ITU-R.

Component digital video, 4:2:2

The standard interface for 4:2:2 component digital video is Rec. ITU-R 601-4. It specifies sampling of luma at 13.5 MHz and sampling of $C_B$ and $C_R$ color difference components at 6.75 MHz. This interface is referred to as 4:2:2, since luma is sampled at four times 3.375 MHz, and each of the $C_B$ and $C_R$ components at twice 3.375 MHz – that is, the color difference signals are horizontally subsampled by a factor of 2:1 with respect to luma. Sampling at 13.5 MHz results in an integer number of samples per total line (S/TL) in both
525/59.94 systems (858 S/TL) and 625/50 systems (864 S/TL). Luma is sampled with 720 active samples per line in both 525/59.94 and 625/50.

Component digital video tape recorders are widely available for both 525/59.94 and 625/50 systems, and have been standardized with the designation D-1. That designation properly applies to the tape format, not the signal interface.

Rec. 601 specifies luma coding that places black at code 16 and white at code 235. Color differences are coded in offset binary, with zero at code 128, the negative peak at code 16, and the positive peak at code 240.

Composite video

The terms NTSC and PAL are often used incorrectly to refer to scanning standards. Since PAL encoding is used with both 625/50 scanning (with two different subcarrier frequencies) and 525/59.94 scanning (with a third subcarrier frequency), the term PAL alone is ambiguous. The notation CCIR is sometimes used to refer to 625/50 scanning, but that is confusing because the former CCIR – now ITU-R – standardized all scanning systems, not just 625/50.

In composite NTSC and PAL video, the color difference signals required to convey color information are combined by the technique of quadrature modulation into a chroma signal using a color subcarrier of about 3.58 MHz in conventional NTSC and about 4.43 MHz in conventional PAL. Luma and chroma are then summed into a composite signal for processing, recording, or transmission. Summing combines brightness and color into one signal, at the expense of introducing a certain degree of mutual interference.

The frequency and phase of the subcarrier are chosen and maintained carefully: The subcarrier frequency is chosen so that luma and chroma, when they are summed, are frequency interleaved. Studio signals have coherent sync and color subcarrier; that is, subcarrier is phase-locked to a rational fraction of the line rate; generally this is achieved by dividing both from a single master clock. In industrial and consumer video, subcarrier usually free-runs with respect to line sync.

Transcoding among different color encoding methods having the same raster standard is accomplished by luma/chroma separation, color demodulation, and color remodulation.

Transport, electrical, and mechanical aspects of 4:2:2 interface are specified in Rec. 656. See page 248.
Composite digital video, $4f_{SC}$

The earliest digital video equipment processed signals in composite form. Processing of digital composite signals is simplified if the sampling frequency is an integer multiple of the color subcarrier frequency. Nowadays, a multiple of four is used: *four-times-subcarrier*, or $4f_{SC}$. For NTSC systems it is standard to sample at about 14.3 MHz. For PAL systems the sampling frequency is about 17.7 MHz.

Composite digital processing was necessary in the early days of digital video, but most image manipulation operations cannot be accomplished in the composite domain. During the 1980s there was widespread deployment of component digital processing equipment and component videotape recorders (DVTRs), recording 4:2:2 signals using the D-1 standard.

However, the data rate of a component 4:2:2 signal is roughly twice that of a composite signal. Four-times-subcarrier composite digital coding was resurrected to enable a cheap DVTR; this became the D-2 standard. The D-2 DVTR offers the advantages of digital recording, but retains the disadvantages of composite NTSC or PAL: Luma and chroma are subject to cross-contamination, and the pictures cannot be manipulated without decoding and reencoding.

The development and standardization of D-2 recording led to the standardization of composite $4f_{SC}$ digital parallel and serial interfaces, which essentially just code the raw 8- or 10-bit composite data stream. These interfaces share the electrical and physical characteristics of the standard 4:2:2 interface, but with about half the data rate. For 8-bit sampling this leads to a total data rate of about 14.3 MB/s for 525/59.94 NTSC, and about 17.7 MB/s for 625/50 PAL.

Analog interface

Video signal amplitude levels in 525/59.94 systems are expressed in IRE units, named after the Institute of Radio Engineers in the United States, the predecessor
Reference blanking level is defined as 0 IRE, and reference white level is 100 IRE. The range between these values is the picture excursion.

Composite 525/59.94 systems have a picture-to-sync ratio of 10:4; consequently, the sync level of a composite 525/59.94 signal is -40 IRE. In composite NTSC systems, except in Japan, reference black is setup the fraction 7.5 percent \((3/40)\) of the reference blanking-to-white excursion: Composite 525/59.94 employs a pedestal of 7.5 IRE. There are exactly 92.5 IRE from black to white: The picture excursion of a 525/59.94 signal is about 661 mV.

Setup has been abolished from component digital video and from HDTV. Many 525/59.94 component analog systems have adopted zero setup, and have 700 mV excursion from black to white, with 300 mV sync. But many component analog systems use setup, and it is a nuisance in design and in operation.

625/50 systems have a picture-to-sync ratio of 7:3, and zero setup. Picture excursion (from black to white) is exactly 700 mV; sync amplitude is exactly 300 mV. Because the reference levels are exact in millivolts, the IRE unit is rarely used, but in 625/50 systems an IRE unit corresponds to exactly 7 mV.

A video signal with sync is distributed in the studio with blanking level at zero \(0 \text{ V}_{\text{DC}}\) and an amplitude from synctip to reference white of one volt into an impedance of 75 Ω. A video signal without sync is distributed with blanking level at zero, and an amplitude from blanking to reference white of either 700 mV or 714 mV.

High-definition television, HDTV

*High-definition television* (HDTV) is defined as having twice the vertical and twice the horizontal resolution of conventional television, a picture aspect ratio of 16:9, a frame rate of 24 Hz or higher, and at least two channels of CD-quality sound.
HDTV studio equipment is commercially with 1125/60/2:1 scanning and 1920×1035 image format, with about two megapixels per frame – six times the number of pixels of conventional television. The data rate of studio-quality HDTV is about 120 megabytes per second. Commercially available HDTV cameras rival the picture quality of the best motion picture cameras and films.

Except for their higher sampling rates, studio standards for HDTV have a close relationship to studio standards for conventional video, which I will describe in the rest of the book. For details specific to HDTV, consult the book from NHK Labs, SMPTE 274M and 296M.

Advanced Television (ATV) refers to transmission systems designed for the delivery of entertainment to consumers, at quality levels substantially improved over conventional television. ATV transmission systems based on 1125/60/2:1 scanning and MUSE compression have been deployed in Japan. The United States has adopted standards for ATV based on 1920×1080 and 1280×720 image formats. MPEG-2 compression can compress this to about 20 megabits per second, a rate suitable for transmission through a 6 MHz terrestrial VHF/UHF channel.

The compression and digital transmission technology developed for ATV has been adapted for digital transmission of conventional television; this is known as standard-definition television (SDTV). MPEG-2 compression and digital transmission allow a broadcaster to place about four digital channels in the bandwidth occupied by a single analog NTSC signal. Digital television services are already deployed in direct broadcast satellite (DBS) systems and are expected soon in cable television (CATV).
With the advent of HDTV, 16:9 widescreen variants of conventional 525/59.94 and 625/50 component video have been proposed and even standardized. In studio analog systems, widescreen is accomplished by having the active picture represent 16:9 aspect ratio, but keeping all of the other parameters of the video standards. Unless bandwidth is increased by the same \( \frac{4}{3} \) ratio as the increase in aspect ratio, horizontal detail suffers.

In digital video, there are two approaches to achieving 16:9 aspect ratio. The first approach is comparable to the analog approach that I mentioned a moment ago: The sampling rate remains the same as conventional component digital video, and horizontal resolution is reduced by a factor of \( \frac{3}{4} \). In the second approach, the sampling rate is increased from 13.5 MHz to 18 MHz. I consider all of these schemes to adapt conventional video to widescreen be unfortunate: None of them offers an increase in resolution sufficient to achieve the product differentiation that is vital to the success of any new consumer product.
FFMPEG Cheat Sheet

1. Converting Files

   #Using default settings w/ default codec library
   ffmpeg -i /path/to/input /path/to/output

   #Specifying a codec library
   ffmpeg -i /path/to/input -c:v libx264 /path/to/output

   #Average bit rate
   ffmpeg -i /path/to/input -c:v libx264 -b:v 100k -bufsize 50k /path/to/output #bufsize tells the encoder how often
   to calculate average bitrate

   #Setting Min and Max bitrate
   ffmpeg -i /path/to/input -c:v libx264 -b:v 100k -bufsize 50k -minrate 50k -maxrate 50k /path/to/output

   #Using constant rate factor (visually lossless)
   ffmpeg -i /path/to/input -c:v libx264 -crf 18 /path/to/output

   #Restricting the output file size
   Bitrate = output file size in MB * 8192 (convert MB to KB) / Duration in seconds

   From there, two-pass, variable bit rate (VBR) compression can be used to yield an approximate file size:

   ffmpeg -i input -c:v libx264 -preset slow -b:v (calculated bitrate) -strict -2
   output.mp4

2. Transcoding Files for Streaming

   #Creating a webm file
   ffmpeg -i /path/to/input -c:v libvpx -crf 23 -b:v 1M -c:a libvorbis /path/to/output.webm #where -b:v is the max
   bitrate

   #Creating an H.264 file
   ffmpeg -i /path/to/input -c:v libx264 -crf 23 -strict -2 -pix_fmt yuv420p -movflags faststart /path/to/output.mp4

3. Stream Mapping

   #Copy video stream and transcode audio stream
   ffmpeg -i /path/to/input -c:v copy -map 0:1 -c:a aac -strict -2 /path/to/output.mp4

   #Remove audio
   ffmpeg -i /path/to/input -c:v copy -map 0:1 -an -strict -2 /path/to/output.mp4 #audio null

   #Export audio and video to separate files
   ffmpeg -i /path/to/input -map 0:0 /path/to/output.mp4 -map 0:1 /path/to/output.wav #defaults to 16 bit pcm

4. Other things

   #Export sections of a video
   ffmpeg -i /path/to/input -ss 00:00:00 -to 00:00:30 -c:v libx264 /path/to/output #export first 30 seconds of a video
# Deinterlace
ffmpeg -i /path/to/input -c:v libx264 -vf yadif /path/to/output.mp4 # yet another deinterlacing filter

# Scale
ffmpeg -i /path/to/input -c:v libx264 -vf scale=1920:1080 /path/to/output.mp4 # w:h

# Export a single frame as an image
ffmpeg -i input.flv -ss 00:00:14.435 -vf frames 1 out.png

# Export to a dpx sequence
ffmpeg -i input -r 24 -an -pix_fmt rgb24 /output/frames/frame_%06d.dpx

# Animate an image sequence
ffmpeg -i frames/frame_%06d.dpx -r 24 -vcodec prores -profile:v 2 out.mov #0 : ProRes422 (Proxy) 1 : ProRes422 (LT) 2 : ProRes422 (Normal) 3 : ProRes422 (HQ)

# Batch process files
for x in *.avi; do
    ffmpeg -i $x -c:v libx264 -strict -2 ${x%.avi}.mp4;
done

5. Monitoring

# Component RGB waveform monitor
ffplay video -vf format=gbrp,waveform=filter=lowpass:components=7:display=parade

python -m SimpleHTTPServer 8000
Solving Advanced Encoding Problems with FFMPEG

Previous articles in the Code4Lib Journal touch on the capabilities of FFMPEG in great detail, and given these excellent introductions, the purpose of this article is to tackle some of the common problems users might face, dissecting more complicated commands and suggesting their possible uses.

by Josh Romphf

Introduction

FFMPEG has been an important encoding tool for technologists, preservationists, and hobbyists for well over a decade. It is a powerful, multi-purpose open-source library that operates from the command-line, and while a basic knowledge of the command-line environment is useful, FFMPEG can be effectively utilized with little to no programming experience. The source files are easily compiled on all major platforms, and static builds are available for those who prefer to install from an executable[1]. Two articles in Code4Lib Issue 23 touch on the capabilities of FFMPEG in great detail[2], and given these excellent introductions, the purpose of this article is to tackle some of the common problems users might face, dissecting more complicated commands and suggesting their possible uses. To refer to the FFMPEG community as an active one is a vast understatement, and it not surprising how quickly one can fall down an encoding rabbit hole when trying to search forums for solutions to their challenges. This article serves to provide some useful suggestions for transcoding techniques within a specific library or archival context.

Note: the following commands rely heavily on VideoLan’s high performance x264 Library, which is licensed under the GNU GPL and can be compiled with FFMPEG, but is not added to the standard build. The x264 Library is used for encoding H.264/MPEG-4 AVC, and undergirds some of the most high profile streaming operations on the web, including YouTube, Vimeo, and Hulu. When compiled with FFMPEG, it is capable of high quality compression at relatively high speeds.

Understanding Your Project’s Needs

Before embarking on an encoding project, consider whether or not FFMPEG is the right tool to be used. For instance, FFMPEG is excellent for batch processing large collections, but is not ideal for detail oriented digital restoration work. Similarly, it may prove to be overkill for simple transcoding projects, such as on-demand delivery of a single file for a patron. It may be more efficient to use a transcoding program such as Handbrake (which is built on FFMPEG and x264), which has a GUI and significantly less
One of the key steps in helping to make this call is defining the desired output. While this may sound obvious, it is important to tailor your tools to the task at hand: what is the method of delivery? Are you producing preservation quality files? Are you streaming? What type of player will be used? Does file size matter? If so, is there an optimal file size? For those who opt for FFmpeg, what follows are some scenarios accompanied by possible solutions.

Preservation and Access Basics: Input, Output, Preservation, and Streaming

Arguably the most common use of H.264/MPEG-4 AVC encoding is for streaming and access solutions, but it can also be used to transcode lossless preservation masters and high quality mezzanine files[3]. When the ultimate goal is preservation, the x264 codec supports lossless encoding, yet it should be noted that a number of streaming services, with the exception of YouTube, are capable of decoding lossless H.264. In addition, a new container format may be desired for posterity; one such format is Matroska (MKV), an open source container designed with the intention of becoming the defacto standard for multimedia containers. It has become known for its support of “all known video and audio compression formats” and has been adopted by a number of archival projects [5].

As libraries and archives begin to digitize their moving image collections and provide online access to them, the quality of streamed files becomes more and more important. Depending on whether or not the file size of the output is an issue, there are a few paths one can take with x264.

1) Transcoding for Quality:

```bash
ffmpeg -i input -c:v libx264 -preset slow -crf 18 -vf yadif -strict -2 output.mp4
```

When file size is not an issue, this is a quick command that will produce a visually lossless H.264 encoding with an mp4 container. C:v is the video codec of choice, preset is the compression preset (in this case slow for higher quality compression), and CRF is the Constant Rate Factor, which preserves an overall level of quality throughout the file by adjusting each frame’s bitrate based on the given quality level. Consequently, the higher the CRF, the lower the overall quality level. The video filter flag (-vf) is used to call FFmpeg’s pre-bundled video filters, while yadif (Yet Another Deinterlacing Filter) deinterlaces an interlaced input, as progressive video is not
only easier to compress and most current computer monitors and televisions are progressive scan. Finally, when encoding with H.264/MPEG-4 AVC, the audio format used is AAC (Advanced Audio Coding); in order to enable FFmpeg’s experimental, native AAC encoder, -strict -2 needs to be added to the command. An external library such as libfaac [4] can also be used, and -strict -2 can be omitted.

A preservation quality master can be output using the same principles, setting the –crf to 0, the preset to veryslow, and the output container to .mkv:

```
ffmpeg -i input -c:v libx264 -preset veryslow -crf 0 -vf yadif -strict -2 output.mkv
```

A preservation quality master can be output using the same principles, setting the –crf to 0, the preset to `veryslow`, and the output container to .mkv:

```
ffmpeg -i input -c:v libx264 -preset veryslow -crf 0 -vf yadif -strict -2 output.mkv
```

**2) Streaming for File Size:**

In a scenario where server space or bandwidth is limited, a desired file size may be required. To calculate the output file’s bitrate based on a target size, the following equation can be used:

\[
\text{Bitrate} = \frac{\text{output file size in MB} \times 8192 \text{ (convert MB to KB)}}{\text{Duration in seconds}}
\]

From there, two-pass, variable bit rate (VBR) compression can be used to yield an approximate file size:

```
```

In the above example, the first command is passed to a null device, the `&&` operator then executes the second command, which yields the mp4 file.

**3) Streaming within a Browser**

If a streaming service such as YouTube or Vimeo is not being used, an option is to add –movflags faststart to the command, which shifts the `moov atom` (the data unit that defines the file’s timescale, duration, and information for each track)[7] to the beginning
of the mp4 container. This allows for playback and seeking to be started before the video has fully downloaded – a feature that is particularly useful when streaming content. Although some delivery mechanisms do not require the position of the moov atom to be located at the beginning of the container, it is required by both Flash video (.flv) and HTML5 video, which are two of the most common web-based video options available. Most other web-based players will be able to function regardless of the moov atom’s position.

**Concatenating Files**

Another essential use of FFmpeg lies in its ability to concatenate (join) multiple video files. The need to concatenate may arise when content is ingested in the form of multiple files, when a patron requests a sequence of clips, or when an image sequence needs to be animated. The use of the **concat** filter is outlined in *Automated Processing of Massive Audio/Video Content Using FFmpeg*, and is suitable when concatenating at the file level, yet it can be lossy and uncooperative with some containers, especially mp4.

The most straightforward means of concatenating files produced with the same codec is by using the **concat demuxer**, which calls on an external .txt file for orders on which files to join and in what order. The text file should appear as follows:

```plaintext
#List of Files to Join (Comment)
file '/usr/path/to/file1'
file '/usr/path/to/file2'
```

The command itself is quite simple, and can be used as a stream copy for efficiency. **–c copy** moves all of the streams over to a new container without decoding and encoding the input:

```bash
ffmpeg -f concat -i /path/to/list/concatList.txt -c copy out.mp4
```

The Unix **cat** command can also be used for concatenating files, and is particularly helpful when pulling .VOB files together from a DVD. A Handbrake alternative, this method is ideal for transcoding to containers not offered by the program, such as .mov and .mp4:

```bash
cat /Volumes/YOURDVTITLE/VIDEO_TS/VTS_01_[1234].VOB | ffmpeg -y -i - -c:v libx264 -preset slow -crf 18 -strict -2 output.mp4
```
The .VOB files are concatenated in order, where [1234] corresponds to the filename; for instance, to concatenate VTS_01_1.VOB and VTS_01_2.VOB, the command is cat /Volumes/YOURDVDTITLE/VIDEO_TS/VTS_01_[12]. This is then piped to FFmpeg, setting the .VOB files as the input. The user can then customize the remainder of the command in any way.

Another means of concatenating files with the same codec is the concat protocol in ffmpeg, which works at the file level as opposed to the stream level. Unlike the Unix cat command, the concat protocol uses pipes “|” to delineate input files:

```
ffmpeg –i “concat:input1|input2|input3” -c copy out.mp4
```

Be aware that some codec and container combinations may not work properly. For instance, H.264 encoded files cannot be losslessly concatenated in ffmpeg. There is, however, a workaround for this which is achieved by first copying them to mpeg2 transport streams (.ts). Using –c copy will yield an error that can be solved by the use of a bitstream filter (–bsf), which performs modifications at the bitstream level, foregoing any decoding and producing the same results. The transport streams can then be written to a temporary directory, concatenated, and removed:

```
ffmpeg -i input1.mp4 -bsf:v h264_mp4toannexb -f mpegts -c:v mpeg2video -b:v 2500k /path/to/temp/directory/intermediate1.ts &&
ffmpeg -i input2.mp4 -bsf:v h264_mp4toannexb -f mpegts -c:v mpeg2video -b:v 2500k /path/to/temp/directory/intermediate2.ts &&
ffmpeg -probesize 100M -analyzeduration 250M -i "concat:/path/to/temp/directory/intermediate1.ts|/path/to/temp/directory/intermediate2.ts" -c:v libx264 -crf 18 -preset slow -strict -2 -movflags faststart output.mp4 &&
rm -rf /path/to/temp/directory
```

The use of –probesize and –analyzeduration help FFmpeg to recognize the audio and video stream parameters of a file, and while they are by no means necessary, their use can help reduce input processing errors, especially when moving across codecs.

Finally, libx264’s powerful animation capabilities can prove incredibly beneficial for preservation projects. For instance, film scans that produce individual frames (for instance, Cineon, JPEG, DPX or TIFF sequences) can be recombined and compressed for access. Another use could be for pulling JPEG2000 images from a DCP (Digital
Cinema Package) and recombining them for viewing. To join image files, use the
command:

ffmpeg -r 24/1 -i filename%04d.jpg -c:v libx264 -r 24 -pix_fmt yuv420p
output.mp4

Where –r 24/1 refers to the duration of each image, in this case 1/24th of a
second, filename%04d.jpg finds all jpeg files with filename followed by a 4 number
sequence (i.e. filename0001.jpg), and –r 24 refers to the framerate. –pix_fmt is also
important, as the pixel format must be set; in this case, the YUV color space with 4:2:0
chroma subsampling is used to maximize playback compatibility. FFMPEG also supports
globbing for image sequences, where –i is supplemented with –pattern_type glob –i,
whereby wildcards (i.e. *.jpg) can be used.

Scripting and Batch Processing
An even greater level of automation can be achieved by scripting FFMPEG and Unix
commands, freeing up time for other tasks. All of the above commands can be scripted
and combined using a number of scripting languages. For example, a script can be
written to batch transcode a preservation MKV and streaming optimized MP4 from any
type of source file. Two of the more popular methods are to use a conventional shell
script or a Python script with subprocess calls. For instance, a simple shell script with a
for loop can be written as such:
1
2
3
4
5
6

#!/bin/bash
#Convert all MOV files to MP4 in a directory
for x in /path/to/files/*.mov;
#Insert Other Commands, such as moving files or creating a lower quality copy
done

The file is then saved with the .sh extension and can be opened in a bash shell. This is
by far the simplest method of the two, as it is language that FFMPEG users are already
familiar with. Python can be used for more complex scripting, such as reading files
through FFMPEG, processing them with Python, and piping them back out to FFMPEG
for an output. With the inclusion of the OS and Subprocess modules, one can execute
shell commands with all the functionality of a shell script. Subprocess calls can also be
couched within a for loop or stored in variables for more efficient coding. A simple
Python script with multiple commands would be written as follows:
1
2
3
4

#!/usr/bin/env python
import os
import subprocess


# Script to compile image sequence and add audio

# If ffmpeg is not in $PATH, use os.chdir('path/to/ffmpeg/binary/directory/')

# Compile image sequence

subprocess.call([  
    'ffmpeg',  
    '-r', '24/1',  
    '-i', 'filename%04d.jpg',  
    '-r', '24',  
    '-c:v', 'libx264',  
    '-preset', 'slow',  
    '-crf', '18',  
    '-pix_fmt', 'yuv420p'  
    '-s', '1920x1080',  
    ' output.mp4'])

# Add audio track to output video

subprocess.call([  
    'ffmpeg',  
    '-i', 'output.mp4',  
    '-i', 'audioTrack.wav',  
    '-c:v', 'libx264',  
    '-c:a', 'aac',  
    '-strict', '-2',  
    '-b:a', '-192k',  
    'composite.mp4'])

The script is saved with the extension .py, and can be run in the same manner as any other python script:

```
python /path/to/script/videoscript.py
```

## Conclusion

One can imagine the possibilities of FFMPEG, especially when combined with a powerful, yet accessible programming language like Python. This article merely scratches the surface of what can be done. An indispensable tool for libraries and archives – especially those with limited technical staffing – FFMPEG can be utilized to solve nearly all the needs of a digital video project. Again, the learning curve can be daunting, and the end result is largely what you make of it. It is hoped that this article served as a gentle introduction to both video encoding principals, as well as a companion for some of FFMPEG and x264’s idiosyncrasies. Please consult the references section for more specific, in-depth commands and explanations.
Notes


[2] Automated Processing of Massive Audio/Video Content Using FFMPEG (KIA Siang Hock and LI Lingxia, January 2014) contains several important, common commands and provides explanations, while Unix Commands and Batch Processing for the Reluctant Librarian or Archivist (Anthony Cocciolo, January 2014) explains how to batch transcode files using FFMPEG commands with a bash for loop.

[3] Mezzanine files are intended to be used as your input when transcoding access files to lessen the risk of damaging the master.


[5] The Matroska (http://matroska.org/technical/specs/index.html) container format has gained a great deal of momentum over the last few years, and has been adopted as the preservation container within the infrastructures of open source digital asset management systems such as Archivematica (https://www.archivematica.org/wiki/Main_Page) and Islandora (https://discoverygarden.ca/).

[6] Visit http://trac.ffmpeg.org/wiki/How%20to%20quickly%20compile%20libx264 for instructions on how to easily compile x264 with FFMPEG. As a side note for Mac users, the x264 Library (and a number of other dependencies) are included in the Homebrew installation.


References:

Several FFMPEG tutorials are available through the official bug tracker and wiki: https://trac.ffmpeg.org/.

VideoLan also maintains excellent documentation on the x264 library: http://www.videolan.org/developers/x264.html.

As always, Stack Overflow, albeit somewhat difficult to navigate, offers a great deal of tips and tricks http://stackoverflow.com/questions/tagged/ffmpeg. After Dawn http://www.afterdawn.com/ also maintains a forum for video encoding related questions and issues.
CHAPTER 1

Introduction

This chapter gives you a background on the creation of the HTML5 media elements. The history of their introduction explains some of the design decisions that were taken, in particular why there is not a single baseline codec. If you are only interested in learning the technical details of the media elements, you can skip this chapter.

The introduction of the media elements into HTML5 is an interesting story. Never before have the needs around audio and video in web pages been analyzed in so much depth and been discussed among this many stakeholders. Never before has it led to a uniform implementation in all major web browsers.

1.1 A Bit of History

While it seems to have taken an eternity for all the individuals involved in HTML and multimedia to achieve the current state of the specifications and the implementations in the web browsers, to the person on the street, it has been a rather surprising and fast innovation.

From the first mention of the possibility of a <video> element in HTML5 in about 2005, to the first trial implementation in February 2007, to the first browser rolling it out in a nightly build in November 2007, and to Microsoft’s Internet Explorer joining the party late in a developer preview in March 2010, it has still been barely five years.

In contrast, other efforts to introduce media functionality natively into HTML without the use of plug-ins in the <embed> or <object> elements have been less successful. HTML+Time was proposed in 1998 by Microsoft and implemented into IE 5, IE 5.5 and IE6, but was never supported by any other browser vendor. SMIL (pronounced “smile”), the Synchronized Multimedia Integration Language, has been developed since 1997 to enable authoring of interactive audiovisual presentations, but was never natively supported in any browser other than the part that matched the HTML+Time specification.

This rapid development was possible only because of the dozens of years of experience with media plug-ins and other media frameworks on the Web, including QuickTime, Microsoft Windows Media, RealNetworks RealMedia, Xiph Ogg, ISO/MPEG specifications, and, more recently, Adobe Media and Microsoft Silverlight. The successes of YouTube and similar hosting sites have vastly shaped the user requirements. Many more technologies, standards, and content sites also had an influence, but it would take too long to list them all here.

All this combined experience led eventually to the first proposal to introduce a <video> element into HTML5. This is the first time that all involved stakeholders, in particular all browser vendors, actually committed to a native implementation of media support in their browsers.

Before the introduction of the <video> and <audio> elements, a web developer could include video and audio in web pages only through <object> and <embed> elements, which required browser plug-ins be installed on user machines. Initially, these plug-ins simply launched a media player that was installed on the user’s system to play back video. Later, they were able to display inside web pages, although often users were taken into a pop-up. This was the case for all of the popular plug-ins, such as RealMedia, QuickTime, and Windows Media. With the release of Flash Player 6 in 2002, Macromedia introduced video support into its browser plug-in. It relied on the Sorenson Spark codec, which was also used by
QuickTime at that time. Most publishers already published their content in RealMedia, QuickTime and Windows Media format to cover as much of the market as possible, so uptake of Flash for video was fairly small at first.

However, Macromedia improved its tools and formats over the next few years with ActionScript. With Flash Player 8 in 2005, it introduced On2’s VP6 advanced video codec, alpha transparency in video, a standalone encoder and advanced video importer, cue point support in FLV files, an advanced video playback component, and an interactive mobile device emulator. All of this made it a very compelling development environment for online media.

In the meantime, through its animation and interactive capabilities, Flash had become the major plug-in for providing rich Internet applications which led to a situation where many users had it installed on their system. It started becoming the solution to publishing video online without having to encode it in three different formats. It was therefore not surprising when Google Videos launched on January 25, 2005 using Macromedia Flash. YouTube launched only a few months later, in May 2005, also using Macromedia Flash.

On December 3, 2005, Macromedia was bought by Adobe and Flash was henceforth known as Adobe Flash. As Adobe continued to introduce and improve Flash and the authoring tools around it, video publishing sites around the world started following the Google and YouTube move and also published their videos in the Adobe Flash format. With the introduction of Flash Player 9, Update 3, Adobe launched support in August 2007 for the MPEG family of codecs into Flash, in particular the advanced H.264 codec, which began a gradual move away from the FLV format to the MP4 format.

In the meantime, discussion of introducing a <video> element into HTML, which had started in 2005, continued. By 2007, people had to use gigantic <embed> statements to make Adobe Flash work well in HTML. There was a need to simplify the use of video and fully integrate it into the web browser.

The first demonstration of <video> implemented in a browser was done by Opera. On February 28, 2007, Opera announced1 to the WHATWG (Web Hypertext Applications Technology Working Group2) an experimental build of a <video> element, which Opera Chief Technology Officer Håkon Wium Lie described as a first step towards making “video a first-class citizen of the web.”3 The specification was inspired by the <img> element and was built similarly to an interface created earlier for an Audio() JavaScript API.

Initially, there was much discussion about the need for a separate <video> element—why wouldn’t the <embed> element be sufficient, why not use SMIL, why not reanimate HTML+Time? Eventually it dawned on people that, unless media was as simple to use as <img> and as integrated into all layers of web applications, including the DOM, CSS, and JavaScript, <video> and <audio> would be hampered from making further progress on the web beyond what was possible with plug-ins. This, of course, includes the need for all browsers to support the specifications in an interoperable way. Thus, the need for standardization of the <video> element was born.

1.2 A Common Format?

An early and ongoing debate around the HTML5 media elements is that of a baseline encoding format, also called a “baseline codec”. A baseline codec is a video and audio encoding format that is supported and implemented by all browser vendors and thus a web developer can rely on it to work in all browsers.

The question of a baseline codec actually goes beyond just the question of codecs. Codec data is only the compressed audio or video data by itself. It never stands on its own, but is delivered in a “container format”, which encapsulates the encoded audio and video samples in a structure to allow

---

1 See http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2007-February/009702.html
2 See http://www.whatwg.org/
later decoding. You can think of it as analogous to packaging data packets for delivery over a computer network, where the protocol headers provide the encapsulation.

Many different encapsulation formats exist, including QuickTime’s MOV, MPEG’s MP4, Microsoft’s WMV, Adobe’s FLV, the Matroska MKV container (having been the basis for the WebM format), AVI and Xiph’s Ogg container. These are just a small number of examples. Each of these containers can in theory support encapsulation of any codec data sequence (except for some container formats not mentioned here that cannot deal with variable bitrate codecs).

Also, many different audio and video codecs exist. Examples of audio codecs are: MPEG-1 Audio Level 3 (better known as MP3), MPEG-2 and MPEG-4 AAC (Advanced Audio Coding), uncompressed WAV, Vorbis, FLAC and Speex. Examples of video codecs are: MPEG-4 AVC/H.264, VC-1, MPEG-2, H.263, VP8, Dirac and Theora.

Even though in theory every codec can be encapsulated into every container, only certain codecs are typically found in certain containers. WebM, for example, has been defined to only contain VP8 and Vorbis. Ogg typically contains Theora, Vorbis, Speex, or FLAC, and there are defined mappings for VP8 and Dirac, though not many such files exist. MP4 typically contains MP3, AAC, and H.264.

For a specification like HTML5, it is important to have interoperability, so the definition of a baseline codec is important. The debate about a baseline codec actually started on the day that Opera released its experimental build and hasn’t stopped since.

A few weeks after the initial proposal of the `<video>` element, Opera CTO Wium Lie stated in a talk given at Google:

“I believe very strongly, that we need to agree on some kind of baseline video format if [the video element] is going to succeed. [...] We want a freely implementable open standard to hold the content we put out. That’s why we developed the PNG image format. [...] PNG [...] came late to the party. Therefore I think it’s important that from the beginning we think about this.”

Wium Lie further stated requirements for the video element as follows:

“It’s important that the video format we choose can be supported by a wide range of devices and that it’s royalty-free (RF). RF is a well-establish[ed] principle for W3C standards. The Ogg Theora format is a promising candidate which has been chosen by Wikipedia.”

The World Wide Web Consortium (W3C) is the standards body that publishes HTML. It seeks to issue only recommendations that can be implemented on a royalty-free (RF) basis.

The “Ogg Theora” format proposed as a candidate by Wium Lie is actually the video codec Theora and the audio codec Vorbis in an Ogg container developed by the Xiph.org Foundation as open source. Theora is a derivative of a video codec developed earlier by On2 Technologies under the name VP3 and released as open source in September 2001. With the release of the code, On2 also essentially provided a royalty-free license to their patents that relate to the VP3 source code and its derivatives. After VP3 was published and turned into Theora, Ogg Theora/Vorbis became the first unencumbered video codec format. Google, which acquired On2 in 2010, confirmed Theora’s royalty-free nature.

---

4 See video of Håkon Wium Lie’s Google talk, http://video.google.com/videoplay?docid=5545573096553082541&ei=L6hSaz0JpbA2AKh40yPDg8hl=un
6 See W3C RF requirements at http://www.w3.org/Consortium/Patent-Policy-20030520.html#sec-Licensing
7 See Xiph.Org’s Website on Theora, http://theora.org/
Note that although the video codec format should correctly be called “Ogg Theora/Vorbis”, in common terminology you will only read “Ogg Theora”.

On the audio side of things, Ogg Vorbis is a promising candidate for a baseline format. Vorbis is an open-source audio codec developed and published by Xiph.Org since about 2000. Vorbis is also well regarded as having superior encoding quality compared with MP3 and on par with AAC. Vorbis was developed with a clear intention of only using techniques that were long out of patent protection. Vorbis has been in use by commercial applications for a decade now, including Microsoft software and many games.

An alternative choice for a royalty-free modern video codec that Wium Lie could have suggested is the BBC-developed Dirac codec. It is based on a more modern compression technology, namely wavelets. While Dirac’s compression quality is good, it doesn’t, however, quite yet expose the same compression efficiency as Theora for typical web video requirements.

For all these reasons, Ogg Theora and Ogg Vorbis were initially written into the HTML5 specification as baseline codecs for video and audio, respectively, at the beginning of 2007:

“User agents should support Ogg Theora video and Ogg Vorbis audio, as well as the Ogg container format.”

However, by December 2007, it was clear to the editor of the HTML5 draft, Ian Hickson, that not all browser vendors were going to implement Ogg Theora and Ogg Vorbis support. Apple in particular had released the first browser with HTML5 video support with Safari 3.1 and had chosen to support only H.264, criticizing Theora for inferior quality, for lack of support on mobile devices, and a perceived increased infringement threat of as-yet unknown patents (also called the “submarine patent” threat).

Nokia and Microsoft confirmed their positions for a similar choice. H.264 has been approved as a standard jointly by the International Telecommunications Union (ITU) and the International Standards Organization (ISO/IEC), but its use requires payment of royalties, making it unacceptable as a royalty-free baseline codec for HTML5. The announcement of MPEG LA on August 26, 2010 that H.264 encoded Internet video that is free to end users will never be charged for royalties is not sufficient, since all other royalties, in particular royalties for commercial use and for hardware products, remain in place.

In December 2007, Ian Hickson replaced the should-requirement for Ogg Theora with the following:

“It would be helpful for interoperability if all browsers could support the same codecs. However, there are no known codecs that satisfy all the current players: we need a codec that is known to not require per-unit or per-distributor licensing, that is compatible with the open source development model, that is of sufficient quality as to be usable, and that is not an additional submarine patent risk for large companies. This is an ongoing issue and this section will be updated once more information is available.”

---

11 See Dirac Website, http://diracvideo.org/
14 See as an example this story in Apple Insider http://www.appleinsider.com/articles/09/07/06/ogg_theora_h_264_and_the_html_5_browser_squabble.html
15 See Nokia submission to a W3C workshop on video for the Web at http://www.w3.org/2007/08/video/positions/Nokia.pdf
16 See W3C HTML Working Group Issue tracker, Issue #7 at http://www.w3.org/html/tracker/issues/7
H.264 has indeed several advantages over Theora. First, it provides a slightly better overall encoding quality. Second, the de-facto standard for video publication on the Web had been set by YouTube, which used Adobe Flash with MP4 H.264/AAC support. Choosing the same codec as Adobe Flash will provide a simple migration path to the HTML5 video element since no additional transcoding would be necessary. Third, there are existing hardware implementations of H.264 for mobile devices, used in particular by Apple’s iPod, iPhone, and iPad, which support this codec out of the box.

However, it is not inconceivable that the market will catch up over the next few years with software support and hardware implementations for Ogg Theora, increasingly enabling professional use of these codecs. In fact, in April 2010, Google funded a free, optimized implementation of Theora for the ARM processor, which runs Google’s Android devices. Theora is praised to be less complex and therefore requiring less dedicated hardware support than H.264, making it particularly useful on mobile devices.

This was the situation until May 19, 2010, when Google announced the launch of the WebM project, which proposes another opportunity to overcome the concerns Apple, Nokia and Microsoft have voiced with Theora. WebM is a new open-source and royalty-free video file format, which includes the VP8 video codec, a codec Google had acquired as part of it acquisition of On2 Technologies, finalized in February 2010. The VP8 video codec, together with the Vorbis audio codec, is placed inside a container format derived from the Matroska file format to make up the full video encoding format called WebM.

Google released WebM with an obvious intention of solving the stalemate around a baseline video codec in HTML5. To that end, Google released WebM and VP8 under a BSD style open-source license, which allows anyone to make use of the code freely. They also grant a worldwide, non-exclusive, no-charge, royalty-free patent license to the users of the codec to encourage adoption. They collaborated with Opera, Mozilla, and Adobe and many others to achieve support for WebM, such as an implementation of WebM in the Opera, Google Chrome, and Firefox browsers, and also move forward with commercial encoding tools and hardware implementations. On October 15, 2010, Texas Instruments was the first hardware vendor to demonstrate VP8 on its new TI OMAP™ 4 processor.

Microsoft’s reaction to the release of WebM was rather positive, saying that it would “support VP8 when the user has installed a VP8 codec on Windows”. Apple basically refrained from making any official statement. Supposedly, Steve Jobs replied to the question "What did you make of the recent VP8 announcement?" in an e-mail with a pointer to a blog post by an X.264 developer. The blog post hosts an initial, unfavorable analysis of VP8’s quality and patent status. Note that X.264 is an open-source implementation of an H.264 decoder, the developer is not a patent attorney, and the analysis was done on a very early version of the open codebase.

As the situation stands, small technology providers or nonprofits are finding it hard to support a non-royalty-free codec. Mozilla and Opera have stated that they will not be able to support MP4 H.264/AAC since the required annual royalties are excessive, not just for themselves, but also for their

---

23 See http://www.matroska.org/
24 See http://webmproject.blogspot.com/2010/05/introducing-webm-open-web-media-project.html
25 See http://www.webmproject.org/license/additional/
26 See http://webmproject.blogspot.com/2010/05/introducing-webm-open-web-media-project.html
27 See http://e2e.ti.com/videos/m/application_specific/240443.aspx
29 See http://x264dev.multimedia.cx/?p=377
downstream users and, more important, because the use of patent encumbered technology is against the ideals of an open Web.\textsuperscript{30} They have both implemented and released exclusive support for Ogg Theora and WebM in their browsers. Apple’s Safari still supports only MP4 H.264/AAC. Google Chrome supports all these three codecs. Table 1–1 has a summary of the current implementation situation.

\textit{Table 1–1. Introduction of HTML5 video support into main browsers}

<table>
<thead>
<tr>
<th>Browser</th>
<th>Nightly</th>
<th>Release</th>
<th>Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safari</td>
<td>November 2007</td>
<td>March 2008 (Safari 3.1)</td>
<td>MP4 H.264/AAC</td>
</tr>
<tr>
<td>Firefox</td>
<td>July 2008</td>
<td>June 2009 (Firefox 3.5)</td>
<td>Ogg Theora, WebM</td>
</tr>
<tr>
<td>Chrome</td>
<td>September 2008</td>
<td>May 2009 (Chrome 3)</td>
<td>Ogg Theora, MP4 H.264/AAC, WebM</td>
</tr>
<tr>
<td>Opera</td>
<td>February 2007 / July 2008</td>
<td>January 2010 (Opera 10.50)</td>
<td>Ogg Theora, WebM</td>
</tr>
<tr>
<td>IE</td>
<td>March 2010 (IE9 dev build)</td>
<td>September 2010 (IE9 beta)</td>
<td>MP4 H.264/AAC</td>
</tr>
</tbody>
</table>

In the publisher domain, things look a little different because Google has managed to encourage several of the larger publishers to join in with WebM trials. Brightcove, Ooyala and YouTube all have trials running with WebM content. Generally, though, the larger publishers and the technology providers that can hand on the royalty payments to their customers are able to support MP4 H.264/AAC. The others can offer only Ogg Theora or WebM (see Table 1–2).

\textit{Table 1–2. HTML5 video support into some major video publishing sites (social and commercial)}

<table>
<thead>
<tr>
<th>Site / Vendor</th>
<th>Announcement</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>Basically since 2004, stronger push since 2009</td>
<td>Ogg Theora, WebM</td>
</tr>
<tr>
<td>Dailymotion</td>
<td>May 27, 2009</td>
<td>Ogg Theora, WebM</td>
</tr>
<tr>
<td>YouTube</td>
<td>January 20, 2010</td>
<td>MP4 H.264/AAC, WebM</td>
</tr>
<tr>
<td>Vimeo</td>
<td>January 21, 2010</td>
<td>MP4 H.264/AAC, WebM</td>
</tr>
<tr>
<td>Kaltura</td>
<td>March 18, 2010</td>
<td>Ogg Theora, WebM, MP4 H.264/AAC</td>
</tr>
<tr>
<td>Ooyala</td>
<td>March 25,2010</td>
<td>MP4 H.264/AAC, WebM</td>
</tr>
<tr>
<td>Brightcove</td>
<td>March 28, 2010</td>
<td>MP4 H.264/AAC, WebM</td>
</tr>
</tbody>
</table>

\textsuperscript{30} See \url{http://shaver.off.net/diary/2010/01/23/html5-video-and-codecs/}
An interesting move is the announcement of VP8 support by Adobe.\textsuperscript{31} When Adobe releases support for WebM, this will imply that video publishers that choose to publish their videos in the WebM format will be able to use the Adobe Flash player as a fallback solution in browsers that do not support the WebM format, which includes legacy browsers and HTML5 browsers with exclusive MP4 H.264/AAC support. This is a very clever move by Adobe and will allow smaller content publishers to stay away from H.264 royalties without losing a large number of their audience and without having to make the content available in multiple formats.

1.3 Summary

In this chapter we have looked back at the history of introducing audio and video on the Web and how that led to the introduction of \texttt{<video>} and \texttt{<audio>} elements into HTML5. We also described the discussions and status around finding a single video codec that every browser vendor could support as a baseline format.

As the situation currently stands, any video publisher that wants to create web pages with videos that are expected to universally work with any browser will be required to publish video in at least two formats: in MP4 H.264/AAC and in either Ogg Theora or WebM. Currently, Ogg Theora support and tools are still further developed than WebM tools, but WebM tools are improving rapidly. If you need to set up a site from scratch, your best choice is probably MP4 H.264/AAC and WebM.

\textsuperscript{31} See http://blogs.adobe.com/flashplatform/2010/05/adobe_support_for_vp8.html
ABSTRACT

"Computer vision" refers to a broad class of algorithms that allow computers to make intelligent assertions about digital images and video. Historically, the creation of computer vision systems has been regarded as the exclusive domain of expert researchers and engineers in the fields of signal processing and artificial intelligence. Likewise, the scope of application development for computer vision technologies, perhaps constrained by conventional structures for research funding, has generally been limited to military and law-enforcement purposes. Recently, however, improvements in software development tools for student programmers and interactive-media artists — in combination with the rapid growth of open-source code-sharing communities, predictable increases in PC processor speeds, and plummeting costs of digital video hardware — have made widespread artistic experimentation with computer vision techniques a reality. The result is a proliferation of new practitioners with an abundance of new application ideas, and the incorporation of computer vision techniques into the design vocabularies of novel artworks, games, home automation systems, and other areas. This article attempts to demystify computer vision for novice programmers, through a survey of new applications in the arts, system design considerations, and contemporary tools.

KEYWORDS

Computer vision, machine vision, interactive art, artistic applications, authoring tools, education.

INTRODUCTION

A well-known anecdote relates how, sometime in 1966, the legendary Artificial Intelligence pioneer Marvin Minsky directed an undergraduate student to solve "the problem of computer vision" as a summer project [Bechtel 2003]. This anecdote is often resuscitated to illustrate how egregiously the difficulty of computational vision has been underestimated. Indeed, nearly forty years later, the discipline continues to confront numerous unsolved (and perhaps unsolvable) challenges, particularly with respect to high-level "image understanding" issues such as pattern recognition and feature recognition. Nevertheless, the intervening decades of research have yielded a great wealth of well-understood, low-level techniques that are able, under controlled circumstances, to extract meaningful information from a camera scene. These techniques are indeed elementary enough to be implemented by novice programmers at the undergraduate or even high-school level.

This paper attempts to demystify computer vision for novice programmers, emphasizing the use of vision-based detection and tracking techniques in the interactive media arts. The first section of this article introduces some of the ways in which computer vision has found artistic applications outside of industrial and military research. Section II, Elementary Computer Vision Techniques, presents an overview of several basic but widely-used vision algorithms, with example code included in appendices at the end of the article. Although it is easy to suppose that sophisticated software is all one needs to create a computer vision system, Section III, Computer Vision in the Physical World, makes the case that a well-prepared physical environment can dramatically improve algorithmic performance and robustness. The remaining sections present a brief survey of several artist-friendly new computer vision toolkits, and an example of a student project, developed by novice programmers in a workshop structured around the considerations presented in this article.
I. COMPUTER VISION IN INTERACTIVE ART

The first interactive artwork to incorporate computer vision was, interestingly enough, also one of the first interactive artworks. Myron Krueger's legendary *Videoplace*, developed between 1969 and 1975, was motivated by his deeply felt belief that the entire human body ought to have a role in our interactions with computers. In the *Videoplace* installation, a participant stands in front of a backlit wall and faces a video projection screen. The participant's silhouette is then digitized, and its posture, shape and gestural movements analyzed. In response, *Videoplace* synthesizes graphics such as small "critters" which climb up the participant's projected silhouette, or colored loops drawn between the participant's fingers. Krueger also allowed participants to paint lines with their fingers, and, indeed, entire shapes with their bodies; eventually, *Videoplace* offered over 50 different compositions and interactions.

*Videoplace* was notable for many "firsts" in the history of human-computer interaction. Some of its interaction modules, for example the ones shown above, allowed two participants in mutually remote locations to participate in the same shared video space, connected across the network — an implementation of the first multi-person virtual reality, or, as Krueger termed it, an "artificial reality." *Videoplace*, it should be noted, was developed before Douglas Englebart's mouse became the ubiquitous desktop device it is today, and was (in part) created to demonstrate interface alternatives to the keyboard terminals which dominated computing so completely in the early 1970's. Remarkably enough, the original *Videoplace* system is still operational as of this writing.
Figure 3. Vocalist Jaap Blonk performing the *Messa di Voce* interactive software by Golan Levin and Zachary Lieberman (2003).

*Messa di Voce*, created by this article's author in collaboration with Zachary Lieberman, uses whole-body vision-based interactions similar to Krueger's, but combines them with speech analysis and situates them within a kind of projection-based augmented reality. In this audiovisual performance, the speech, shouts and songs produced by two abstract vocalists are visualized and augmented in real-time by synthetic graphics. To accomplish this, a computer uses a set of vision algorithms to track the locations of the performers' heads; this computer also analyzes the audio signals coming from the performers' microphones. In response, the system displays various kinds of visualizations on a projection screen located just behind the performers; these visualizations are synthesized in ways which are tightly coupled to the sounds being spoken and sung. With the help of the head-tracking system, moreover, these visualizations are projected such that they appear to emerge directly from the performers' mouths [Levin and Lieberman].
Rafael Lozano-Hemmer's installation *Standards and Double Standards* (2004) incorporates full-body input in a less direct, more metaphorical context. This work consists of fifty leather belts, suspended at waist height from robotic servo-motors mounted on the ceiling of the exhibition room. Controlled by a computer vision-based tracking system, the belts rotate automatically to follow the public, turning their buckles slowly to face passers-by. Lozano-Hemmer's piece "turns a condition of pure surveillance into an 'absent crowd' using a fetish of paternal authority: the belt" [Lozano-Hemmer].
The theme of surveillance plays a foreground role in David Rokeby's *Sorting Daemon* (2003). Motivated by the artist's concerns about the increasing use of automated systems for profiling people as part of the "war on terrorism", this site-specific installation works toward the automatic construction of a diagnostic portrait of its social (and racial) environment. Rokeby writes: "The system looks out onto the street, panning, tilting and zooming, looking for moving things that might be people. When it finds what it thinks might be a person, it removes the person's image from the background. The extracted person is then divided up according to areas of similar colour. The resulting swatches of colour are then organized [by hue, saturation and size] within the arbitrary context of the composite image" projected onsite at the installation's host location [Rokeby].
Another project themed around issues of surveillance is *Suicide Box* by the Bureau of Inverse Technology (Natalie Jeremijenko and Kate Rich). Presented as a device for measuring the hypothetical "Despondency Index" of a given locale, the Suicide Box nevertheless records very real data regarding suicide jumpers from the Golden Gate Bridge. According to the artists, "The *Suicide Box* is a motion-detection video system, positioned in range of the Golden Gate Bridge, San Francisco in 1996. It watched the bridge constantly and when it recognised vertical motion, captured it to a video record. The resulting footage displays as a continuous stream the trickle of people who jump off the bridge. The Golden Gate Bridge is the premiere suicide destination in the United States; a 100 day initial deployment period of the *Suicide Box* recorded 17 suicides. During the same time period the Port Authority counted only 13." [Bureau of Inverse Technology]. Elsewhere, Jeremijenko has explained that "the idea was to track a tragic social
phenomenon which was not being counted — that is, doesn't count” [Shachtman]. The Suicide Box has met with considerable controversy, ranging from ethical questions about recording the suicides, to others disbelieving that the recordings could be real. Jeremijenko, whose aim is to address the hidden politics of technology, has pointed out that such attitudes express a recurrent theme — "the inherent suspicion of artists working with material evidence" — evidence obtained, in this case, with the help of machine-vision based surveillance.

Figures 7, 8. Stills from Cheese, an installation by Christian Möller (2003).

Considerably less macabre is Christian Möller's clever Cheese installation (2003), which the artist developed in collaboration with the Machine Perception Laboratories of the University of California, San Diego. Motivated, perhaps, by the culture-shock of his relocation to Hollywood, the German-born Möller directed "six actresses to hold a smile for as long as they could, up to
one and half hours. Each ongoing smile is scrutinized by an emotion recognition system, and whenever the display of happiness fell below a certain threshold, an alarm alerted them to show more sincerity” [Möller]. The installation replays recordings of the analyzed video on six flat panel monitors, with the addition of a fluctuating graphic level-meter to indicate the strength of each actress’ smile. The technical implementation of this artwork’s vision-based emotion recognition system is quite sophisticated.

As can be seen from the examples above, artworks employing computer vision range from the highly formal and abstract, to the humorous and sociopolitical. They concern themselves with the activities of willing participants, paid volunteers, or unaware strangers. And they track people of interest at a wide variety of spatial scales, from extremely intimate studies of their facial expressions, to the gestures of their limbs, and to movements of entire bodies. The examples above represent just a small selection of notable works in the field, and of ways in which people (and objects) have been tracked and dissected by video analysis. Other noteworthy artworks which use machine vision include Marie Sester's Access; Joachim Sauter and Dirk Lüsebrink's Zerseher and Bodymover; Scott Snibbe's Boundary Functions and Screen Series; Camille Utterback and Romy Achituv's TextRain; Jim Campbell's Solstice; Christa Sommerer and Laurent Mignonanue's A-Volve; Danny Rozin's Wooden Mirror; Chico MacMurtrie's Skeletal Reflection, and various works by Simon Penny, Toshio Iwai, and numerous others. No doubt many more vision-based artworks remain to be created, especially as these techniques gradually become incorporated into developing fields like physical computing and robotics.

II. ELEMENTARY COMPUTER VISION TECHNIQUES

To understand how novel forms of interactive media can take advantage of computer vision techniques, it is helpful to begin with an understanding of the kinds of problems that vision algorithms have been developed to address, and their basic mechanisms of operation. The fundamental challenge presented by digital video is that it is computationally "opaque." Unlike text, digital video data in its basic form — stored solely as a stream of rectangular pixel buffers — contains no intrinsic semantic or symbolic information. There is no widely agreed-upon standard for representing the content of video, in a manner analogous to HTML, XML or even ASCII for text (though some new initiatives, notably the MPEG-7 description language, may evolve into this in the future). As a result, a computer, without additional programming, is unable to answer even the most elementary questions about whether a video stream contains a person or object, or whether an outdoor video scene shows daytime or nighttime, etcetera. The discipline of computer vision has developed to address this need.

Many low-level computer vision algorithms are geared to the task of distinguishing which pixels, if any, belong to people or other objects of interest in the scene. Three elementary techniques for accomplishing this are frame differencing, which attempts to locate features by detecting their movements; background subtraction, which locates visitor pixels according to their difference from a known background scene; and brightness thresholding, which uses hoped-for differences in luminosity between foreground people and their background environment. These algorithms, described below, are extremely simple to implement and help constitute a base of detection schemes from which sophisticated interactive systems may be built. (Complete implementations of these algorithms, written in the popular Processing flavor of Java, appear in code listings at the end of this article.)

Detecting motion. (Code Listing 1)
The movements of people (or other objects) within the video frame can be detected and quantified using a straightforward method called frame differencing. In this technique, each pixel in a video frame F1 is compared with its corresponding pixel in the subsequent frame F2. The difference in color and/or brightness between these two pixels is a measure of the amount of
movement in that particular location. These differences can be summed across all of the pixels’ locations, in order to provide a single measurement of the aggregate movement within the video frame. In some motion detection implementations, the video frame is spatially subdivided into a grid of cells, and the values derived from frame differencing are reported for each of the individual cells. For accuracy, the frame differencing algorithm depends on relatively stable environmental lighting, and on having a stationary camera (unless it is the motion of the camera which is being measured).

**Detecting presence.** (Code Listing 2)
A technique called *background subtraction* makes it possible to detect the presence of people or other objects in a scene, and to distinguish the pixels which belong to them from those which do not. The technique operates by comparing each frame of video with a stored image of the scene's background, captured at a point in time when the scene was known to be empty. For every pixel in the frame, the absolute difference is computed between its color and that of its corresponding pixel in the stored background image; areas which are very different from the background are likely to represent objects of interest. Background subtraction works well in heterogeneous environments, but it is very sensitive to changes in lighting conditions, and depends on objects of interest having sufficient contrast against the background scene.

**Detection through brightness thresholding.** (Code Listing 3)
With the aid of controlled illumination (such as backlighting) and/or surface treatments (such as high-contrast paints), it is possible to ensure that objects of interest are considerably darker than, or lighter than, their surroundings. In such cases objects of interest can be distinguished based on their brightness alone. To do this, each video pixel's brightness is compared to a threshold value, and tagged as foreground or background accordingly.

**Simple object tracking.** (Code Listing 4)
A rudimentary scheme for object tracking, ideal for tracking the location of a single illuminated point (such as a flashlight), finds the location of the single brightest pixel in every fresh frame of video. In this algorithm, the brightness of each pixel in the incoming video frame is compared with the brightest value yet encountered in that frame; if a pixel is brighter than the brightest value yet encountered, then the location and brightness of that pixel are stored. After all of the pixels have been examined, then the brightest location in the video frame is known. This technique relies on an operational assumption that there is only one such object of interest. With trivial modifications, it can equivalently locate and track the darkest pixel in the scene, or track multiple, differently-colored objects.

**Basic Interactions.**
Once a person's body pixels have been located (through the aid of techniques like background subtraction and/or brightness thresholding), this information can be used as the basis for graphical responses in interactive systems. In a 2003 Master's thesis, *Unencumbered Full Body Interaction in Video Games*, Jonah Warren presents an elegant vocabulary of various essential interaction techniques which can use this kind of body-pixel data. These schema are useful in "mirror-like" contexts, such as Myron Krueger's *Videoplace*, or a game like the PlayStation *Eye-Toy*, in which the participant can observe his own image or silhouette composited into a virtual scene.
Three of the interactions Warren identifies and explains are the Contact interaction, which can trigger an event when a user's digital silhouette comes into contact with a graphic object; the Overlap interaction, which is a continuous metric based on the percentage of pixels shared between a user's silhouette and a graphic object; and the Reflect interaction, which computes the angle of reflection when a moving object strikes the user's silhouette (and deflects the object appropriately). Documentation of several charming games which make use of these interactions can be found in Warren's site. As Warren explains it, the implementation of these interactions requires little more than counting pixels [Warren].

Naturally, many more software techniques exist, at every level of sophistication, for detecting, recognizing, and interacting with people and other objects of interest. Each of the tracking algorithms described above, for example, can be found in elaborated versions which amend its various limitations. Other easy-to-implement algorithms can compute specific features of a tracked object, such as its area, center of mass, angular orientation, compactness, edge pixels, and contour features such as corners and cavities. On the other hand, some of the most difficult-to-implement algorithms, representing the cutting edge of computer vision research today, are able (within limits) to recognize unique people, track the orientation of a person's gaze, or correctly identify facial expressions. Pseudocodes, source codes, and/or ready-to-use, executable implementations of all of these techniques can be found on the Internet in excellent resources like Daniel Huber's Computer Vision Homepage [Huber], Robert Fisher's HIPR (Hypermedia Image Processing Reference) [Fisher], or in the software toolkits discussed in Section IV, below.
Unlike the human eye and brain, no computer vision algorithm is completely "general", which is to say, able to perform its intended function given any possible video input. Instead, each software tracking or detection algorithm is critically dependent on certain unique assumptions about the real-world video scene it is expected to analyze. If any of these expectations is not met, then the algorithm can produce poor or ambiguous results, or even fail altogether. For this reason, it is essential to design physical conditions in tandem with the development of computer vision code, and/or to select software techniques which are best compatible with the available physical conditions.

Background subtraction and brightness thresholding, for example, can fail if the people in the scene are too close in color or brightness to their surroundings. For these algorithms to work well, it is greatly beneficial to prepare physical circumstances which naturally emphasize the contrast between people and their environments. This can be achieved with lighting situations that silhouette the people, for example, or through the use of specially-colored costumes. The frame-differencing technique, likewise, fails to detect people if they are stationary, and will therefore have very different degrees of success detecting people in videos of office waiting rooms compared with, for instance, videos of the Tour de France bicycle race.

A wealth of other methods exists for optimizing physical conditions in order to enhance the robustness, accuracy and effectiveness of computer vision software. Most are geared towards ensuring a high-contrast, low-noise input image. Under low-light conditions, for example, one of the most helpful such techniques is the use of infrared (IR) illumination. Infrared, which is invisible to the human eye, can supplement the light detected by conventional black-and-white security cameras. Using IR significantly improves the signal-to-noise ratio of video captured in low-light circumstances, and can even permit vision systems to operate in (apparently) complete darkness.

Another physical optimization technique is the use of retroreflective marking materials, such as those manufactured by 3M Corporation for safety uniforms. These materials are remarkably efficient at reflecting light back towards their source of illumination, and are ideal aids for ensuring high-contrast video of tracked objects. If a small light is placed coincident with the camera’s axis, objects with retroreflective markers will be detected with tremendous reliability.

Finally, some of the most powerful physical optimizations for machine vision can be made without intervening in the observed environment at all, through well-informed selections of the imaging system’s camera, lens, and frame-grabber components. To take one example, the use of a "telecentric" lens can significantly improve the performance of certain kinds of shape-based or size-based object recognition algorithms. For this type of lens, which has an effectively infinite focal length, magnification is nearly independent of object distance. As one manufacturer describes it, "an object moved from far away to near the lens goes into and out of sharp focus, but its image size is constant. This property is very important for gaging three-dimensional objects, or objects whose distance from the lens is not known precisely" [Melles Griot]. Likewise, polarizing filters offer a simple, non-intrusive solution to another common problem in video systems, namely glare from reflective surfaces. And a wide range of video cameras is available, optimized for conditions like high-resolution capture, high-frame-rate capture, short exposure times, dim light, ultraviolet light, or thermal imaging. Clearly, it pays to research imaging components carefully.

As we have seen, computer vision algorithms can be selected to best negotiate the physical conditions presented by the world, and likewise, physical conditions can be modified to be more easily legible to vision algorithms. But even the most sophisticated algorithms and highest-quality hardware cannot help us find meaning where there is none, or track an object which cannot be described in code. It is therefore worth emphasizing that some visual features contain more information about the world, and are also more easily detected by the computer, than others. In
designing systems to "see for us," we must not only become freshly awakened to the many things about the world which make it visually intelligible to us, but also develop a keen intuition about their ease of computability. The sun is the brightest point in the sky, and by its height also indicates the time of day. The mouth cavity is easily segmentable as a dark region, and the circularity of its shape is also closely linked to vowel sound. The pupils of the eye emit an easy-to-track infrared retroreflection, and they also indicate a person's direction of gaze. Or in the dramatic case of Natalie Jeremijenko's *Suicide Box*, discussed earlier: vertical motion in the video frame is easy to find through simple frame-differencing, and (in a specific context) it can be a stark indicator of a tragic event. In judging which features in the world are most profitably selected for analysis by computer vision, we will do well to select those graphical facts about the world which not only are easy to detect, but also simplify its semantic understanding.

IV. COMPUTER VISION IN MULTIMEDIA AUTHORING TOOLS

The last decade has witnessed a profound transformation in the ease-of-use of software authoring tools for art and design students, and for novice programmers generally. While multimedia authoring environments are now commonly used to create interactive experiences for the World Wide Web, it is now equally common that these tools are used to create art installations, performances, commercial kiosks, and interactive industrial design prototypes. With the gradual incorporation of live video cameras into the palette of available computer inputs, the demand for straightforward computer vision capabilities has grown as well.

It can be an especially rewarding experience to implement machine vision techniques directly from first principles, using code such as the examples provided in this article. To make this possible, the only requirement of one's software development environment is that it should provide direct read-access to the array of video pixels obtained by the computer's frame-grabber. *Processing* is one such environment, which, through an abundance of graphical capabilities, is extremely well-suited to the electronic arts and visual design communities. Used worldwide by students, artists, designers, architects, and researchers for learning, prototyping, and production, Processing obtains live video through a QuickTime-based interface, and allows for fast manipulations of pixel buffers with a Java-based scripting language [Fry]. The examples which appear in this article are written in Processing code.

Hopefully, the example algorithms discussed earlier illustrate that creating low-level vision algorithms from first principles isn't so hard. Of course, a vast range of functionality can also be immediately obtained from readymade, "off-the-shelf" solutions. Some of the most popular machine vision toolkits take the form of "plug-ins" or extension libraries for commercial authoring environments geared towards the creation of interactive media. Such plug-ins simplify the developer's problem of connecting the results of the vision-based analysis to the audio, visual and textual affordances generally provided by such authoring systems.

Aficionados of Macromedia's popular *Director* software, for example, can choose vision plug-ins ("Xtras") such as Danny Rozin's *TrackThemColors*, and Joshua Nimoy's *Myron* (named in honor of Myron Krueger). Rozin's inexpensive plug-in can track multiple objects in the video according to their chroma or brightness [Rozin]. Nimoy's newer toolkit, which is freeware and open source, provides more detailed data about the tracked objects in the scene, such as their bounding quads and contour pixels [Nimoy]. Through Director, the features detected by these Xtras can be linked to the control of sound playback, 2D and 3D graphics, text, and serial communications.

Many vision plug-ins have been developed for *Max/MSP/Jitter*, a visual programming environment which is widely used by electronic musicians and VJs. Originally developed at the Parisian IRCAM research center in the mid-1980s, and now marketed commercially by the
California-based Cycling’74 company, this extensible environment offers powerful control of (and connectivity between) MIDI devices, real-time sound synthesis and analysis, OpenGL-based 3D graphics, video filtering, network communications, and serial control of hardware devices [Cycling’74]. The various computer vision plug-ins for Max/MSP/Jitter, such as David Rokeby’s SoftVNS, Eric Singer’s Cyclops, and Jean-Marc Pelletier’s CV.Jit, can be used to trigger any Max processes or control any system parameters. Pelletier’s toolkit, which is the most feature-rich of the three, is also the only which is freeware. CV.Jit provides abstractions to assist users in tasks such as image segmentation, shape and gesture recognition, motion tracking, etc. as well as educational tools that outline the basics of computer vision techniques [Pelletier].

Some computer vision toolkits take the form of stand-alone applications, and are designed to communicate the results of their analyses to other environments (such as Processing, Director or Max) through protocols like MIDI, serial RS-232, UDP or TCP/IP networks. BigEye, developed by the STEIM (Studio for Electro-Instrumental Music) group in Holland, is a simple and inexpensive example. BigEye can track up to 16 objects of interest simultaneously, according to their brightness, color and size. The software allows for a simple mode of operation, in which the user can quickly link MIDI messages to many object parameters, such as position, speed and size [STEIM]. Another example is the powerful EyesWeb open platform, a free system developed at the University of Genoa. Designed with a special focus on the analysis and processing of expressive gesture, EyesWeb includes a collection of modules for real-time motion tracking and extraction of movement cues from human full-body movement; a collection of modules for analysis of occupation of 2D space; and a collection of modules for extraction of features from trajectories in 2D space [Camurri]. EyesWeb’s extensive vision affordances make it highly recommended for students.

The most sophisticated toolkits for computer vision generally demand greater familiarity with digital signal processing, and require developers to program in compiled languages like C++, rather than interpreted languages like Java, Lingo or Max. The Intel Integrated Performance Primitives (IPP) library for example, is among the most general commercial solutions available for computers with Intel-based CPUs [Intel]. The OpenCV library, by contrast, is a free, open-source toolkit with nearly similar capabilities, and a tighter focus on commonplace computer vision tasks [Davies]. The capabilities of these tools, as well as all of those mentioned above, are continually evolving.

V. AN EXAMPLE: LIMBOTIME

In October 2004, I conducted a workshop in machine vision for young artists and designers at the Benetton Fabrica center in Treviso, Italy. The first day of the workshop covered the art-historical uses of computer vision, presented in Section I, and the design considerations discussed in Sections II, and III; on the second day, the participants broke into small groups and were charged with the task of designing and programming a vision-based system "from scratch." Processing was used as the development environment; the workshop participants were, for the most part, either novice programmers or intermediate-level programmers with little exposure to machine vision techniques.

LimboTime is an example of an interactive game which emerged from this workshop. In LimboTime, two players stand at opposite sides of the field of vision of a regular webcam, with their fingers or arms extended towards each other. The game system locates their extended fingers, and connects a horizontal line between them. This line continually connects these points, even if the players move their hands. A third player then tries to "limbo" underneath the imaginary line created by the first two players. The application tracks the vertical position of the "limboer" relative to the imaginary line; if the limboer goes above the limbo line, then the system sounds an
alarm and the limboer is retired. If the limboer can pass completely under the line, however, then her companions lower their line-generating hands somewhat, and the process begins again.
Figures 12-15. Stills captured from the *LimboTime* game system developed by workshop participants at the Fabrica research center in Treviso, Italy. The participant shown attempts to pass below an imaginary line drawn between the two black rectangles. If she crosses above the line, the game rings an alarm. The black rectangles are temporary substitutes for the extended hands of two other participants (not shown).

*LimboTime* is a simple game, conceived and implemented in a single afternoon. Its implementation grew organically from its creators' discovery of a wall-size sheet of white Foamcore in the scrap closet. Realising that they possessed an ideal environment for brightness-based thresholding, they used this technique in order to locate the games' three players against their background. Detecting the players' hands and heads was then accomplished with simple heuristics, e.g. the limboer's head is the topmost point of the middle (non-side-touching) blob of black pixels. *LimboTime* was just one of the many interesting applications developed in the Fabrica workshop; other examples included a system to detect and record the arrivals and
departures of birds in a nearby tree, and a system which allowed its creators to "paint" an image by waving their glowing mobile phones around a dark room.

VI. CONCLUSION

Computer vision algorithms are increasingly used in interactive and other computer-based artworks to track people's activities. Techniques exist which can create real-time reports about people's identities, locations, gestural movements, facial expressions, gait characteristics, gaze directions, and other characteristics. Although the implementation of some vision algorithms require advanced understandings of image processing and statistics, a number of widely-used and highly effective techniques can be implemented by novice programmers in as little as an afternoon. For artists and designers who are familiar with popular multimedia authoring systems like Macromedia Director and Max/MSP/Jitter, a wide range of free and commercial toolkits are additionally available which provide ready access to more advanced vision functionalities.

Since the reliability of computer vision algorithms is limited according to the quality of the incoming video scene, and the definition of a scene's "quality" is determined by the specific algorithms which are used to analyze it, students approaching computer vision for the first time are encouraged to apply as much effort to optimizing their physical scenario as they do to their software code. In many cases, a cleverly designed physical environment can permit the tracking of phenomena that might otherwise require much more sophisticated software. As computers and video hardware become more available, and software-authoring tools continue to improve, we can expect to see the use of computer vision techniques increasingly incorporated into media-art education, and into the creation of games, artworks and many other applications.

VII. ACKNOWLEDGEMENTS

The author is grateful to Andy Cameron of Fabrica Center, Treviso, for organizing the workshop which led to this article; Jonathan Harris, Francesca Granato and Juan Ospina for their LimboTime project and its documentation; David Rokeby and Rafael Lozano-Hemmer for their kind permissions; and Casey Reas, who commissioned this article and made it possible in numerous different ways.

SOURCE CODE EXAMPLES

The following programs are known to work with Processing version 135.

Code Listing 1: Frame Differencing.  
Locate and Quantify the amount of movement in the video frame using frame-differencing.

Code Listing 2: Background Subtraction.  
Detect the presence of people and objects in the frame using a simple background-subtraction technique.

Code Listing 3: Brightness Thresholding.  
Determines whether a test location (such as the cursor) is contained within the silhouette of a dark object.
Code Listing 4: Brightness Tracking.
Tracks the brightest pixel in a live video signal.

REFERENCES

COMPUTER VISION SOFTWARE TOOLKITS


TEXTS AND ARTWORKS


Huber, Daniel et. al. *The Computer Vision Homepage*. 
http://www-2.cs.cmu.edu/~cil/vision.html


http://www.tmema.org/messa

http://www.flong.com/texts/publications/pub_npar04/

http://www.fundacion.telefonica.com/at/rh/e proyecto.html


http://www.christian-moeller.com/

http://homepage.mac.com/davidrokeby/sorting.html

http://www.wired.com/news/culture/0,1284,64720,00.html


Automation of Sight:
From Photography to Computer Vision

Lev Manovich

An earlier version of this paper was presented at a conference "Photography and the Photographic: Theories, Practices, Histories," University of California at Riverside, April 1994.

Prologue

Nothing perhaps symbolizes mechanization as dramatically as the first assembly lines installed by Henry Ford in U.S. factories in 1913. It seemed that mechanical modernity was at its peak. Yet, in the same year the Spanish inventor Leonardo Torres y Quevedo had already advocated the industrial use of programmed machines.¹ He pointed out that although automatons existed before, they were never used to perform useful work:

The ancient automatons...imitate the appearance and movement of living beings, but this has not much practical interest, and what is wanted is a class of apparatus which leaves out the mere visible gestures of man and attempts to accomplish the results which a living person obtains, thus replacing a man by a machine.²

With mechanization, work is performed by a human but his or her physical labor is augmented by a machine. Automation takes mechanization one step further: the machine is programmed to replace the functions of human organs of observation, effort, and decision.

Mass automation was made possible by the development of digital computers during World War II and thus became synonymous with computerization. The term "automation" was coined in 1947; and in 1949 Ford began the construction of the first automated factories.

Barely a decade later, automation of imaging and of vision were well under way. By the early 1960s, construction of static and moving two-dimensional and perspectival images, correction of artifacts in photographs, the identification of objects from their images, and many other visual tasks were already handled by computers. A number of new disciplines were emerging as well -- computer image processing, computer vision, computer graphics, computer-aided design.
What these new disciplines had all in common is that they employed perspectival images. In other words, automation of imaging and vision was first of all directed at perspectival sight.

The reasons for this are two-fold. On the one hand, by the time digital computers became available, modern society was already heavily invested in lens-based methods of image gathering (photography, film, television) which all produced perspectival images. Therefore, it is not surprising that it would want first of all to automate various uses of such images in order to obtain a new return from its investment. On the other hand, the automation of perspectival sight has already begun well before this century with the development of perspective machines, descriptive and perspective geometry and, of course, photography. Computers certainly proved to be very fast perspectival machines, but they were hardly the first.

Perspective, Perspectival Machines, Photography

From the moment of adaptation of perspective, artists and draftsmen have attempted to aid the laborious manual process of creating perspectival images. Between the sixteenth and the nineteenth century various "perspectival machines" were constructed. They were used to construct particularly challenging perspectival images, to illustrate the principles of perspective, to help students learn how to draw in perspective, to impress artists' clients, or to serve as intellectual toys. Already in the first decades of the sixteenth century, Dürer described a number of such machines. One device is a net in the form of a rectangular grid, stretched between the artist and the subject. Another uses a string representing a line of sight. The string is fixed on one end, while the other end is moved successively to key points on the subject. The point where the string crosses the projection plane, defined by a wooden frame, is recorded by two crossed strings. For each position, a hinged board attached to the frame is moved and the point of intersection is marked on its surface. It is hard to claim
that such a device, which gave rise to many variations, made the creation of perspectival images more efficient, however the images it helped to produce had reassuring mechanical precision. Other major types of perspectival machines that appeared subsequently included the perspectograph, pantograph, physionotrace, and optigraph.

Why manually move the string imitating the ray of light from point to point? Along with perspectival machines a whole range of optical apparatuses was in use, particularly for depicting landscapes and in conducting topographic surveys. They included versions of camera obscura from large tents to smaller, easily transportable boxes. After 1800, the artist's ammunition was strengthened by camera lucida, patented in 1806.\(^5\) Camera lucida utilized a prism with two reflecting surfaces at 135°. The draftsman carefully positioned his eye to see both the image and the drawing surface below and traced the outline of the image with a pencil.

Optical apparatuses came closer than previous perspectival devices to the automation of perspectival imaging. However, the images produced by camera obscura or camera lucida were only ephemeral and considerable effort was still required to fix these images. A draftsman had to meticulously trace the image to transform it into the permanent form of a drawing.

With photography, this time-consuming process was finally eliminated. The process of imaging physical reality, the creation of perspectival representations of real objects was now automated. Not surprisingly, photography was immediately employed in a variety of fields, from aerial photographic surveillance to criminal detection. Whenever the real had to be captured, identified, classified, stored, photography was put to work.

Photography automated one use of perspectival representation -- but not others. According to Bruno Latour, the greatest advantage of perspective over other kinds of representations is that it establishes a “four-lane freeway” between physical reality and its representation.\(^6\) We can combine real and imagined objects in a single geometric model and go back and forth between reality and the model. By the twentieth century, the creation of a geometric model of both existing and imagined reality still remained a time consuming manual process, requiring the techniques of perspectival and analytical
Manovich -- 5

geometry, pencil, ruler, and eraser. Similarly, if one wanted to visualize the model in perspective, hours of drafting were required. And to view the model from another angle, one had to start all over again. The automation of geometrical modeling and display had to wait the arrival of digital computers.

3-D Computer Graphics: Automation of Perspectival Imaging

Digital computers were developed towards the end of World War II. The automation of the process of constructing of perspectival images of both existent and non-existent objects and scenes followed quite soon. By the early 1960s Boeing designers already relied on 3-D computer graphics for the simulation of landings on the runway and of pilot movement in the cockpit.

By automating perspectival imaging, digital computers completed the process which began in the Renaissance. This automation became possible because perspectival drawing has always been a step-by-step procedure, an algorithm involving a series of steps required to project coordinates of points in 3-D space onto a plane. Before computers the steps of the algorithm were executed by human draftsmen and artists. With a computer, these steps can be executed automatically and, therefore, much more efficiently.

The details of the actual perspective-generating algorithm which could be executed by a computer were published in 1963 by Lawrence G. Roberts, then a graduate student at MIT. The perspective-generating algorithm constructs perspectival images in a manner quite similar to traditional perspectival techniques. In fact, Roberts had to refer to German textbooks on perspectival geometry from the early 1800s to get the mathematics of perspective. The algorithm reduces reality to solid objects, and the objects are further reduced to planes defined by straight lines. The coordinates of the endpoint of each line are stored in a computer. Also stored are the parameters of a virtual camera -- the coordinates of a point of view, the direction of sight, and the position of a
projection plane. Given this information, the algorithm generates a perspectival image of an object, point by point.

The subsequent development of computer graphics can be seen as the struggle to automate other operations involved in producing perspectival stills and moving images. The computerization of perspectival construction made possible the automatic generation of a perspectival image of a geometric model as seen from an arbitrary point of view -- a picture of a virtual world recorded by a virtual camera. But, just like with the early perspectival machines described by Dürer, early computer graphics systems did not really save much time over traditional methods. To produce a film of a simulated landing, Boeing had to supplement computer technology with manual labor. As in traditional animation, twenty-four plots were required for each second of film. These plots were computer-generated and consisted of simple lines. Each plot was then hand-colored by an artist. Finished plots were filmed, again manually, on an animation stand. Gradually, throughout the 1970s and the 1980s, the coloring stage was automated as well. Many algorithms were developed to add the full set of depth cues to a synthetic image -- hidden line and hidden surface removal, shading, texture, atmospheric perspective, shadows, reflections, and so on.

At the same time, to achieve interactive perspectival display, special hardware was built. Each step in the process of 3-D image synthesis was delegated to a special electronic circuit: a clipping divider, a matrix multiplier, a vector generator. Later on, such circuits became specialized computer chips, connected together to achieve real-time, high resolution, photorealistic 3-D graphics. Silicon Graphics Inc., one of the major manufacturers of computer graphics hardware, labeled such a system "geometry engine."

The term appropriately symbolizes the second stage of the automation of perspectival imaging. At the first stage, the photographic camera, with perspective physically built into its lens, automated the process of creating perspectival images of existing objects. Now, with the perspectival algorithm and other necessary geometric operations embedded in silicon, it become possible to display and interactively manipulate models of non-existent objects as well.
Computer Vision: Automation of Sight

In his papers, published between 1963 and 1965, Roberts formalized the mathematics necessary for generating and modifying perspective views of geometric models on the computer. This, writes William J. Mitchell, was "an event as momentous, in its way, as Brunelleschi's perspective demonstration." However, Roberts developed techniques of 3-D computer graphics having in mind not the automation of perspectival imaging but another, much more daring goal -- "to have the machine recognize and understand photographs of three dimensional objects." Thus, the two fields were born simultaneously: 3-D computer graphics and computer vision, automation of imaging and of sight.

The field of computer vision can be seen as the culmination of at least two centuries-long histories. The first is the history of mechanical devices designed to aid human perception, such as Renaissance perspectival machines. This history reaches its final stage with computer vision, which aims to replace human sight altogether. The second is the history of automata, whose construction was especially popular in the seventeenth and eighteenth centuries. Yet, despite similarity in appearance, there is a fundamental difference between Enlightenment automata which imitated human's or animal's bodily functions and the modern day robots equipped with computer vision systems, artificial legs, arms, etc. As noted by Leonardo Torres, old automata, while successfully copying the appearance and movement of living beings, had no economic value. Indeed, such voice synthesis machines as Wolgang von Kempelen's 1778 device which directly imitated the functioning of the oral cavity or Abbé Mical's Têtes Parlantes (1783) operated by a technician hiding offstage and pressing a key on a keyboard were used only for entertainment. When in 1913 Torres called for automata that would "accomplish the results which a living person obtains, thus replacing a man by a machine" he was expressing a fundamentally new idea of using automata for productive labor. A few years later, the brother of the Czech writer Karel Capek coined the word robot from the Czech word robota, which
means "forced labor."\textsuperscript{16} Capek’s play \textit{R.U.R.} (1921) and Fritz Lang’s \textit{Metropolis} (1927) clearly demonstrate this new association of automata with physical industrial labor.

Therefore, it would be erroneous to conclude that, with computer vision, twentieth century technology simply added the sense of sight to eighteenth century mechanical statues. But even to see computer vision as the continuation of Torres’, Capek’s or Lang’s ideas about industrial automation which replaces manual labor would not be fully accurate. The idea of computer vision became possible and the economic means to realize this idea became available only with the shift from industrial to post-industrial society after World War II. The attention turned from the automation of the body to the automation of the mind, from physical to mental labor. This new concern with the automation of mental functions such as vision, hearing, reasoning, problem solving is exemplified by the very names of the two new fields that emerged during the 1950s and 1960s -- artificial intelligence and cognitive psychology. The latter gradually replacing behaviorism, the dominant psychology of the "Fordism" era. The emergence of the field of computer vision is a part of this cognitive revolution, a revolution which was financed by the military escalation of the Cold War.\textsuperscript{17} This connection is solidified in the very term "artificial intelligence" which may refer simultaneously to two meanings of "intelligence": reason, the ability to learn or understand, and information concerning an enemy or a possible enemy or an area. Artificial intelligence: artificial reason to analyze collected information, collected intelligence.

In the 1950s, faced with the enormous task of gathering and analyzing written, photographic, and radar information about the enemy, the CIA and the NSA (National Security Agency) began to fund the first artificial intelligence projects. One of the earliest projects was a Program for Mechanical Translation, initiated in the early 1950s in the attempt to automate the monitoring of Soviet communications and media.\textsuperscript{18} The work on mechanical translation was probably the major cause of many subsequent developments in modern linguistics, its move towards formalization; it can be discerned in Noam Chomsky's early theory
which, by postulating the existence of language universals in the domain of
grammar, implied that translation between arbitrary human languages could be
automated. The same work on mechanical translation was also one of the
catalysts in the development of the field of pattern recognition, the precursor to
computer vision. Pattern recognition is concerned with automatically detecting
and identifying predetermined patterns in the flow of information. A typical
example is character recognition, the first stage in the process of automating
translation. Pattern recognition was also used in the U.S. for the monitoring of
Soviet radio and telephone communication. Instead of listening to every
transmission, an operator would be alerted if computer picked up certain words in
the conversation.

As a "logistics of perception" came to dominate modern warfare and
surveillance and as the space race began, image processing became another
major new field of research. Image processing comprises techniques to
improve images for human or computer interpretation. In 1964, the space
program for the first time used image processing to correct distortions in the
pictures of the Moon introduced by a on-board television camera of Ranger 7. In
1961, the National Photographic Interpretation Center (NPIC) was created to
produce photoanalysis for the rest of the U.S. intelligence community and, as
Manual De Landa points out, by the end of the next decade computers "were
routinely used to correct for distortions made by satellite's imaging sensors and
by atmospheric effects, sharpen out-of-focus images, bring multicolored single
images out of several pictures taken in different spectral bands, extract particular
features while diminishing or eliminating their backgrounds altogether..." De
Landa also notes that computer analysis of photographic imagery became the
only way to deal with the pure volume of intelligence being gathered: "It became
apparent during the 1970s that there is no hope of keeping up with the millions of
images that poured into NPIC...by simply looking at them the way they had been
looked at in World War II. The computers therefore also had to be taught to
compare new imagery of a given scene with old imagery, ignoring what had not
changed and calling the interpreter's attention to what had."
The techniques of image processing, which can automatically increase an image's contrast, remove the effects of blur, extract edges, record differences between two images, and so on, greatly eased the job of human photoanalysts. And the combining of image processing with pattern recognition made it possible in some cases to delegate the analysis of photographs to a computer. For instance, the technique of pattern matching used to recognize printed characters can also be used to recognize objects in a satellite photograph. In both cases, the image is treated as consisting of two-dimensional forms. The contours of the forms are extracted from the image are then compared to templates stored in computer memory. If a contour found in the image matches a particular template, the computer signals that a corresponding object is present in a photograph.

A general purpose computer vision program has to be able to recognize not just two-dimensional but three-dimensional objects in a scene taken from an arbitrary angle. Only then it can be used to recognize an enemy's tank, to guide an automatic missile towards its target or to control a robotic arm on the factory floor. The problem with using simple template matching is that "a two-dimensional representation of a two-dimensional object is substantially like the object, but a two-dimensional representation of a three-dimensional object introduces a perspective projection that makes the representation ambiguous with respect to the object." While pattern recognition was working for images of two-dimensional objects, such as letters or chromosomes, a different approach was required to "see" in 3-D.

Roberts' 1965 paper "Machine Recognition of Three-dimensional Solids" is considered to be the first attempt at solving the general task of automatically recognizing three-dimensional objects. His program was designed to understand the artificial world composed solely of polyhedral blocks -- a reduction of reality to geometry that would have pleased Cézanne. Using image processing techniques, a photograph of a scene was first converted into a line drawing. Next, the techniques of 3-D computer graphics were used:
Roberts' program had access to three-dimensional models of objects: a cube, a rectangular solid, a wedge, and a hexagonal prism. They were represented by coordinates \((x, y, z)\) of their vertices. A program recognized these objects in a line drawing of the scene. A candidate model was selected on the basis of simple features such as a number of vertices. Then the selected model was rotated, scaled, projected, and matched with the input line drawing. If the match was good, the object was recognized, as were its position and size. Roberts' program could handle even a composite object made of multiple primitive shapes; it subtracted parts of a line drawing from the drawing as they were recognized, and the remaining portions were analyzed further.\(^25\)

Was this enough to completely automate human vision? This depends upon how we define vision. The chapter on computer vision in The Handbook of Artificial Intelligence (1982) opens with the following definition: "Vision is the information-processing task of understanding a scene from its projected images."\(^26\) But what does "understanding a scene" mean? With computer vision research financed by the military-industrial complex, the definition of understanding becomes highly pragmatic. In the best tradition of the pragmatism of James and Pierce, cognition is equated with action. The computer can be said to "understand" a scene if it can act on it -- move objects, assemble details, destroy targets. Thus, in the field of computer vision "understanding a scene" implies two goals. First, it means the identification of various objects represented in an image. Second, it means reconstruction of three-dimensional space from the image. A robot, for instance, need not only recognize particular objects, but it has to construct a representation of the surrounding environment to plan its movements. Similarly, a missile not only has to identify a target but also to determine the position of this target in three-dimensional space.

It can be seen that Roberts' program simultaneously fulfilled both goals. His program exemplified the approach taken by most computer vision researchers in the following two decades. A typical program first reconstructs the
three-dimensional scene from the input image and then matches the reconstructed objects to the models stored in memory. If the match is good, the program can be said to recognize the object, while simultaneously recording its position.

A computer vision program thus acts like a blind person who "sees" objects (i.e., identifies them) by reading their shapes through touch. As for a blind person, understanding the world and the recognition of shapes are locked together; they cannot be accomplished independently of one another.

In summary, early computer vision was limited to recognition of two-dimensional forms. Later, researchers began to tackle the task of recognizing 3-D objects which involves reconstruction of shapes from their perspectival representations (a photograph or a video image). From this point on, the subsequent history of computer vision research can be seen as a struggle against perspective inherent to the photographic optics.

The Retreat of Perspective

With the emergence of the field of computer vision, perspectival sight reaches its apotheosis and at the same time begins its retreat. At first computer vision researchers believed that they could invert the perspective and reconstruct the represented scene from a single perspectival image. Eventually, they realized that it is often easier to bypass perspectival images altogether and use other means as a source of three-dimensional information.

Latour points out that with the invention of perspective it became possible to represent absent things and plan our movement through space by working on representations. To quote him again, "one cannot smell or hear or touch Sakhalin island, but you can look at the map and determine at which bearing you will see the land when you send the next fleet." Roberts' program extended these abilities even further. Now the computer could acquire full knowledge of the three-dimensional world from a single perspectival image! And because the
program determined the exact position and orientation of objects in a scene, it
became possible to see the reconstructed scene from another viewpoint. It also
became possible to predict how the scene would look from an arbitrary
viewpoint. Finally, it also became possible to guide automatically the movement
of a robot through the scene.

Roberts' program worked using only a single photograph -- but only
because it was presented with a highly artificial scene and because it "knew"
what it could expect to see. Roberts limited the world which his program could
recognize to simple polyhedral blocks. The shapes of possible blocks were
stored in a computer. Others simplified the task even further by painting all
objects in a scene the same color.

However, given an arbitrary scene, composed from arbitrary surfaces of
arbitrary color and lighted in an arbitrary way, it is very difficult to reconstruct the
scene correctly from a single perspectival image. The image is
"underdetermined." First, numerous spatial layouts can give rise to the same two-
dimensional image. Second, "the appearance of an object is influenced by its
surface material, the atmospheric conditions, the angle of the light source, the
ambient light, the camera angle and characteristics, and so on," and all of these
different factors are collapsed together in the image. Third, perspective, as any
other type of projection, does not preserve many geometric properties of a
scene. Parallel lines turn into convergent lines; all angles change; equal lines
appear unequal. All this makes it very difficult for a computer to determine which
lines belong to a single object.

Thus, perspective, which until now served as a model of visual
automation, becomes the drawback which needs to be overcome. Perspective,
this first step towards the rationalization of sight (Ivins) has eventually become a
limit to its total rationalization -- the development of computer vision.

The realization of the ambiguities inherent in a perspectival image itself
came after years of vision research. In trying to compensate for these
ambiguities, laboratories began to scrutinize the formal structure of a perspectival
image with a degree of attention unprecedented in the history of perspective. For
instance, in 1968 Adolfo Guzman classified the types of junctions that appear in line representations after he realized that a junction type can be used to deduce whether regions of either side of a junction line were part of the same object. In 1986 David Lowe presented a method to calculate the probability that a particular regularity in an image (for instance, parallel lines) reflects the physical layout of the scene rather than being an accident due to a particular viewpoint. All other sources of depth information such as shading, shadows or texture gradients were also systematically studied and described mathematically.

Despite these advances, a single perspectival image remained too ambiguous a source of information for practical computer vision systems. An alternative has been to use more than one image at a time. Computer stereo systems employ two cameras which, like human eyes, are positioned a distance apart. If the common feature can be identified in both images, then the position of an object can be simply determined through geometric calculations. Other systems use a series of continuous images recorded by a video camera.

But why struggle with the ambiguity of perspectival images at all? Instead of inferring three-dimensional structure from a two-dimensional representation, it is possible to measure depth directly by employing various remote sensing technologies. In addition to video cameras, modern vision systems also utilize a whole range of different range finders such as lasers or ultrasound. Range finders are devices which can directly produce a three-dimensional map of an object. The same basic principle employed in radar is used: the time required for an electro-magnetic wave to reach an object and reflect back is proportional to the distance to the object. But while radar reduces an object to a single point and in fact is blind to close-by objects, a range finder operates at small distances. By systematically scanning the surface of an object, it directly produces a "depth map," a record of an object's shape which can be then matched to geometric models stored in computer memory thus bypassing the perspectival image altogether.

Thus, perspective occupies a special role in the history of computer imaging. A first algorithm .... Yet, while giving rise to new technologies of "geometric vision," perspective also becomes a limit to the final automation of sight -- recognition of objects by a computer.
Perspective, this first step towards the "rationalization of sight" (Ivins) has eventually become a limit to its total rationalization -- the development of computer vision. The perspective algorithm, a foundation of both computer graphics and computer vision, is used to generate perspectival views given a geometric model and to deduce the model given a perspectival view. Yet, while giving rise to new technologies of "geometric vision," perspective also becomes a limit to the final automation of sight -- recognition of objects by a computer. Finally, it is displaced from its privileged role, becoming just one among other techniques of space mapping and visualization.

Epilogue

The Renaissance's adaptation of perspective represented the first step in the automation of sight. While other cultures used sophisticated methods of space mapping, the importance of perspective lies not in its representational superiority but in its algorithmic character. This algorithmic character enabled the gradual development of visual languages of perspective and descriptive geometry and, in parallel, of perspectival machines and technologies, from a simple net described by Dürer to photography and radar. And when digital computers made possible mass automation in general, automation of perspectival vision and imaging followed soon.

The use of computers allowed to extend perspective, utilizing to the extreme its inherent qualities such as the algorithmic character and the reciprocal relationship it establishes between reality and representation. The perspective algorithm, a foundation of both computer graphics and computer vision, is used to generate perspectival views given a geometric model and to deduce the model given a perspectival view. Yet, while giving rise to new technologies of "geometric vision," perspective also becomes a limit to the final automation of sight -- recognition of objects by a computer. Finally, it is
displaced from its privileged role, becoming just one among other techniques of space mapping and visualization.


2 Qtd. in ibid., 67.

3 For a survey of perspectival instruments, see Martin Kemp, The Science of Art (New Haven: Yale University Press, 1990), 167-220.

4 Ibid., 171-172.

5 Ibid., 200.


This mixture of automated and pre-industrial labor is characteristic of the early uses of computers for the production of images. In 1955 the psychologist Attneave was the first to construct an image which was to become one of the icons of the age of digital visuality -- random squares pattern. A pattern consisted of a grid made from small squares colored black or white. A computer generated table of random numbers has been used to determine the colors of the square -- odd number for one color, even number for another. Using this procedure, two research assistants manually filled in 19,600 squares of the pattern. Paul Vitz and Arnold B. Glimcher, *Modern Art and Modern Science* (New York: Praeger Publishers, 1984), 234. Later, many artists, such as Harold Cohen, used computers to generate line drawings which they then colored by hand, transferred to canvas to serve as a foundation for painting, etc.


14 "Retrospectives II: The Early Years in Computer Graphics at MIT, Lincoln Lab, and Harvard,"

57.

15 Remko Scha, "Virtual Voices," MediaMatic 7, no. 1 (1992): 33. Scha describes two fundamental approaches taken by the developers of voice imitating machines: the genetic method which imitates the physiological processes that generate speech sounds in the human body and the gennematic method which is based on the analysis and reconstruction of speech sounds themselves without considering the way in which the human body produces them. While the field of computer vision, and other fields of artificial intelligence, first clearly followed gennematic method, in the 1980s, with the
growing popularity of neural networks, there was a shift towards the genetic method -- direct imitation of the physiology of the visual system. In a number of laboratories, scientists begin to build artificial eyes which move, focus, and analyze information exactly like human eyes.

16 Eames and Eames, A Computer Perspective, 100.


18 Ibid., 214.

19 The first paper on image processing was published in 1955.


22 Within the field of computer vision, a scene is defined as a collection of three-dimensional objects depicted in an input picture. David McArthur, "Computer Vision and Perceptual Psychology," *Psychological Bulletin* 92, no. 2 (1982): 284.


24


26 Ibid., 127.


29 Ibid., 128.

30 Ibid., 131.

Videotape Identification and Assessment Guide

Texas Commission on the Arts
2004
Museum staff are increasingly faced with the identification, care, and conservation of videotape formats found in their collections and archives, some of which are many decades old. Staff must inspect the works to assess condition – often without the use of playback equipment – and catch problems before they compromise the works. Videotape has particular storage, housing, and handling needs that when addressed, will substantially prolong its shelf life. At some point in its lifecycle, more drastic measures, such as cleaning and re-formatting, will be necessary to allow future generations to experience art works created through the use of electronic tools.

This guide was created to answer questions commonly asked by custodians of video materials:

**HOW CAN I IDENTIFY A VIDEO FORMAT AND KNOW ITS CHARACTERISTICS?**

**IDENTIFICATION OF VIDEO FORMATS** provides pictures and identifying features of 15 videotape formats.  ➤ page 3

**WHAT ARE THE RISKS TO VIDEO MATERIALS?**

**RISKS TO VIDEOTAPE LONGEVITY** gives an overview of common damage caused by people, machines, the environment, and equipment obsolescence.  ➤ page 37

**HOW CAN I EVALUATE THE CONDITION OF A VIDEOTAPE?**

**CONDITION ASSESSMENT** explains how to inspect videotapes and determine condition through observation and inference.  ➤ page 41

**WHAT CONSERVATION ACTIONS SHOULD BE TAKEN IF THE VIDEO IS AT RISK?**

**CONSERVATION ACTIONS** provides guidance for typical first steps through in-house actions and/or the use of vendor services.  ➤ page 44

**WHERE CAN I LEARN MORE ABOUT VIDEO PRESERVATION?**

**RESOURCES FOR VIDEO PRESERVATION** leads you to sources for more in-depth information, conservation supplies, and key vendors.  ➤ page 49

**GLOSSARY**

Gives definitions of terminology used within the guide and accompanying web site.  ➤ page 52

**SOURCES AND CREDITS**

Is our thanks to the many people that made this guide and accompanying web site possible.  ➤ page 54
The formats below represent formats or families of formats that are likely to be found in museum collections. See Resources for links to more guides including more obscure formats.

**Tape Formats**

- **OPEN REEL**
  - 2” Quad ➤ page 4
  - 1” Type C ➤ page 6
  - 1/2” Open Reel ➤ page 8

- **CASSETTE**
  - 3/4” Umatic and 3/4” Umatic SP ➤ page 10
  - Betamax ➤ page 13
  - VHS & S-VHS ➤ page 15
  - Betacam & BetacamSP ➤ page 17
  - Video8 & Hi8 ➤ page 20
  - D2 ➤ page 23
  - D3 ➤ page 25
  - Digital Betacam ➤ page 27
  - DVCAM ➤ page 29
  - MiniDV ➤ page 31
  - DVCPro ➤ page 33
  - Digital 8 ➤ page 35
## Identification

### 2” Quad

<table>
<thead>
<tr>
<th><strong>Format Name</strong></th>
<th>2” Quad (aka 2” Quadruplex)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analog or Digital</strong></td>
<td>Analog</td>
</tr>
<tr>
<td><strong>Date Introduced</strong></td>
<td>1956</td>
</tr>
<tr>
<td><strong>Dates in Use</strong></td>
<td>1956 – early 1980s</td>
</tr>
<tr>
<td><strong>Tape Width</strong></td>
<td>2”</td>
</tr>
</tbody>
</table>

**Reel Dimensions**

Approximately 12” in diameter.

**Tape Container**

Shipping cases for 2” Quad are 4” deep, 15” square and can weigh 20-30 lbs. You may also find these tapes in cardboard boxes.

**Tape Variations and/or Identifying Features**

Once you have seen a quad case, it will be easy to identify. It is substantially larger and heavier than any of the other video formats. The containers will typically have large labels with program titles, show air dates, record dates, and/or other information consistent with a broadcast television environment.

**Common Manufacturers/Brands**

3M, Ampex. Some tapes or containers will indicate the manufacturer as the Minnesota Mining and Manufacturing Company, later shortened to 3M.

**Tape Players/Recorders**

Machines for this format are very rare, especially in working order. They are found with preservation/ restoration vendors and occasionally in television stations. It is almost impossible to find parts, and some parts need to be manufactured. It is also extremely difficult to find people to repair these machines. Not all 2” quad tapes will play on the same 2” quad machine.
2” Quad

**PRIMARY USAGE**  ➤ 2” was the first video format, developed for in-studio use. Artists who have taped in this format generally produced in a studio setting, such as through a public television residency program or other television project.

**RISKS**  ➤  Due to its sheer age, 2” quad is at great risk of signal loss due to problems with the physical material, and from hardware and media obsolescence. See Risks.

**CONDITION ASSESSMENT**  ➤  Some 2” quad shipping containers have a spongy lining that deteriorates over time into a powdery and then gummy substance that is very difficult to clean off of the tape. Extensive inspection is difficult without playback, though assessment can be made for signs of binder deterioration, mold, and other problems. See Risks and Condition Assessment.

**CONSERVATION ACTIONS**  ➤  Immediate re-mastering is recommended through a vendor with proven experience with this format. Cleaning may be required before transfer. Re-housing of the original may be required. See Conservation Actions.

**RESOURCES**  ➤  The following web sites have additional information on 2” quadruplex.
  LabGuy’s World – http://www.labguysworld.com/
  Tim Stoffel’s Quadruplex Park – http://www.lionlamb.us/quadpark.html
  Vidipax – http://www.vidipax.com

See also Resources.
Identification

➤ 1” Type C

<table>
<thead>
<tr>
<th>Format Name</th>
<th>1” Type C (aka 1”)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog OR Digital</td>
<td>Analog</td>
</tr>
<tr>
<td>Date Introduced</td>
<td>1978</td>
</tr>
<tr>
<td>Dates In Use</td>
<td>1978 – 1990s</td>
</tr>
<tr>
<td>Tape Width</td>
<td>1”</td>
</tr>
</tbody>
</table>

Reel Dimensions ➤ Approximately 12” in diameter and have a distinctive wide hub.

Tape Container ➤ Common containers are made from plastic and are rectangular, or have one curved side with a handle. 1” can also be found in cardboard containers. The containers will typically have large labels with program titles, show air dates, record dates, and/or other information consistent with a broadcast television environment.

Tape Variations And/OR Identifying Features ➤ There are numerous forms of 1” tape that did not succeed in the marketplace, and some are indistinguishable except on playback. Audio recordings are also made on 1” tape and may be confused with video recordings.

Common Manufacturers/Brands ➤ Sony, Ampex, Panasonic and others.

Tape Players/Recorders ➤ Machines for this format are rare, especially in working order. They are found primarily with preservation/restoration vendors, though some television stations and collectors still have working 1” machines. It is difficult to find parts, and some parts need to be manufactured. It is also difficult to find people to repair these machines.
> 1" Type C

**PRIMARY USAGE** > 1" was typically used for in-studio recording. Artists who have tape in this format produced in a television station or other studio setting. It was also used as a format for preservation masters in the late 1980s and early 1990s.

**RISKS** > Due to its sheer age, 1" is at great risk of signal loss due to problems with the physical material, and from hardware and media obsolescence. See Risks.

**CONDITION ASSESSMENT** > Extensive inspection is difficult without playback, though assessment can be made for signs of binder deterioration, mold, and other problems. See Risks and Condition Assessment.

**CONSERVATION ACTIONS** > Re-mastering is recommended through a vendor with proven experience with this format. Cleaning may be required before transfer. Re-housing of the original may be required. See Conservation Actions.

**RESOURCES** > The following web sites have additional information on 1" quadruplex.


Vidipax – http://www.vidipax.com

See also Resources.
Identification

› 1/2" Open Reel

<table>
<thead>
<tr>
<th>FORMAT NAME</th>
<th>1/2&quot; Open Reel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL</td>
<td>Analog</td>
</tr>
<tr>
<td>DATE INTRODUCED</td>
<td>1965</td>
</tr>
<tr>
<td>DATES IN USE</td>
<td>1965 – late 1970s</td>
</tr>
<tr>
<td>TAPE WIDTH</td>
<td>1/2&quot;</td>
</tr>
</tbody>
</table>

REEL DIMENSIONS ➤ Commonly used reels are 5" in diameter or 7 1/4" in diameter.

TAPE CONTAINER ➤ The most common containers for smaller tapes are approximately 5 1/2" x 5 1/2" x 1" and made of a soft or hard plastic. Larger reels will be in containers approximately 8 3/8" x 8 3/8" x 1 1/4". The plastic container may also be inside of a separate paperboard sleeve. There are other containers that are approximately the same size but have handles.

TAPE VARIATIONS AND/OR IDENTIFYING FEATURES ➤ There are two common 1/2" tape versions – CV and AV (EIAJ Type 1) – they look the same, but will playback on different decks. CV tapes were manufactured beginning in 1965; AV was introduced in 1969. The tape reels are typically made from translucent plastic. The Sony tape containers are typically black and will say "for helical scan video recorders."

COMMON MANUFACTURERS/BRANDS ➤ Sony, Panasonic, Ampex, and others.

TAPE PLAYERS/RECORDER ➤ As noted above, not all 1/2" tapes will play on the same deck. The AV or EIAJ format became the recording standard. Decks for this format are rare, especially in working order. They are found primarily with preservation/restoration.
vendors, media arts centers, schools, and with artists and collectors. It is difficult to find parts, and some parts need to be manufactured. It is also difficult to find people to repair these decks.

**PRIMARY USAGE** ➤ The 1/2” open reel format was developed for the industrial, educational and consumer markets. Artists, independent producers, community organizers, schools, and television stations were among the groups using this format. The smaller reels were used in the first portable video recorders or “portapaks.” The larger reels were typically used for stationary recording and/or video editing. 1/2” open reel was gradually replaced with cassette formats in the mid to late 1970s.

**RISKS** ➤ Due to its sheer age, 1/2” open reel is at great risk of signal loss due to problems with the physical material, and from hardware and media obsolescence. See Risks.

**CONDITION ASSESSMENT** ➤ Extensive inspection is difficult without playback, though assessment can be made for signs of binder deterioration, mold, and other problems. 1/2” open reel tapes are known for problems with sticky shed syndrome. See Risks and Condition Assessment.

**CONSERVATION ACTIONS** ➤ Re-mastering is recommended through a vendor with proven experience with this format. Cleaning is likely to be needed before transfer. Re-housing of the original may be required. See Conservation Actions.

**RESOURCES** ➤ The following web sites have additional information on 1/2” open reel.

- BAVC Video Preservation Resources – Hardware http://www.bavc.org/preservation/dvd/resources/hardware.htm
- Vidipax – http://www.vidipax.com

See also Resources.
Identification

➤ 3/4” Umatic and 3/4” Umatic SP

**FORMAT NAME** ➤ 3/4” Umatic (aka 3/4” or Umatic) and 3/4” Umatic SP (aka 3/4” SP or Umatic SP)

**ANALOG OR DIGITAL** ➤ Analog

**DATE INTRODUCED** ➤ 3/4” Umatic – 1971
               ➤ 3/4” Umatic SP – 1986

**DATES IN USE** ➤ 3/4” Umatic – 1971 to present
               ➤ 3/4” Umatic SP – 1986 to present

**TAPE WIDTH** ➤ 3/4”

**CASSETTE DIMENSIONS** ➤ Full-size cassettes are 8 5/8” x 5 3/8” x 1 3/16”, and small cassettes are 7 1/4” x 4 5/8” x 1 3/16”.

**TAPE CONTAINER** ➤ Most common tape containers are heavy-duty plastic snap-closure boxes – typically blue, gray, black or tan. Some 3M tape boxes are black with rounded corners and have a sliding closure mechanism on the opening side.

**TAPE VARIATIONS AND/OR IDENTIFYING FEATURES** ➤ Full size cassettes are for use in recording and editing decks, and record up to 60 minutes. Small cassettes are used in field recording decks, and record up to 20 minutes in length. 3/4” Umatic cassettes are typically made from gray, black or tan plastic. Hubs can be any number of colors (blue, tan, red), and cassettes have a clear (or slightly blue) window that shows both reels. A small red plastic dot (which must be in place to record on the tape) may be found on the back of cassette. 3/4” Umatic SP cassettes are dark brown/maroon and have SP and the length written on the spine (3M brand).

**COMMON MANUFACTURERS/BRANDS** ➤ Sony, 3M (Scotch), Fuji and others.
3/4” Umatic and 3/4” Umatic SP

**TAPE PLAYERS/RECORDERS**

3/4” Umatic was a widely used format, and although production of these decks ceased in the 1990s, it is still in limited use. 3/4” Umatic SP has a superior picture quality to regular 3/4”. The number of available working decks will continue to diminish, and parts are becoming increasingly difficult to obtain. They are found primarily with preservation/restoration vendors, small production houses, media arts centers, schools, and with artists and collectors. New tape stock is still generally available. 3/4” Umatic tapes cannot be played in 3/4” Umatic SP decks. 3/4” Umatic SP tapes can be played on regular 3/4” Umatic decks, but without any improvement to the picture.

**PRIMARY USAGE**

The 3/4” Umatic format was developed for industrial and educational markets. In the 1970s and 1980s it was widely used for electronic news gathering, and up through the 1990s, it was a primary format for many artists, community activists, academic institutions, and production houses. Many artist and community videos are in this format; it was a preferred format for edit masters in the 1980s.

**RISKS**

Older 3/4” Umatic tapes, such as those from the 1970s and 1980s, are at great risk of signal loss due to problems with the physical material. Newer tapes may be in better shape, but hardware and media obsolescence is still a major issue. See Risks.

**CONDITION ASSESSMENT**

Given that 3/4” Umatic has been in use since 1971, a tape you encounter can be over 30 years old, or relatively new. Determining the age of the tape will help you judge its condition. Newer tapes can be played back for examination of picture and sound quality. For older tapes, extensive inspection is difficult without playback, though assessment can be made for...
Identification

➤ 3/4” Umatic and 3/4” Umatic SP

signs of binder deterioration, mold, and other problems. See Risks and Condition Assessment.

CONSERVATION ACTIONS ➤ Re-mastering is recommended through an experienced vendor. Cleaning may be needed before transfer. Re-housing of the original may be required. See Conservation Actions.

RESOURCES ➤ The following web sites have additional information on 3/4” Umatic.

BAVC Video Preservation Resources – Hardware – http://www.bavc.org/preservation/dvd/resources/hardware.htm
Vidipax – http://www.vidipax.com
LabGuy’s World – http://www.labguysworld.com/
See also Resources.
Betamax

<table>
<thead>
<tr>
<th>FORMAT NAME</th>
<th>Betamax (aka Beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL</td>
<td>Analog</td>
</tr>
<tr>
<td>DATE INTRODUCED</td>
<td>1975</td>
</tr>
<tr>
<td>DATES IN USE</td>
<td>1975 to late 1980s in the US</td>
</tr>
<tr>
<td>TAPE WIDTH</td>
<td>1/2&quot;</td>
</tr>
</tbody>
</table>

CASSETTE DIMENSIONS | 6 1/8" x 3 3/4" x 15/16"

TAPE CONTAINER | The most common containers are paperboard or plastic sleeves, although they can be found in hard plastic containers. (Sleeves do not significantly alter the dimensions.)

TAPE VARIATIONS AND/OR IDENTIFYING FEATURES | Cassettes are typically made from gray or black plastic, with white hubs, and a clear window that shows only the left-hand reel. The Greek symbol for Beta can be found imprinted in the upper right corner of cassette (Sony) or on the left side of the cassette spine (Scotch). Beta is now also used as a generic term for Betacam tape.

COMMON MANUFACTURERS/BRANDS | Sony, Scotch and others.

TAPE PLAYERS/RECORDERS | Betamax was a short-lived format in the US, quickly losing its place in the consumer market to VHS. Production of the decks continued until 2002 in the PAL format (a recording standard used in Europe). In the US, few decks survive, especially in working order. They are found primarily with preservation/restoration vendors, media arts centers, schools, and with artists and collectors. It is difficult to find parts or people to repair these decks.
Identification

➤ Betamax

**PRIMARY USAGE** ➤ The Betamax format was developed for the consumer, industrial, and educational markets. Although Betamax lost in the consumer marketplace in the US, this format was considered to be technically superior to VHS and was used extensively in schools, community media centers, and by artists.

**RISKS** ➤ Due to its sheer age, Betamax is at great risk of signal loss due to problems with the physical material, and from hardware and media obsolescence. See Risks.

**CONDITION ASSESSMENT** ➤ Extensive inspection is difficult without playback, though assessment can be made for signs of binder deterioration, mold, and other problems. See Risks and Condition Assessment.

**CONSERVATION ACTIONS** ➤ Re-mastering is recommended through an experienced vendor. Cleaning may be needed before transfer. Re-housing of the original may be required. See Conservation Actions.

**RESOURCES** ➤ The following web sites have additional information on Betamax.

Vidipax – http://www.vidipax.com

See also Resources.
## Identification

- **VHS and S-VHS**

<table>
<thead>
<tr>
<th>FORMAT NAME</th>
<th>VHS and S-VHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL</td>
<td>Analog</td>
</tr>
</tbody>
</table>
| DATE INTRODUCED | VHS – 1976  
S-VHS – 1987 |
| DATES IN USE | VHS – 1976 to present  
S-VHS – 1987 to present |
| TAPE WIDTH | 1/2” |

### Cassette Dimensions

- 7 3/8” x 4 1/16” x 1” for both VHS and S-VHS.

### Tape Container

- The most common containers are hard plastic snap-closure boxes measuring 8” x 4 11/16” x 1 1/8”, but tapes can also be found in paperboard or plastic sleeves.

### Tape Variations and/or Identifying Features

- Cassettes are typically made from black plastic, with white hubs, and clear windows that show both reels. A VHS (or S-VHS) logo is usually imprinted on the cassette in the upper right-hand corner, while manufacturer name is imprinted in the upper left. S-VHS tapes have an additional hole on the back of the cassette.

### Common Manufacturers/Brands

- Sony, Panasonic, JVC, Fuji, Maxell, and others.

### Tape Players/Recorders

- Decks for VHS are still in wide use, primarily in consumer markets. However, use is in decline due to widespread distribution of mainstream film on DVD. S-VHS gives a superior picture to standard VHS, and has other technical advantages. VHS and S-VHS are referred to as ‘upward compatible’: VHS tapes can be played in S-VHS decks, but tapes recorded in
VHS and S-VHS

S-VHS will not play in standard VHS decks. Users can choose multiple speeds for recording on tapes from 10 – 120 minutes long.

**PRIMARY USAGE** The VHS format was developed for the consumer market, whereas S-VHS was geared towards consumer, industrial, and educational markets. VHS was initially used as a camera and mastering format, but more recently has been used primarily for distribution (for multiple viewing copies/dubs). S-VHS was also used as a camera and mastering format. In art and education sectors, it was largely supplanted by digital video in the 1990s.

**RISKS** VHS tapes from the 1970s and 1980s are at great risk of signal loss due to problems with the physical material. See Risks.

**CONDITION ASSESSMENT** Given that VHS has been in use since 1976, a tape you encounter can be over 25 years old, or relatively new. Determining the age of the tape will help you judge its condition. Newer tapes can be played back for examination of picture and sound quality. For older tapes, extensive inspection is difficult without playback, though assessment can be made for signs of binder deterioration, mold, and other problems. See Risks and Condition Assessment.

**CONSERVATION ACTIONS** Given the length of time these formats have been in use, conservation actions should be based on age and condition assessment. For older tapes, re-mastering is recommended. Cleaning may be needed before transfer. Re-housing of the original may be required. See Conservation Actions.
Identification

➤ Betacam and BetacamSP

**FORMAT NAME** ➤ Betacam and BetacamSP

(aka Beta)

**ANALOG OR DIGITAL** ➤ Analog

**DATE INTRODUCED** ➤ Betacam – 1982

➤ BetacamSP – 1986

**DATES IN USE** ➤ Betacam – 1982 to present

➤ BetacamSP – 1986 to present

**TAPE WIDTH** ➤ 1/2"

**CASSETTE DIMENSIONS** ➤ Large cassettes are 9 5/16” x 5 11/16” x 1. They are used in recording and editing decks, and will record up to 194 minutes in length. Small cassettes measure 6 1/8” x 3 3/4” x 15/16” and are for use in cameras, recording up to 62 minutes in length.

**TAPE CONTAINER** ➤ Most common tape containers are hard plastic snap-closure boxes. Large cassette cases are 10 5/8” x 6 3/8” x 1 1/4”. Small cassette cases are 6 3/4” x 4 3/8” x 1 3/16”.

**TAPE VARIATIONS AND/OR IDENTIFYING FEATURES** ➤ Tape cassettes are generally labeled in the upper right-hand corner as Betacam or BetacamSP. Tape length is often written on the anti-static cover (door) that retracts to reveal the tape. Sony cassettes are dark blue with a black anti-static cover, and Maxell cassettes are dark gray with purple anti-static cover. Note: small Betacam and BetacamSP cassette tapes are the same size as Betamax tapes, so look carefully for a distinguishing logo.

**COMMON MANUFACTURERS/BRANDS** ➤ Sony, Ampex, Fuji, Maxell and others.
➤ Betacam and BetacamSP

**TAPE PLAYERS/RECORDERS** ➤ BetacamSP (Superior Performance) has technical advantages over Betacam. Betacam decks are no longer in production, but BetacamSP decks continue to be manufactured in a limited line. Betacam and BetacamSP are referred to as ‘upward compatible’: Betacam tapes can be played in BetacamSP decks, but tapes recorded in BetacamSP will not play in standard Betacam decks. Also, most newer decks in the Betacam family, such as Digital Betacam decks, will play BetacamSP tapes. Betacam and BetacamSP tape stocks are still available.

**PRIMARY USAGE** ➤ The Betacam and BetacamSP formats were developed for broadcast industrial, educational, and professional markets. BetacamSP has been used extensively as a broadcast format, and as a mastering format by commercial and independent producers, and by artists. It has been used as an exhibition format for artists, and a collections format for distributors of independent media. BetacamSP has also been a common choice for preservation masters in the last 10-15 years.

**RISKS** ➤ Betacam formats and decks are considered to be very durable, and parts are still available. However, Betacam runs the same risk of signal loss as other formats, due to problems with the physical material itself. Playback for Betacam will become more of an issue if BetacamSP is discontinued. However, it is expected that Sony will technically support the line for a period of years after manufacturing ceases. The obsolescence of Betacam is less of an issue as long as newer decks such as Digital Betacam continue to be options for playback.
Identification

➤ Betacam and BetacamSP

**CONDITION ASSESSMENT** ➤ A Betacam or BetacamSP tape you encounter can be 15-20 years old, or relatively new. Determining the age of the tape will help you judge its condition. Most tapes can be played back for examination of picture and sound quality. See Risks and Condition Assessment.

**CONSERVATION ACTIONS** ➤ Actions should be based on age and condition assessment. For many in the archival and conservation fields, Betacam SP is still an option for preservation masters, rather than a format needing re-mastering. See Conservation Actions.
Identification

➤ Video8 and Hi8

**Format Name** ➤ Video8 (aka 8mm) & Hi8

**Analog or Digital** ➤ Analog

**Date Introduced** ➤ Video8 – 1984
➤ Hi8 – 1989

**Dates in Use** ➤ Video8 – 1984 to present
➤ Hi8 – 1989 to present

**Tape Width** ➤ 5/16" (8mm)

**Cassette Dimensions** ➤ 3 11/16" x 2 3/8" x 9/16" for both Video8 and Hi8.

Tape container: Most common tape containers are heavy-duty clear plastic hinged boxes or heavy-duty plastic snap-closure boxes - 4" x 2 5/8" x 3/4". Some tapes (i.e. Fuji) are in a plastic sleeve.

**Tape Variations and/or Identifying Features** ➤ Tape cassettes are the same size. They are usually labeled in the bottom middle of the cassette (between the two reels) as 8mm or Hi8.

**Common Manufacturers/Brands** ➤ Sony, Panasonic, Fuji, and others.

**Tape Players/Recorders** ➤ Decks for Video8 and Hi8 formats are still in use, are still being produced and marketed, and are readily available. However, digital formats (such as MiniDV) have overtaken much of the Video8 and Hi8 market, raising questions as to how long these formats will be supported. Hi8 gives a superior picture to Video8, and has other technical advantages. Video8 and Hi8 are referred to as ‘upward compatible’.
Identification

➤ Video8 and Hi8

Video8 tapes can be played in Hi8 decks, but tapes recorded in Hi8 will not play in standard Video8 decks. Both formats can also be played in Digital 8 decks.

PRIMARY USAGE > The Video8 format was developed for the consumer market, where it was widely used through the late 1980s and 1990s. Hi8 was geared towards consumer, industrial, and educational markets. Usage of Hi8 in industrial and educational markets has decreased as use of digital formats (such as MiniDV) has increased. However, for much of the 1990s, Hi8 was a popular format for artists, community video centers, the media arts, and colleges/universities. In the consumer market Video8 is the lowest cost format, followed by Hi8, with digital formats priced higher. This may account for the format's continuing popularity.

RISKS > Video8 and Hi8 are made from thin tape that is subject to stretching. The shorter tapes – 30 and 60 min. - are more durable than the longer tapes. Users have reported dropout soon after the first recording. Metal Evaporated (ME) tape particularly is reported to have durability problems. Also, market factors suggest that Video8 and Hi8 will be phased out, in favor of digital formats. Unfortunately, due to the size, the decks are not as durable as those of larger formats, are difficult to work on, and thus are more expensive to repair relative to their cost. These factors may affect the availability of decks as these formats are phased out. On the positive side, the recent introduction of Digital 8 decks offers a new playback option. See Risks.

CONDITION ASSESSMENT > Since Video8 tapes could be approaching 20 years old, determining the age of the tape will help you judge its condition. ME tapes (see above)
Identify Marks

➤ Video8 and Hi8

should be identified. Newer tapes can be played back for examination of picture and sound quality; however, if dropout or other tape problems are observed, playback should stop until the point of re-mastering. For older tapes, extensive inspection is difficult without playback, and it will be wise to err on the side of caution. See Risks and Condition Assessment.

CONSERVATION ACTIONS ➤ Although these formats are relatively new, they are fragile and do not appear to have a very long shelf life. Re-mastering is recommended as soon as possible. Re-housing of the original may be required. See Conservation Actions.
Identification

D2

**FORMAT NAME**  ➤ D2

**ANALOG OR DIGITAL**  ➤ Digital

**DATE INTRODUCED**  ➤ 1988

**DATES IN USE**  ➤ 1988 to present

**TAPE WIDTH**  ➤ 3/4"

**CASSETTE DIMENSIONS**  ➤ Medium cassettes are 10" x 5 7/8" x 1 5/16" and small cassettes are 6 3/4" x 4 1/4" x 1 5/16". Medium cassette pictured.

**TAPE CONTAINER**  ➤ Most common tape containers are hard plastic snap-closure boxes measuring 11 1/4" x 6 13/16" x 1 9/16" for medium cassettes and 7 3/4" x 5 1/16" x 1 9/16". Small box pictured.

**TAPE VARIATIONS AND/OR IDENTIFYING FEATURES**  ➤ Tape cassettes are typically gray and are generally labeled with the name in the upper right-hand corner.

**COMMON MANUFACTURERS/BRANDS**  ➤ Sony, Ampex and others.

**TAPE PLAYERS/RECORDERS**  ➤ D2 is a relatively new format, so decks are in use, are still being produced and marketed, and are readily available. However, the decks are very expensive so in-house playback is not available within the non-profit cultural community.

**PRIMARY USAGE**  ➤ Introduced in the late 1980s, D2 was among the first digital tape formats for high-end production. D2 was developed for the high-end professional market, and has been used as a mastering...
format in such areas as advertising, television programming, and corporate applications. Artists working within a television setting may have mastered to D2.

RISKS ➤ It is difficult to predict how long D2 will be supported, considering the “format wars” among digital tape manufacturers. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape. See Risks.

CONDITION ASSESSMENT ➤ Most D2 tapes can be played back for examination of picture and sound quality, however equipment availability may be an issue. See Condition Assessment.

CONSERVATION ACTIONS ➤ D2 is not recommended as an archival video format, but given the age and durability of the media, re-mastering is not an immediate need. See Conservation Actions.
Identification

➤ D3

<table>
<thead>
<tr>
<th>FORMAT NAME</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL</td>
<td>Digital</td>
</tr>
<tr>
<td>DATE INTRODUCED</td>
<td>1990</td>
</tr>
<tr>
<td>DATES IN USE</td>
<td>1990 to present</td>
</tr>
<tr>
<td>TAPE WIDTH</td>
<td>1/2&quot;</td>
</tr>
</tbody>
</table>

| CASSETTE DIMENSIONS | 8 1/4” x 4 7/8” x 15/16". |
| TAPE CONTAINER | Most common tape containers are hard plastic snap-closure boxes measuring 9 1/16” x 5 5/8” x 1 1/4” |
| TAPE VARIATIONS AND/OR IDENTIFYING FEATURES | Tape cassettes are typically gray and are generally labeled with the name in the upper right-hand corner. |
| COMMON MANUFACTURERS/BRANDS | Panasonic and others. |
| TAPE PLAYERS/RECORDERS | D3 is a relatively new format, so decks are in use, are still being produced and marketed, and are readily available. However, the decks are very expensive so in-house playback is not available within the non-profit cultural community. |
| PRIMARY USAGE | D3 was developed for the high-end professional market, with longer tapes and costs somewhat less than with D2. D3 has been used as a mastering format in such areas as advertising, television programming, and corporate applications. Artists working within a television setting may have mastered to D3. |
| RISKS | It is difficult to predict how long D3 will be supported, considering the “format
wars” among digital tape manufacturers. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape. See Risks.

CONDITION ASSESSMENT ➤ Most D3 tapes can be played back for examination of picture and sound quality, however equipment availability may be an issue. See Condition Assessment.

CONSERVATION ACTIONS ➤ D3 is not recommended as an archival video format, but given the age and durability of the media, re-mastering may not be an immediate need. See Conservation Actions.
Identification

➤ Digital Betacam

<table>
<thead>
<tr>
<th>FORMAT NAME</th>
<th>Digital Betacam (aka DigiBeta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL</td>
<td>Digital</td>
</tr>
<tr>
<td>DATE INTRODUCED</td>
<td>1993</td>
</tr>
<tr>
<td>DATES IN USE</td>
<td>1993 to present</td>
</tr>
<tr>
<td>TAPE WIDTH</td>
<td>1/2”</td>
</tr>
</tbody>
</table>

CASSETTE DIMENSIONS ➤ Large cassettes are 9 5/16” x 5 11/16” x 1, and are for use in recording and editing decks. They will record up to 194 minutes in length. Small cassettes measure 6 1/8” x 3 3/4” x 15/16” and are for use in cameras, recording up to 62 minutes in length.

TAPE CONTAINER ➤ Most common tape containers are hard plastic snap-closure boxes. Large cassette cases are 10 5/8” x 6 3/8” x 1 1/4”. Small cassette cases are 6 3/4” x 4 3/8” x 1 3/16”.

TAPE VARIATIONS AND/OR IDENTIFYING FEATURES ➤ Tape cassettes are light gray blue (Sony and Maxell) and are generally labeled as Digital Betacam in the upper right-hand corner, and ‘for Digital’ in the upper left corner. Tape length is often written on the anti-static cover (door) that retracts to reveal the tape.

COMMON MANUFACTURERS/BRANDS ➤ Sony.

TAPE PLAYERS/RECORDERS ➤ Digital Betacam is a relatively new format, so decks are in use, are still being produced and marketed, and are readily available. Digital Betacam decks will also play Betacam SP tapes. The decks are very expensive so in-house playback is generally not available within the non-profit cultural community.
PRIMARY USAGE ► Digital Betacam was developed for the professional market. It is widely in use, especially in industrial and professional sectors. It is the industry standard in electronic newsgathering and broadcast television, and as a mastering format in advertising, high-end television programming, and corporate applications. It is also used as a format for video preservation masters.

RISKS ► With any newer video format, it is difficult to predict how long the format will be supported. Digital Betacam’s market share implies that it will not be phased out in the near future. A positive trend is that as Sony continues to introduce new formats for the broadcast market, they are also introducing decks that will play back multiple formats from the Betacam family. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape. See Risks.

CONDITION ASSESSMENT ► Most Digital Betacam tapes can be played back for examination of picture and sound quality, however equipment availability may be an issue. See Condition Assessment.

CONSERVATION ACTIONS ► For many in the archival and conservation fields, Digital Betacam is the best choice for preservation masters, and so is generally considered to be a solution, rather than a problem requiring action. See Conservation Actions.
ID: VIDEOTAPE IDENTIFICATION AND ASSESSMENT GUIDE

Identification

➤ DVCAM

**FORMAT NAME** ➤ DVCAM

**ANALOG OR DIGITAL** ➤ Digital

**DATE INTRODUCED** ➤ 1995

**DATES IN USE** ➤ 1995 to present

**TAPE WIDTH** ➤ 1/4”

**CASSETTE DIMENSIONS** ➤ Large cassettes measure 4 7/8” x 3” x 9/16”. They are used in editing decks and will record up to 184 minutes in length. Small cassettes measure 2 9/16” x 1 7/8” x 9/16”, and are used in cameras to record up to 40 minutes in length.

**TAPE CONTAINER** ➤ Most common tape containers are hard plastic snap-closure boxes. Large cassette cases are 5 3/8” x 3 11/16” x 3/4”. Small cassette cases are 3 1/16” x 2 1/2” x 3/4”.

**TAPE VARIATIONS AND/OR IDENTIFYING FEATURES** ➤ Tapes are generally bluish gray and are labeled as DVCAM in the upper right-hand corner. Large cassettes generally have a black anti-static cover, whereas small cassettes generally have a light blue anti-static cover.

**COMMON MANUFACTURERS/BRANDS** ➤ Sony.

**TAPE PLAYERS/RECORDERs** ➤ DVCAM is a relatively new format, so decks are in use, are still being produced and marketed, and are readily available.

**PRIMARY USAGE** ➤ The DVCAM format was developed by Sony for industrial, educational, and professional markets. It is used extensively for electronic news gathering, cable television, and other field production. It is also used as a mastering format by artists.
and independent producers, especially for long-form programming (such as documentaries), because the maximum tape length on a single cassette is 184 minutes.

**Risks** With any newer video format, it is difficult to predict how long the format will be supported. However, DVCAM’s market share implies that it will not be phased out in the near future. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape, and the size and durability of DVCAM is a concern. However, DVCAM is a higher quality product than older mini formats, such as Hi8; it is comparable to DVCPro. See Risks.

**Condition Assessment** Most DVCAM tapes can be played back for examination of picture and sound quality. See Condition Assessment.

**Conservation Actions** DVCAM is not an archival video format, but given the age of the media, re-mastering may not be an immediate need in the context of an institution’s overall media preservation plans. See Conservation Actions.
## Identification

- **MiniDV**

### Format Name
- MiniDV (aka DV or DVC)

### Analog or Digital
- Digital

### Date Introduced
- 1995

### Dates in Use
- 1995 to present

### Tape Width
- 1/4”

### Cassette Dimensions
- 2 9/16” x 1 7/8” x 7/16”

### Tape Container
- Most common tape containers are heavy-duty clear plastic hinged boxes - 2 7/8” x 2” x 5/8”.

### Tape Variations and/or Identifying Features
- This format was originally called DV, but is commonly known as MiniDV. Tape cassettes are generally labeled in the lower left hand (Sony) or on a sticker on the right side (Panasonic). Note that all MiniDV designations for Panasonic tapes are on stickers or the packaging, and may be covered with labels that list title or production information.

### Common Manufacturers/Brands
- Sony, Panasonic, and others.

### Tape Players/Recorders
- MiniDV is a relatively new format, so decks are in use, are still being produced and marketed, and are readily available. MiniDV uses the same tape width and signal compression as DVCAM tapes, but they record at different speeds. MiniDV tapes can be played in most DVCAM decks, but tapes recorded in DVCAM cannot be played back on a MiniDV camera or deck.

### Primary Usage
- The MiniDV format was developed for consumer, industrial, and
Educational markets. It is used extensively by artists and community activists, both in the educational sector and in independent production. Its small size and high visual quality make it popular for field acquisition (camera recording).

**Risks**

With any newer video format, it is difficult to predict how long this format will be supported. However, MiniDV’s market share implies that it will not be phased out in the near future. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape. The size and durability of MiniDV, as with any small, thin tape, is a concern, with similar issues to Hi8. See Risks.

**Condition Assessment**

Most MiniDV tapes can be played back for examination of picture and sound quality. See Condition Assessment.

**Conservation Actions**

MiniDV is not an archival video format and will need to be considered for re-mastering. See Conservation Actions.
Identification

➤ DVCPro

<table>
<thead>
<tr>
<th>FORMAT NAME ➤</th>
<th>DVCPro (aka D7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL ➤</td>
<td>Digital</td>
</tr>
<tr>
<td>DATE INTRODUCED ➤</td>
<td>1995</td>
</tr>
<tr>
<td>DATES IN USE ➤</td>
<td>1995 to present</td>
</tr>
<tr>
<td>TAPE WIDTH ➤</td>
<td>1/4”</td>
</tr>
</tbody>
</table>

CASSETTE DIMENSIONS ➤ Large cassettes measure 4 7/8” x 3” x 9/16”. They are used in editing decks and will record up to 126 minutes in length. Medium cassettes measure 3 13/16” x 2 1/2” x 9/16”, and are used in cameras to record up to 66 minutes in length. Note: Medium cassette pictured.

TAPE CONTAINER ➤ Tape containers are hard plastic snap-closure boxes. Large cassette cases are 5 1/4” x 3 5/16” x 3/4”. Medium cassette cases are 4 1/8” x 2 3/4” x 3/4”.

TAPE VARIATIONS AND/OR IDENTIFYING FEATURES ➤ Cassettes are generally dark gray with a yellow anti-static cover (Panasonic) or black with a yellow anti-static cover (Maxell) or black with a red anti-static cover (Fuji). DVCPro logo is in the upper right-hand corner, and tape length is generally listed on left. Panasonic cassettes also designate large cassettes with the letter ‘L’ after tape length (i.e. 126L).

COMMON MANUFACTURERS/BRANDS ➤ Panasonic, Maxell, Fuji.

TAPE PLAYERS/RECORDER ➤ DVCPro is a relatively new format, so decks are in use, are still being produced and marketed, and are readily available.
The DVCPro format was developed by Panasonic for industrial, educational, and professional markets. It is used for electronic news gathering, cable television, and other field production, including independent production. One of the first small digital formats, it was initially popular, but more recently has lost ground to other DV products.

**Risks** The DVCPro format uses the same tape width and compression rate as DVCAM, but the cassette housing is different, and it is not fully compatible with other digital video (DV) products. DVCPro decks will play MiniDV and DVCAM tapes; however, few DVCAM decks will play DVCPro tapes. Considering these “format wars,” the future of DVCPro is unknown. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape, and the size and durability of DVCPro is a concern. However, DVCPro is a higher quality product than older mini formats, such as Hi8. See Risks.

**Condition Assessment** Most DVCPro tapes can be played back for examination of picture and sound quality. See Condition Assessment.

**Conservation Actions** DVCPro is not an archival video format, but given the age of the media, re-mastering may not be an immediate need in the context of an institution’s overall media preservation plans. See Conservation Actions.
Identification

➤ Digital 8

<table>
<thead>
<tr>
<th>FORMAT NAME</th>
<th>Digital 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALOG OR DIGITAL</td>
<td>Digital</td>
</tr>
<tr>
<td>DATE INTRODUCED</td>
<td>1999</td>
</tr>
<tr>
<td>DATES IN USE</td>
<td>1999 to present</td>
</tr>
<tr>
<td>TAPE WIDTH</td>
<td>5/16&quot; 8mm</td>
</tr>
</tbody>
</table>

| CASSETTE DIMENSIONS | 3 11/16" x 2 3/8" x 9/16" |
| TAPE CONTAINER | Most common tape containers are heavy-duty clear plastic hinged boxes or heavy-duty plastic snap-closure boxes - 4" x 2 5/8" x 3/4" |
| TAPE VARIATIONS AND/OR IDENTIFYING FEATURES | Digital 8 is recorded on standard Hi8 tapes. The cassettes are generally labeled in the bottom middle of the cassette (between the two reels) as Hi8. There may not be any distinguishing marks to indicate that the recording is in Digital 8, though ME (metal evaporated) tapes are often used because Digital 8 records at faster head speeds |
| COMMON MANUFACTURERS/BRANDS | Sony, Panasonic, Fuji, and others |
| TAPE PLAYERS/RECORDERS | Digital 8 is a relatively new format, so decks are still in use, are still being produced and marketed, and are readily available |
| PRIMARY USAGE | The Digital 8 format was developed for the consumer market, and is sometimes used in the educational sector. Digital 8 cameras are marketed primarily to |
IDENTIFICATION

➤ Digital 8

consumers who already had 8mm or Hi-8 tapes – which could be played on the Digital 8 cameras.

ADDITIONAL TECHNICAL INFORMATION ➤
Video 8 and Hi8 tapes can be played in Digital 8 decks, but tapes recorded in Digital 8 will not play on Video8 or Hi8 decks.

RISKS ➤ With any newer video format, it is difficult to predict how long this format will be supported. Digital 8 has a marginal market share when compared to mini-DV. In terms of physical characteristics, digital tape has the same issues with deterioration as analog tape. Hi8 are made from thin tape that is subject to stretching. The shorter tapes – 30 and 60 min. - are more durable than the longer tapes. Users have reported dropout soon after the first recording. Metal Evaporated (ME) tape particularly is reported to have durability problems. Unfortunately, due to the size, the decks are not as durable as those of larger formats, are difficult to work on, and thus are more expensive to repair relative to their cost. These factors may affect the availability of decks as these formats are phased out. See Risks.

CONDITION ASSESSMENT ➤ Most Digital 8 can be played back for examination of picture and sound quality; however, if dropout or other tape problems are observed, playback should stop until the point of re-mastering. See Condition Assessment.

CONSERVATION ACTIONS ➤ Although these formats are relatively new, they are fragile and do not appear to have a very long shelf life. The need for re-mastering will need to be evaluated. See Conservation Actions.
Risks to Videotape Longevity

Videotape was manufactured for short-term production use, not as an archival medium for the long-term. Information about risks to videotape is largely anecdotal; very few controlled studies have been done. Dr. John W.C. Van Bogart, in the 1995 publication *Magnetic Tape Storage and Handling: A Guide for Libraries and Archives*, estimates a maximum life expectancy of thirty years for magnetic tape. (See Resources section.) In practice, useful shelf life depends on such variables as tape format and/or brand, storage conditions, number of recordings, tape handling, and conditions of playback. Periodic condition assessments and conservation actions can help slow deterioration and improve the chances of having viable information once an older tape is transferred to a contemporary format.

Below are basic descriptions of common risks; for more information, consult the Resources and Sources sections of this document.

**Equipment and media obsolescence and fragility**

Since videotape is dependent on playback equipment to be seen and heard, the loss of this equipment creates the greatest risk. Format identification will help you establish whether playback equipment is available for tapes in your collection. For example, some formats, such as 2” quadruplex, have been obsolete for over 25 years, so equipment, parts, and experienced technicians for formats are difficult to find. Some tape formats require specific machines to playback properly; for example, not all 3/4” Umatic tapes will play back on the same Umatic deck. The media or tape formats are also changing. One only needs to remember laser disks – marketed as a revolutionary new format – as a recent example of media obsolescence.

Machines that play back small tape formats are not considered durable; the same can be said about the tapes themselves. Formats such as 8mm and Hi8 fall into this category - thin tape in a tiny container. In addition, small decks, developed for the consumer or “pro-consumer” market, are difficult to work on, considered more consumable, and often expensive to repair relative to their original value.

**Common chemical deterioration**

Simply speaking, most videotape is composed of iron oxide particles imbedded in a binder on a base film of polyester terephthalate (PET). The binder is actually composed of a number of different substances, the primary one being polyester polyurethane – but different tape manufacturers created different formulations. These different formulations make some tapes more susceptible to deterioration than others.

The most common problem with videotape deterioration is sticky shed syndrome, where the binder absorbs moisture and undergoes chemical changes through a process called hydrolysis. These changes cause the binder and magnetic particles to become sticky and to detach, or shed, from the base film. When these substances are shed during playback, the machine can stop playing altogether.

Some older tapes may be found to have an acetate base. Acetate is subject to vinegar syndrome, where the base decomposes and creates acidic acid. High temperatures hasten the process of deterioration, whether vinegar syndrome or sticky shed syndrome. Cool, dry storage will slow chemical deterioration.
Newer tapes have different problems. For example, “metal evaporated” tapes, common in small formats like Hi8, have been seen to have corrosion problems, leading to a different type of metal oxide “shedding” with the same result - the loss of oxides leads to signal loss. See risks associated with particular formats in the format identification section.

**Mold**

Humid, warm conditions can promote the growth of mold on tape surfaces. Moldy tapes will require specialized handling and cleaning. Since mold can be toxic, care should be taken during tape inspection.

**Mechanical damage**

Playback of videotapes on inferior or improperly maintained equipment can damage or stretch a tape, resulting in signal loss. An uneven tape pack can also be created through the use of poor playback, subjecting the tape to possible edge damage and playback errors.

**Improper care and handling**

Precarious storage or careless handling can cause damage to the tape cassettes, such as cracks and breakage, or can result in creased or twisted tape. Tapes kept in dusty, dirty conditions will accumulate particulate matter. This tape debris can interfere with the signal being read by the playback deck, resulting in dropout - the loss of magnetic particles. Debris on a tape can also be deposited on the tape path of a playback machine. Careless handling can also transfer oils and other chemical compounds to the tape. Ultraviolet light, particularly direct sunlight, is also damaging to videotape.

Poor housing also creates a risk for videotape. Paperboard tape cases create dust as they deteriorate, and paper materials tend to hold moisture and become more acidic with time. Tapes without cases obviously have no protection from particulate matter.

Tapes that are not re-wound after use, or those that are stacked horizontally instead of stored on edge, are subject to pack problems. Overtime, the tape pack can become uneven, exposing tape edges to possible damage, and making playback more difficult. Important information is held near the edges of
magnetic tape. In the case of open reel tapes, those not taped down at the end will become loose over time. These loose ends tend to fold over or deform. Loose tapes are also more susceptible to dirt and dust entering the tape pack.

Unintended recording

All cassette videotapes are designed to have a mode in which they can be recorded, and one in which they cannot. This is accomplished through a button or tab on the side or bottom of a tape that is detected by a video recorder. Often many tapes in a collection will come from the donor unprotected, and could be inadvertently recorded over. See identification pages to distinguish between tapes that are protected and unprotected. There is no such protection for open reel tapes.

Magnetic fields

A tape's signal (the information carrier) is represented on a tape by the arrangement of the magnetic particles into a particular pattern. Strong magnetic fields can affect the signal on a tape, causing it to become unreadable or adding to errors in playback. Common mistakes include leaving tapes on top of, or next to, a television, computer monitor, speaker, or microphone. Motors, transformers, generators, and industrial cleaning equipment are other devices that can cause demagnetization if a tape is within close proximity.

In recent years, there has been more discussion about transporting videotapes through airport security. It has been reported that airport scanning devices for checked baggage are damaging to videotapes. Biological decontamination devices are also reported to be damaging. Should these risks apply in the work of your organization, it is recommended that you keep current on these topics through such resources as AMIA-L, the listserv of the Association of Moving Image Archivists.

Unwise decisions about duplication

Videotapes often exist in multiples; many copies may be made from a single edit master or original recording. Several different organizations may hold the same works in different formats, in different generations, and the tapes may be in different conditions. An artist/producer may retain an original or
Master on professional stock such as BetacamSP, while an organization holds a viewing copy on a lower quality format such as VHS.

The image quality of an analog tape degrades each time a copy is made. Although digital tape copies are called “clones,” errors may occur as tapes are transferred, causing artifacts, or image distortions. If an inferior tape is re-mastered, or the preservation process is unprofessional, the result may be a version that is substantially degraded, and substantially altered from the artist/producer’s original work.

**Inadequate description or documentation**

Videotapes are typically minimally labeled, and often the case and tape contain different information. Also, tape labels commonly peel off and become separated from the tapes. As time passes, the label information often has less meaning to the preservationist or researcher in and of itself. Additional information can be gained from watching the tape, but as a tape ages, playback becomes more risky, making description even more difficult. Tapes without adequate description become low priorities for preservation, and are at increased risk of being lost through benign neglect. Also, one can waste precious preservation funds re-mastering tapes that are of lesser value, because time was not spent properly describing the tape at an earlier point in its life cycle.
Condition Assessment

Background

In general, this guide is oriented toward an initial, external examination of videotapes to notice significant physical problems, or to identify classes of tapes that can be assumed to be at risk. Understanding risks to videotape will provide helpful background. Once you have finished your inspection, you can take appropriate conservation actions. Since on a practical level inspection and simple conservation actions (such as re-housing) are often done at the same time, you may find it helpful to read through that section before you begin.

If you have a large collection, treatment of the tapes will need to be done in stages. A thorough examination of the tapes will help determine priorities for preservation. Priorities often develop from an assessment of age, condition, and significance or value.

Identification of tape formats is the first step to condition assessment, giving clues to age and condition, as well as alerting you to special characteristics of a particular type of tape. For example, if you have 2” quadruplex tapes in your collection, they can be assumed to be in poor condition simply due to age. Also, the fact that 2” equipment is rare will cause this format to be a high priority for preservation, regardless of condition. An initial inventory of your collection by tape format will also be helpful in creating an overall plan and budget for preservation projects.

External inspection is valuable for many reasons. For example, the inspection will tell you whether you need to hire an expert in disaster recovery, or a vendor with more general experience. Inspection will allow you to gauge whether or not the materials have particular damage, such as mold, that may be a health hazard for staff. You may discover vinegar syndrome, which has the potential to put other works at risk of contamination.

Newer tape formats with no external evidence of damage can be screened to give further information about condition. You will want to use a well-maintained and calibrated machine for this purpose. However, it is difficult for the untrained eye to make sense of errors common in tape recording and playback. Also, older recordings may look “soft,” or have glitches that were common given technological limitations of the time. Furthermore, some effects that mimic tape problems may represent purposeful actions by an artist. A media specialist and/or discipline specialist will be helpful to consult with during the inspection process.

However, problems with older tapes will not always be evident from an external visual examination. Playing back older tapes could cause further damage, especially if they are not cleaned first, or they are played back on a machine that is not well maintained. If there is a question about condition, err on the side of caution and do not play the tape until it is re-mastered. Tape condition should be evaluated and addressed during transfer to a contemporary medium.

As noted in the Risks section, there have been few scientific studies concerning videotape longevity. Best practices have evolved primarily from practical experience, advice from engineers and other technical experts, and from manufacturers. For more information, see the Resources and Sources sections.

Set-up for inspection

Determine how you will record the information you gain from the inspection. The best approach is to enter it into a field in a computer catalog or collection management system. Record at a minimum the
date, tape number, tape title, format, tape brand, conditions found, and the name of the person recording the information.

Work on a clean, dry surface. Covering a table with clean paper is a good approach. Lint-free cotton gloves are used by some preservationists, both to protect one's hands and to protect the tape itself. However, some people find it difficult to work with gloves on, and are concerned with transferring debris from the gloves to the tape as they handle the cases and other materials. Let common sense guide you. Gloves are not necessary when handling cassette tapes, because the tape surface is not exposed; however, open reel tapes are very vulnerable. Nevertheless, gloves and facemasks should be on hand for handling problematic tapes such as those with mold. In any case, your hands should be clean and dry before handling videotape, and free from hand lotions and oils. Wash your hands as frequently as needed to maintain cleanliness.

Examine the tape case

Tape cases will give clues to tape condition. If the case is dirty, use a lint-free cloth to wipe the surfaces of the tape case before opening it. This will avoid further contamination of the tape with particulate matter. The inside of the case may also have debris and dirt that can be removed with a lint-free cloth or through the use of compressed air.

A tape case may be bent or broken, suggesting rough handling. If the case looks water-damaged, you will want to check carefully for mold or evidence of water damage on the tape. Also, note the condition of any tape or case labels, and secure the labels if necessary. Note the presence of paper materials that will need to be organized during re-housing.

Examine the tape

Work with gloves and possibly a facemask if a tape looks or smells suspicious. Examine the tape at arm's length first, rather than bringing it up close to your face. Waving your hand over the tape can help the odor come to you; don’t bring the tape up to your nose.

If the tape is damp or wet, contact a professional immediately. See also “Video Preservation Facts Sheets” listed in Resources. Before picking the tape up, look for signs of mold. There are many types of mold; small white or brown strands or filaments have been reported as signs of mold on videotape. If you think the tape is moldy, it should be isolated immediately and treated by a professional.

Older tapes often have a pungent smell that has been reported to be associated with binder
deterioration. You may smell a vinegar-like odor that is typically associated with vinegar syndrome. White or brown powder has additionally been reported as a sign of binder deterioration. Also look for particulate matter or evidence of improper handling on the tape and reel, such as dirt, dust, or fingerprints.

If there is no evidence of mold, pick up the tape and look for cracks or breakage in the cassette or reels. With cassettes, check to see if the any of the tape has come out of the cassette, or if the tape is loose around the reels. Over time, open reel tapes may become loose on the reel and begin to deform. If the deformation is not severe, securing the tape around the reel(s) will help prevent further damage.

By holding the tape at a 45° angle and looking at the horizontal surface of the reel, you may discern an uneven pack, and note edge damage. Proper storage can help prevent further damage to the tape pack.

Again, carefully record the problems you see. They will determine the conservation actions you will take.
Conservation Actions

Identifying tape formats, understanding risks, and doing a condition assessment are recommended before taking conservation actions. On a practical level, inspection and simple conservation actions (such as re-housing) are often done at the same time.

As noted in the Risks section, there have been few scientific studies concerning videotape longevity. Best practices have evolved primarily from practical experience, advice from engineers and other technical experts, and from manufacturer literature. For more information, see Resources and Sources sections.

The following actions are directed toward those new to videotape preservation. They are written with the assumption that your organization does not have in-house technical expertise or the necessary equipment to do videotape cleaning or re-mastering. Before you begin a preservation project, you will want to consult others who have more experience – such as members of the key conservation/preservation organizations.

Documentation

Documenting preservation actions is essential to maintaining a history of the tape and any copies that may be generated through re-mastering. Creating a catalog or tracking the tapes in a collection management system is often the first step in a preservation project. However, often collection management systems will not have the necessary fields to describe and manage videotapes. Organizations such as Independent Media Arts Preservation and the Association of Moving Image Archivists can be helpful when developing descriptive tools.

Information recorded during a preservation project often includes such things as condition, storage history, description(s) of conservation/preservation actions, dates and individual(s) responsible, vendors, numbers of preservation master(s) and/or copies generated, tape brand(s), and technical data about the transfer. Using authority lists or glossaries for key terminology (such as tape format), even if only internal to your organization, is strongly recommended. Standardizing the type of information you collect and how it is recorded will make it much easier to manage the tape collection. Standards would ideally be developed prior to beginning a preservation project.

Maintaining/improving intellectual description

Tape labels may fall off of the cases or tapes during handling. It is important to keep the labels with the tapes, as the labels contain important information. In fact, they may represent the only available information about a tape. The label may also provide additional clues to a researcher about the time period or context in which the tape was recorded. Some repositories are scanning labels to provide a record. Paper conservators may be helpful in re-attaching labels.

You may find papers in tape cases that will help with identification and/or technical requirements. Remove the papers and store them in associated file folders. However, critical information from the papers should be entered into the computer catalog, and a note should be made in the record telling the location of the paper files. If tapes are re-housed, maintain original labels. Some organizations are starting to scan case labels or other pertinent information, creating digital image files.
As noted above, cataloging or recording tape information in a collection management system is recommended. This will often involve the recording of information from the labels, and re-numbering of the tapes with a local number.

When a tape is re-mastered, tape labels should include at a minimum the title, artist/producer, duration, original record date (if available), record date of the new tape, and the generation or type of material (such as “preservation master” or “viewing copy”). Additionally the label should indicate whether the tape is color or black and white, and should include any key audio elements; i.e., “sound, audio channels 1 & 2.”

**Videotape handling and housing**

Familiarize yourself with risks to videotape and follow common sense – treat videos gently and protect them from hazards from the environment, machines, and people.

Inert polypropylene plastic cases are available for most video formats, and should be kept on hand, especially to house those tapes that are without cases altogether. Some repositories temporarily re-house tapes in cardboard archival (acid-free) boxes until plastic cases are available, or if they are unavailable altogether (such as for 2” quadruplex or 1”). As noted above, original labels from cases may be important to maintain. For suppliers, see the Resources section.

Plastic bags inside of tape cases are not recommended. In the case of open reel tapes, if the tape is loose, gently re-wind the tape onto the reel and secure the tape ends with a small piece of acid-free, removable tape.

Tapes should always be played on well-maintained, clean, and calibrated machines that are of good quality. Schedules for cleaning and maintenance should be established with the advice of a tape engineer, reputable vendor, or other video expert. Never play a tape that is cold, wet, dirty, or contaminated (such as with mold).

It is recommended that after playing videotape, it is fast-forwarded to the end and then re-wound to the beginning. It is thought that this practice results in a more consistent wind and thus, a more stable tape pack. Newer tapes can be properly re-wound; however, re-winding can be risky for tapes thought to be in poor condition. When a tape is re-mastered by a vendor, all tapes should be returned to your organization re-wound to the beginning.

**Addressing molds and vinegar syndrome**

Tapes suspected of being moldy should be placed in plastic bags and sealed for evaluation by a professional. Conservators or other experts may be helpful in identifying a mold, or more importantly, in giving advice as to the environmental conditions – cool and dry – that will cause mold to become dormant.
A vinegar-like smell indicates that it is likely that active deterioration is occurring with acetate-based tapes. Isolating these tapes until they can be re-mastered is wise, and recommendations for cool, dry storage (see below) should be followed to slow the deterioration.

### Storage

Generally accepted practices call for tapes to be stored upright on metal shelves in the dark, away from magnetic fields. Clean, secure, cool, dry storage is essential. Standards for the storage of videotape are set by the International Standard Organization for Standards (ISO 18923:2000) and are summarized in "Videotape Preservation Fact Sheets" (http://www.amianet.org/11_Information/Information.html) as follows:

"Acceptable extended-term storage conditions for polyester-based magnetic tape, such as videotape, are: 20°C (68°F) and 20-30% RH; 15°C (59°F) and 20-40% RH; or 10°C (50°F) and 20-50% RH. The best long-term storage temperature is approximately 8°C (46°F) (never below) and 25% RH. Humidity variation should be less than ±5% RH and the temperature variation should be less than ±2°C (±4°F) within a 24-hour period."

The "Videotape Preservation Fact Sheets" and other resources can provide detailed information about other specifics of tape storage. For example, tapes coming out of cold storage must be brought to room temperature slowly to prevent condensation and should never be placed on machines when they are cold. Use common sense and plan accordingly – avoid extremes of temperature and humidity when tapes are not able to be in proper storage (hot summer days are not the best time to ship large quantities of your collection). Learn as much as you can about your existing storage conditions - some preservationists think they are maintaining proper conditions, only to find out that the HVAC is turned off over the weekend. Regular monitoring of temperature and relative humidity is essential.

Ideally an organization will have two copies of any important videotape, and the two will be stored in separate locations. If funds allow, it is recommended that when re-mastering is undertaken, two preservation masters are created to allow for geographic separation.

### Tape cleaning and re-mastering

Tape cleaning is most often recommended with open reel video formats, due to problems with sticky shed syndrome. Tapes with sticky shed deposit debris on the tape path of playback machines, making transfer difficult to impossible.
Methods of open reel tape cleaning have been developed by re-mastering facilities such as Vidipax and the Bay Area Video Coalition. For cassettes, many facilities use RTI (Research Technology International) machines and other tape conditioning units. Tape cleaning machines typically include a combination of polyester pylon rollers, vacuum chambers, and burnishers. The use of cleaning fluids is not recommended.

Re-mastering is a process of making a duplicate of a tape onto a contemporary videotape format. The resulting tape is often referred to as a preservation master. The preservation master is stored in secure, climate-controlled storage and not played except for emergencies. Viewing copies or duplication masters are also usually struck at the time of re-mastering.

Choosing a tape format for re-mastering can be challenging, as there is no tape format that is the agreed-upon best solution. Digital tape, though not free from problems, is here to stay, supplanting analog tape formats in the broadcast and professional production sectors. Factors to consider in choosing a preservation tape format include:

- A large, strong tape stock will allow for the storage of more information; a tape stock must be physically durable. For example, the tape format Digital Betacam is a thicker, bigger, more durable stock than Hi8 or MiniDV.
- A tape stock whose related playback machines are professional, durable, and reliable is recommended. The example above is also an illustration of this point.
- A tape stock whose related playback machines are the least subject to obsolescence is recommended. Tape formats that have the greatest market penetration in the broadcast and professional production sectors are thought to be less subject to obsolescence.
- A tape stock that allows for the greatest amount of information to be maintained is recommended. Digital tape formats involve compression of data. The lowest amount of compression is preferred when re-mastering. For example, D1 is an uncompressed digital video format, and Digital Betacam has a low rate of compression.
- An organization’s resources and workflow must be considered. Some tape stock – such as D2 – is very expensive. Playback decks for D2 are out of reach for most organizations in the non-profit sector.

At the time of this writing, many people agreed that Digital Betacam, a 1/2” format, is a good choice for a preservation master. It is a digital format with low compression that is thought to have good market penetration. The decks are professional quality and thought to be reliable. Some organizations distrust digital formats or find them too expensive, and choose BetacamSP. Although BetacamSP shares many of the same advantages as Digital Betacam - in terms of reliability, durability, and market share - production of the decks is expected to be discontinued within the next 3 – 5 years.

Following the logic of the factors above, DVD is not considered to be a good media for preservation. Video stored on DVD is likely to be highly compressed and the durability of optical media is unknown at this time.

With all re-mastering projects, quality control must be maintained. Real-time viewing of re-mastered tapes, duplication copies, and screening copies on playback equipment kept in optimal working condition is the best way to ensure quality.
Locating a high-quality version

Since videotapes often exist in multiples, the tape your organization owns may not be the highest quality version of the work you are seeking to preserve. It will be important to determine if another organization or individual holds an original, master, or high-quality copy before you begin a preservation project. An analog tape degrades each time a copy is made, and errors may occur on digital tape copies. It is important to re-master from the highest quality tape to ensure a faithful rendering of the original video work.

Preservation planning and budgeting

Ongoing videotape care and preservation requires planning and dedicated line items in your organization’s budget. Costs for videotape preservation will vary widely according to the size of your holdings, their condition, and your existing resources, such as storage vaults. Typical budget items will include: shelving, archival cases, labeling supplies, vendor fees for cleaning and re-mastering, and storage fees. Project costs will vary widely, depending on formats, tape condition, and the number of copies requested. Funds may also be needed for professional collection surveys or development of descriptive databases.
See also the Sources section, which lists resources consulted when creating this website.

**General Video Preservation Resources**

The website of the Association of Moving Image Archivists, a non-profit professional association that serves the field of moving image archiving, contains extensive information on video preservation issues. Under the heading “AMIA IPublications and Resources” relevant sections include: “Storage Standards and Guidelines for Film and Video” and “Videotape Preservation Fact Sheets.” AMIA also holds an annual conference (see below) and has a listserv, AMIA-L, which is an excellent source of information on media preservation. http://www.amianet.org; Email: amia@amianet.com

**Basic Inspection Techniques to Sample the Condition of Magnetic Tape** is a checklist prepared by the vendor Spec Brothers that is helpful in diagnosing tape problems. http://www.specsbros.com/whitepaper.html

**Bay Area Video Coalition (BAVC)** is a non-profit media arts center that offers video re-mastering for 3/4” U-matic, 1/2” open reel, and other formats. The “Video Preservation Resources” section of the BAVC website contains a comprehensive glossary of video preservation terms, links to a wide range of organizations active in preservation, sections on cataloging and hardware, and an events archive that charts the history of conferences and panels on independent media arts preservation. In 2003, BAVC produced an interactive DVD, *Playback: Preserving Analog Video*, that contains information on analog videotape composition and deterioration, as well as a case study that follows the preservation of a work of video art. An earlier monograph, *Playback: A Preservation Primer for Video* (BAVC, 1998), is also a very useful resource. The DVD can be ordered from the website at http://www.bavc.org/classes/dvd/preservation.htm; questions about the DVD and monograph can be directed to presdvd@bavc.org.

**Conservation OnLine (CoOL)** is an on-line text library of conservation resources created by the American Institute for the Conservation of Historic and Artistic Works (AIC). The video preservation section offers information on standards, guidelines and best practices for video preservation, and bibliographic resources (including a large number of articles that exist on-line), among other preservation resources. AIC also operates a listserv that is an excellent source of information on conservation. http://aic.stanford.edu/pubs/porder.html

**Independent Media Arts Preservation (IMAP)** is a non-profit organization that serves the field of independent media, providing preservation resources, information, and training. IMAP distributes a cataloging template, which is based on national standards, through its website. http://www.imappreserve.org; Email: imap@imappreserve.org.

**MIC: Moving Image Collections** is a portal to moving image collections for educators, researchers, exhibitors, and the general public. It is also a resource for preservationists seeking to describe and maintain these kinds of collections. Additionally, it serves as a source for controlled vocabulary for moving image cataloging. http://gondolin.rutgers.edu/MIC/

**Video Format Identification Guide** – Created by Sarah Stauderman, this guide includes a wide range of formats and assigns obsolescence ratings to each one. http://www.paulmessier.com/VideoID/
Resources for Video Preservation

Video Preservation: the Basics – an on-line resource published as part of the Experimental Television Center’s “Video History Project.” This website provides a thorough overview of all aspects of video preservation. Included are preservation terms and definitions, preservation planning and management, handling of tape, storage guidelines, cataloging, cleaning and remastering, disaster planning, and links to additional preservation resources. ETC also published Reel to Real: A Case Study of BAVC’s Remastering Model by Luke Hones (2002), which details the process of re-mastering 1/2” open reel tapes. http://www.experimentaltvcenter.org/history/preservation/preservation.php3


Key Organizations

The Association of Moving Image Archivists is a non-profit professional association that serves the field of moving image archiving through information-sharing, local and national initiatives, publications, and professional development. Most of AMIA’s work is conducted through Committees (such as Conservation, Cataloging and Documentation, Digital Initiatives) Interest Groups (such as Digital Archives, Independent Media, Small Gauge Film), and Task Forces (such as Diversity, Local Television, Digital Issues). The list of committees and interest groups can be found at http://www.amianet.org/05_Committees/committees.html; Email: amia@amianet.com

The American Institute for Conservation of Historic and Artistic Works (AIC) is a national membership organization of conservation professionals dedicated to preserving art and historic artifacts. AIC devoted Volume 40 No. 3 (2002) of their journal to the issue of installation art preservation. Titled TechArcheology: Journal of the American Institute for Conservation, the publication can be ordered online at http://aic.stanford.edu/pubs/porder.html.

The Electronic Media Specialty Group (EMG) is a group within AIC focused on the preservation of electronic art and electronic-based cultural materials. EMG provides a forum for art conservators and related professionals to develop and maintain knowledge of new media and emerging technologies. Meetings of the EMG occur at the annual meetings of AIC. http://aic.stanford.edu/conspec/emg/

Independent Media Arts Preservation (IMAP) is a service, education, and advocacy organization that provides access to information about preservation practices and research. They offer workshops, one-on-one assessments, and technical assistance to the field; promote media preservation activities through publications, forums, and conferences; and distribute a cataloging template (see above). The IMAP website (http://www.imappreserve.org) also contains a full compliment of information resources on all aspects of media preservation. Email: imap@imappreserve.org

Vendors

Bay Area Video Coalition offers preservation services including cleaning and transfer from 1/2” open reel, 3/4” U-Matic, and VHS tapes. Location: San Francisco, CA. Phone: (415) 861-3282; http://www.bavc.org; Email: preservation@bavc.org

DCVideo Post provides transfer services from a wide range of formats including obsolete formats such as 2” quad, 1/2” open reel, and 1/2” Betamax. Located in Burbank, CA. Phone: (818) 563-1073; http://dcvideo.com/index.html; Email: david@dcvideo.com
SPECS Brothers, a video and audiotape restoration facility located in Ridgefield Park, New Jersey, offers services in archival restoration, disaster recovery and consulting on archival management, disaster planning, and other areas. The website also contains information on a range of preservation issues including sections on Disaster Recovery and Disaster Planning. Phone: 800.852.7732; http://www.specsbros.com/; Email: admin@specsbros.com

Vidipax offers video and audiotape restoration, as well as archival film to tape transfers. Vidipax can restore and transfer a wide range of obsolete formats including 2” Quad, 1”, 1/2” Open Reel, Betamax, and 3/4” U-Matic. They also offer disaster recovery, encoding, and a wide range of consulting services. The website also contains extensive resource links, and videotape and audiotape format guides. Their New York City location also houses the Vidipax Museum of Magnetic Recorders. Phone: 800.653.8434; http://www.vidipax.com/; Email: info@vidipax.com

Supplies

Gaylord (aka Gaylord.com) offers archival supplies such as gloves and acid-free labels, as well as a wide range of acid-free containers for videotapes, audiocassettes and reels, film and multimedia. Phone: (800) 448-6160; http://www.gaylord.com/; Email: customerservice@gaylord.com

Image Permanence Institute (IPI) at the Rochester Institute of Technology offers a downloadable preservation calculator for evaluating storage conditions at http://www.rit.edu/~661www1/sub_pages/8page20.htm. Phone: (716) 475-5199

University Products (aka archivalsuppliers.com) offers archival supplies such as gloves and acid-free labels, as well as a wide range of acid-free containers for videotapes, audiocassettes and reels, film and multimedia. Phone: (800) 448-6160; http://www.archivalsuppliers.com; Email: custserv@archivalsuppliers.com

Funding

Funding also may be available from local and statewide arts and historic preservation organizations.

Institute of Museum and Library Services
IMLS is a federal agency supporting U.S. museums and libraries providing funding for conservation projects, conservation assessments, and national leadership projects. http://www.imls.gov/

National Endowment for the Arts
The goals of the NEA Heritage & Preservation are to “assist, preserve, document, and present those artists and forms of artistic expression that reflect our nation’s diverse cultural traditions, and to conserve important works of art.” http://www.nea.gov/

National Endowment for the Humanities
The NEH Division of Preservation and Access provides grants to preserve and create access to humanities collections and to conduct preservation research and training. http://www.neh.fed.us/

National Historic Publications and Records Commission
NHPRC is part of the National Archives and Records Administration (NARA) and funds nonprofit organizations to preserve “documentary sources significant to the history of the United States.” It also funds archival preservation, processing, and the development of archival tools and standards. http://www.archives.gov/grants/index.html
Glossary

Definitions of tape problems – dropout, hydrolysis, sticky shed, and vinegar syndrome – are used with permission from the 1996 glossary “Video Preservation: Glossary of Terms” by Rebecca Bachman (http://palimpsest.stanford.edu/byorg/bavc/bavcterm.html). The glossary is a great resource for other terms you may hear as you become more involved with video preservation.

**ANALOG TAPE** ➤ Videotape that records a representation of a continuous electronic signal.

**ARTIFACTS** ➤ The distortion of a digital video image caused by such factors as errors, loss of information, or computer processes during decompression. Common artifacts are blocks of large pixels or the presence of jerky motion in playback.

**COMPONENT VIDEO** ➤ With component video, the luminance (black and white levels) and chrominance (color information) are transmitted as separate signals. The picture quality is superior to composite video.

**COMPOSITE VIDEO** ➤ All color, luminance, and synchronizing information is carried together as part of the same signal. Composite video was the norm until the early 1990s.

**COMPRESSION** ➤ A term for the process of reducing the size of a digital file, to help with storage or transmission, through a codec (a compression/decompression algorithm or formula).

**DIGITAL TAPE** ➤ Videotape that records a numerical representation of how an electronic signal changes over time. A digital recording is produced from a digital file that may be compressed or decompressed as part of the recording/duplication process.

**DROPOUT** ➤ Brief signal loss caused by a tape head clog, defect in the tape, debris, or other feature that causes an increase in the head-to-tape spacing. A dropout can also be caused by missing magnetic material. A video dropout generally appears as a white spot or streak on the video monitor. When several video dropouts occur per frame, the TV monitor will appear snowy. The frequent appearance of dropouts on playback is an indication that the tape or recorder is contaminated with debris and/or that the tape binder is deteriorating.

** Duplication Master (aka Dubmaster)** ➤ A tape that is used for creating viewing or reference copies. In the case of analog tape, a dubmaster is usually one generation off of an original recording, edit master, or preservation master. In the case of a digital tape, a dubmaster may be, for all practical purposes, a “clone” of the original or master tape.

**FORMAT** ➤ In video, a term used for the size, packaging, and sometimes the recording standard of a certain family of videotapes. Tapes that are the same format, though different brands or made by different manufacturers, will play back on the same equipment.

**GENERATION** ➤ A term, typically used with analog recordings, that commonly refers to the relationship between original or master recording and subsequent copies.

**HYDROLYSIS** ➤ The chemical process in which scission of a chemical bond occurs via reaction with water. The polyester chemical bonds in tape binder polymers are subject to hydrolysis, producing alcohol and acid end groups. Hydrolysis is a reversible reaction, meaning that the alcohol and acid groups can react with each other to produce a polyester bond and water as a by-product. In practice, however, a severely degraded tape binder layer will never fully reconstruct back to its original integrity when placed in a very low-humidity environment.
**PRESERVATION MASTER** ➤ The common term for a tape that is created through the process of re-mastering. Preservation masters are ideally only accessed when a duplication master is no longer useful for making viewing copies.

**RE-MASTERING** ➤ Copying a tape to a new, contemporary format, following standard television practices to ensure the best possible result. The new tape is commonly called a preservation master.

**STICKY SHED** ➤ The gummy deposits left on tape path guides and heads after a sticky tape has been played. The phenomenon whereby a tape binder has deteriorated to such a degree that it lacks sufficient cohesive strength so that the magnetic coating sheds on playback. The shedding of particles by the tape as a result of binder deterioration that causes dropouts on [video]tapes.

**VINEGAR SYNDROME** ➤ Characteristic of the decomposition of acetate-based magnetic tape where acetic acid is a substantial by-product that gives the tape a vinegar-like odor. After the onset of the vinegar syndrome, acetate tape backings degrade at an accelerated rate—the hydrolysis of the acetate is catalyzed further by the presence of acetic acid by-product.
Sources


Credits

The authors wish to thank Sarah Stauderman of the Smithsonian Institution Archives for inspiration - from her creation of the “Video Format Identification Guide” (see above) to her sage advice in all aspects of production of this guide. The authors also give great thanks to Laura M. Wiegand of the Texas Commission on the Arts, who gave invaluable feedback on the content and layout of the guide. Also, many thanks to Rebecca Bachman for use of her excellent “Video Preservation: Glossary of Terms” and Suzanne Quigley of the Whitney Museum of American Art for her thoughts on scope and presentation.

Many thanks to Chris Lacinak of Vidipax, Daisy Pommer and Leslie Rosen of Thirteen/WNET, Nancy Cricco and Melitte Buchman of the Elmer Holmes Bobst Library, New York University, and Peter Bauce of Photo/Video for the loan/donation of videotape formats.
Sources and Credits

Project Coordination ➤ Laura M. Wiegand, Director for Programs & Technology, Texas Commission on the Arts

Research, Information Design, Writing, Editing, and Photography ➤ Mona Jimenez and Liss Platt, Materia Media

Advice and Consultation ➤ Sarah Stauderman, Preservation Manager, Smithsonian Institution Archives

Layout of PDF ➤ Liss Platt

Copy Editor ➤ Claudia B. Manley

Disclaimer

The information provided in the *Videotape Identification and Assessment Guide* has been prepared carefully using reputable sources and the advice of respected experts. However, the Guide is for general informational purposes only and should not be treated as a substitute for the advice of a qualified professional. The authors, advisors, and the Texas Commission on the Arts are not liable for the validity of the claims published in this guide and cannot be held responsible for any damage or loss that is incurred as a result of the misuse of the information presented herein.