Creating collections and evaluating viewpoints: Selection techniques for interface design

by

Adrian Joseph Secord

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September, 2010

Denis Zorin
New York University

Andrew Nealen
Rutgers University

Dedication

To my parents, my little sister, and my big sister.

Acknowledgements

First, I would like to thank my advisor Denis Zorin, who sharpened my approach to academic research and supported me while I found a path.

The time I spent at Adobe's Creative Technology Lab was inspiring. I would particularly like to thank David Salesin, along with Mira Dontcheva, Wilmot Li, and Holger Winnemöller, who have generously extended their time, expertise, and friendship over the last four years. Mira, Wil, Holger and I collaborated closely on work that became Chapters 2 and 3.

Many of my colleagues at NYU became good friends: Elif Tosun, Harper Langston, Matt Grimes and Yotam Gingold all deserve more than I can express here. I was also lucky to work with the following great lab-mates at NYU, all of whom spark with wit and charm: Chris Wu, Ilya Rosenberg, Philip Davidson, Jason Reisman, Jeff Han, Eitan Grinspun, Denis Kovacs, Ian Spiro, Robb Bifano, Ayse Erkan, Raia Hadsell, Meghan Hartley, Lyuba Chumakova, Shravan Veerapaneni, Ashish Myles and Kenshi Takayama. You'd be lucky to work with them, too.

Ken Perlin, Olga Sorkine, Margaret Wright and Rosemary Amico all work very hard to make the lab and the department a supportive environment. I value their efforts and appreciate that the task must occasionally seem fruitless. It is not.

I am indebted to Adam Finkelstein for his generosity in time, expertise, and support. I'm glad that we finally got to work together after all these years.

Andy Nealen walked the difficult line of being a friend and co-advisor, which turned out to be both more challenging and more rewarding than we expected, but our friendship is deeper for it. My work with Andy and Adam forms the basis for Chapter 4.

Vicki Weafer supported me unconditionally with love and good humor, despite the weight of her own dissertation.

My family in Canada—Mom, Dad, Andrea, and Jenn—supported me from afar with all their love and only the occasional "Get on with it!" I'm lucky to have both of these things.

Finally, while finishing a PhD is generally considered the end of a journey, for Kate and myself it is the beginning. Kate's fierce love and energy contributed more to this dissertation than one might expect given our brief time together.

Thanks!

Preface

Some of the material of Chapters 1 and 3 is to be published in the proceedings of the 23rd annual ACM symposium on User Interface Software and Technology (UIST 2010) [SWLD10], co-authored with Holger Winnemöller, Wilmot Li and Mira Dontcheva.

At the time of writing, a previous version of Chapter 4 is in submission to the ACM Transactions on Graphics [SLF+], co-authored with Jingwan (Cynthia) Lu, Adam Finkelstein, Manish Singh and Andy Nealen.

Abstract

In computer graphics and user interface design, *selection problems* are those that require the user to select a *collection* consisting of a small number of *items* from a much larger *library*. This dissertation explores selection problems in two diverse domains: large personal multimedia collections, containing items such as personal photographs or songs, and camera positions for 3D objects, where each item is a different viewpoint observing an object. Multimedia collections have by discrete items with strong associated metadata, while camera positions form a continuous space but are weak in metadata. In either domain, the items to be selected have rich interconnections and dependencies, making it difficult to successfully apply simple techniques (such as ranking) to aid the user. Accordingly, we develop separate approaches for the two domains.

For personal multimedia collections, we leverage the semantic metadata associated with each item (such as song title, artist name, etc.) and provide the user with a simple query language to describe their desired collection. Our system automatically suggests a collection of items that conform to the user's query. Since any query language has limited expressive power, and since users often create collections via exploration, we provide various refinement techniques that allow the user to expand, refine and explore their collection directly through examples.

For camera positioning, we do not have the advantage of having semantic metadata for each item, unlike in media collections. We instead create a proxy *viewpoint goodness* function which can be used to guide the solution of various selection problems involving camera viewpoints. This function is constructed from several different attributes of the viewpoint, such as how much surface area is visible, or how "curvy" the silhouette is. Since there are many possible viewpoint goodness functions, we conducted a large user study of viewpoint preference and use the

results to evaluate thousands of different functions and find the best ones. While we suggest several goodness functions to the practitioner, our user study data and methodology can be used to evaluate any proposed goodness function; we hope it will be a useful tool for other researchers.

Contents

List of Figures

List of Tables

Introduction

This dissertation describes approaches to the *selection problem*, which is the problem of selecting a *collection* of *items* from a much larger *library* of items. The original motivation for exploring the selection problem came from a colleague who, after returning from a vacation in Alaska, found she had well over 1000 photographs to sort through, no time to do so, and yet still wanted to quickly upload a set of highlights to friends. Most users tend to solve the selection problem manually and tediously—see Chapter 1 for evidence of this in the domain of selecting photo highlights. Our exploration of improved semi-automatic interfaces to such problems lead to Chapters 2 and 3.

A different domain: when interacting with 3D objects using a 2D interface, most users have difficulty effectively selecting an appropriate camera viewpoint using, for example, the standard virtual trackball device. In this domain, we consider the items to be the camera viewpoints, the collection to be the set of useful viewpoints for a particular task, and the library to be the set of all possible viewpoints. Our approach to this problem lead to the improved interfaces described in Chapter 4.

What characterizes the selection problem? In this dissertation, *items have rich interrelations.* For example, when selecting a set of highlight photographs to share with family from a recent vacation, a user might want to choose photographs that show each family member in a pleasing light, that tell a story of the various places visited, and that include or do not include specific family members, etc. Not all of these qualities are simultaneously achievable; for example, if there are no pictures of John at the waterfalls, then it's difficult to show every family member at each location visited. Problems in which the items have no interrelations, or that the relationships are irrelevant, such as "select 30 photos of your vacation with at least one photo of John," are less interesting, since a simple filtering operation would

Table 0.1: *Examples of different selection problem domains used in this dissertation.*

Domain	Item	Collection	Library
Photo highlights	Photograph	Photos to be shared	Personal photos
Music playlist	Song	Playlist	Personal music
Viewpoint selection	3D camera position	Views of an object	All possible views

suffice to solve the problem.

The selection problem is most interesting when *the collection size is large enough to make manual selection difficult*. In the case of a user seeking a collection with a single item, the user could either manually scan the entire library, or, if the library is too large (such as when searching the web), use some form of ranking. However, when the collection size becomes larger, say, 30–50 items, then the relationships between objects tend make these approaches tedious and difficult.

Finally, our selection problems involve *a large library of items*. If the library of items is fairly small, then the manual approaches already used by many users (see Chapter 1) will continue to suffice.

Each of the following chapters describes an approach to solving the selection problem in one or more domains, listed in Table 0.1. Chapter 1 describes the user research that formed the basis for Chapters 2 and 3. Although it was chronologically interspersed with the design and evaluation of the systems described in Chapters 2 and 3, it is consolidated into a single chapter for clarity. Chapter 2 applies a mixture of traditional and new user interface elements to aid the user with selecting a set of photo highlights from a large collection of personal photographs. Chapter 3 continues this work with the introduction of a simple query language to help create an initial collection, and several new interface techniques for refining the collection. The domains of Chapter 3 are collections of songs for musical playlists and a revisitation of collections of photo highlights. Chapter 4 breaks conceptually with the previous chapters to explore the fundamentally different domain of camera positions for 3D objects, requiring different techniques than those used in the

previous chapters.

Chapter 1

User behavior around creating collections

To understand users' behaviors associated with creating and sharing collections of items, we first surveyed prior research on practices around creating collections [BMH06, FKP+02, KSRW06, VGD+05, SM09, SKL+08b] and then conducted three studies: a broad online survey of users of digital photography, a set of in-home interviews with the same photography users, and a set of interviews with creators of music playlists. Here we describe the studies and draw some conclusions about user behaviors.

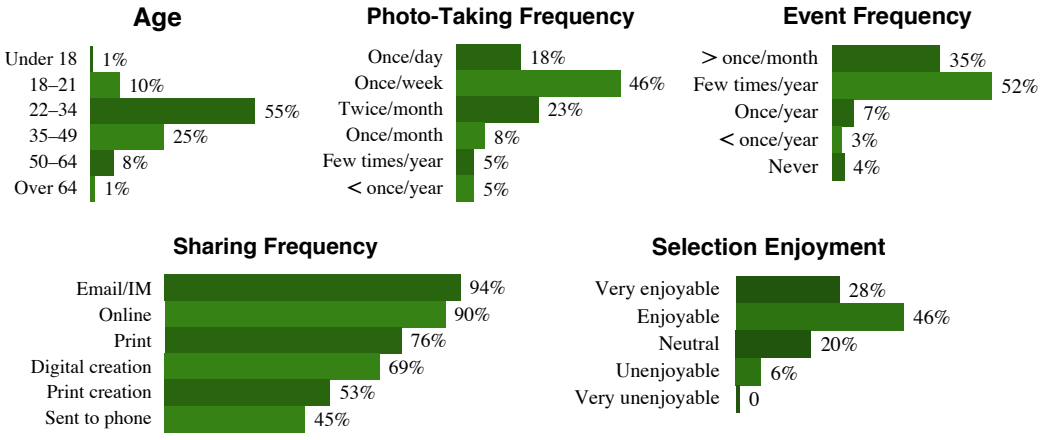


Figure 1.1: *Summary of our respondents' photo-sharing habits (values rounded).*

1.1 SURVEY: DIGITAL PHOTOGRAPHY

We recruited participants for our study of digital photography through Craigslist [Cra09], a large online forum. The survey was described as part of the preliminary screening process for in-person interviews. Our advertisement stated that we were "looking for casual photographers to learn more about their experiences with digital photographs." The survey asked questions about "occasions, trips, or events" during which the respondent personally took more than 100 photographs. Of the 913 people who responded, 40% were male, 60% were female and they had a wide range of occupations. The respondents were mostly casual photographers (76%); the rest were either serious or professional photographers. Figure 1.1 summarizes the data from the respondents. From the survey, we gained insights into how often and by what means respondents shared photos from such events and by which process they selected the shared collections.

Our survey reinforces findings from previous work that sharing photos is a nearly universal activity. Among the respondents, 94% emailed or instant-messaged photos to friends or family, and 90% uploaded photos to an online sharing site. In addition, nearly all the respondents who reported *not* sharing their photos planned on doing so but cited a lack of sufficient time. Furthermore, our survey results indicate that users very much enjoy the process of selecting collections of photos; 74% found the selection task enjoyable or very enjoyable. Previous work and our field studies suggest that users intermingle the task of selecting photos with reminiscing, exploring their photos, and storytelling.

We were particularly interested in "occasions, trips, or events" during which the user personally took more than 100 photographs, since we expected that creating collections from such large sets would be more difficult. We allowed the users to determine what constituted an occasion, trip, or event, but we will refer them all as *events* in the following. Most users (87%) photographed a large event at least a few times a year; and a significant portion (35%) did so once a month or more. Typical large events included: bridal and baby showers, weddings, vacations, parties, cruises, reunions, new babies/pets, and inaugurations. From the most recent large event,

55% personally took 100–200 photos and 19% took 200–300 photos.

Many respondents also found the sharing process *tedious and sometimes difficult*, particularly with larger libraries. In terms of difficulty, 41% of respondents were either neutral or found it difficult to very difficult to select a collection to share. As for the time it took, 58% were either neutral or found the process slow to very slow.

In addition to the expected difficulties with software, slow file transfers, etc., *respondents called out the selection task as a difficulty*: "The [software process] is not the issue…it's more deciding what photo you want to share with friends and family that capture the moment best and communicates this moment to others." Further quotes from our survey respondents:

> *It takes a long time for me to decide.… I enjoy picking them out but it is neither an easy nor quick process.*
>
> *Picking photos out require* [sic] *a decent amount of time. Not all photos I want to share with family, some just with friends, some to only social networking.*
>
> *Its* [sic] *always enjoyable, very enjoyable to share my work.*
>
> *I was on a group trip and took 1800+ pictures…choosing which to upload is incredibly daunting, especially when 30 people are counting on your photos.…*
>
> *…I needed to choose the correct selection to show people what my true experience had been…*
>
> *There were so many beautiful pictures, I only shared those that went with a conversation about the activity we took part of.*

1.2 INTERVIEWS: DIGITAL PHOTOGRAPHY

To gain a deeper understanding of how and why people select photos for sharing, we visited the homes of seven non-professional users who had photos to share

from a recent event or trip. For each home visit, we first questioned the users on their general behaviors regarding digital photography. We then had the users walk through their photo library to examine their organization, tool usage, etc. Finally, we asked the users to perform a basic selection task using photos from a recent event. Based on our observations, we gained several insights into the selection task that directly informed the design of our interface.

We found that users' motivations for sharing match those found in [FKP+02, KSRW06], namely for capturing memories and life events, sharing with friends and family, and creating keepsakes such as photo books. Users would typically form a collection to share using the operating system's file browser, making two passes over their library: the general intent of the first pass is to identify the good photos and weed out the bad ones. The second pass refines and culls the collection to the desired size and ensures that all the goals are met.

Users brought definite collection goals to the selection task. Depending on the intended audience and medium, users might: select the number and orientation/size of photos for a particular medium (e.g. for email, web or print); exclude photos that contain people unknown to a recipient; or include at least one photo from some group (e.g. at least one photo that contains Rich). "Balance" goals were also common: balancing the number of photos of each person and place, balancing group and individual shots, or balancing photos throughout time. All of these goals were loose and adjustable, but in general *users had clear goals for the collection*.

In terms of choosing individual photos, users took into account several criteria, including technical qualities like focus and lighting, whether the photo contained people or key landmarks, whether the photo would resonate with the intended recipient, or whether the photo had some underlying story, personal meaning, or inside joke. Although some of these qualities could be computed automatically using existing algorithms (e.g., proper focus, good lighting, the presence of people), many of the criteria are highly subjective and involve a deep semantic understanding of the people and places depicted in the photographs. It is hard to imagine how any automatic algorithm would be able to reliably evaluate photos based on such subjective criteria. In other words, *many of the qualities that users look for in a*

photo are not easily computable, which indicates that a purely automatic system for selecting collections of photos is unlikely to succeed.

Finally, at the level of individual photographs, we observed a few common ways in which users added and removed photos from their collections. Nearly universal behaviors include choosing a single best photo from a series of near-duplicates, and including remarkably good photos or excluding remarkably bad photos irrespective of their goals for the collection. Users often search out and select a photo of a particular person, place or event that fits with the rest of the collection. Finally, users would occasionally look for a photo that was complementary to other particular photos (say, for a page layout in a photo book). Users certainly showed variation from individual to individual, but they typically made *photo collections using a relatively small set of strategies*.

1.3 INTERVIEWS: MUSIC PLAYLISTS

Although digital photography is possibly the most common domain for creating collections, many other exist such as creating musical playlists, choosing investment portfolios, etc. To balance the work on digital photography, we also interviewed three users who regularly create collections of songs into playlists. We asked the users 25 questions about the size of their music libraries, how frequently they create playlists, what they use their playlists for, etc. The users' collections ranged from roughly 5000–10000 songs, and the users spent anywhere between five minutes to 1.5 hours making a playlist. They made playlists every day, one a week, or once every couple of weeks. They described many motivations for making playlists: for fun, for sharing with friends, for entertaining at gatherings, for teaching, and for motivation during exercise. In addition to these general statistics, we asked questions about how they created their collections, and what their motivations were while doing so. Many of their responses echoed those of the users who were interviewed about their photography collections, and we summarize key points for our research next.

We identify five key observations that inform the design of digital tools for creating playlists and photo collections. Although we focus on music and

photographs here, we believe these observations may also apply to working with other types of collections.

Users have specific goals when creating collections. Collections are often created with respect to events [BMH06], situations [CJJ04], and people [Vie00]. For example, playlists are often situation-specific (e.g. for driving, entertaining, exercising), and photo sharing can be person-specific (e.g. photos for friends, parents, colleagues). Tools should allow users to achieve their goals by allowing them to describe their preferences and aiding them in understanding when a collection fits or does not fit these goals.

People satisfice when creating collections. Bentley et al. [BMH06] performed an in-depth analysis of user behavior with photographs and music and found many similarities in how users work with personal photograph and music libraries. They proposed that when working with photographs and music, users are often satisficing, i.e. they are not looking for a specific item and will often stop when they find a "good-enough" photo or song instead of continuing to search for the best item. They claimed that the reason people satisfice is because they have too many items to evaluate them all. This satisficing behavior implies that automation has a place in helping users create photo and music collections.

Users refine collections iteratively by exploring related items. Users start with an initial rough collection and then iteratively grow and shrink the collection as they encounter new items. For example, when creating a playlist, users may start by adding many potential songs to the playlist and then perform a second pass winnowing down the collection and replacing specific songs to make them fit better with the overall collection. They may look for different songs from the same album or a related song by the same artist. When working with photos users often look at sequences of photos and choose one from multiple similar shots. This iterative process is not merely a consequence of todays manual tools. As users create collections they stumble upon forgotten items that remind them of past events and

lead them to explore related songs or photos. To support this iterative process, tools must support users' needs for browsing along varied dimensions and inter-item relationships.

Adding an item to a collection can be just as much about the item's fit in the collection as the item's individual quality [HG09]. For example, parents often want all children to appear in a set of photographs equally and may include a specific photograph to satisfy this requirement even if the photograph is not of high quality. A music playlist may have a maximum duration, and so a song may be included because it is just the right length. Tools for creating collections should make it easy for people to manage these inter-item dependencies and overall collection requirements.

Collections that are shared with others can be highly personal [VGD+05, SM09] and specialized. They often tell stories and include items that may seem unrelated to the casual observer. For example, music playlists are often created as gifts and include songs that are meaningful to the person giving or receiving the gift, such as the classical "mixed tape." Similarly, photos often tell personal stories of travels and events. It is hard to imagine that a fully automatic tool for creating collections could incorporate all of the subjective criteria that may be part of selecting items. Tools for collections need to be flexible to shifting user concerns and preferences.

The following two chapters examine two different proposed systems for helping the user semi-automatically create collections given the above observations.

Chapter 2

A goal-driven selection interface for personal photos

2.1 INTRODUCTION

Photographs have become one of the most popular ways to capture and document important life-moments. Memory-keeping and *sharing with others* are the two most important reasons for taking pictures [Rod99, FKP+02]. Many share photographs from social events with friends who did not attend, or, after a group vacation, everyone shares pictures from the trip with travel companions. Similarly, parents often send pictures of young children to remote family members. The availability of digital cameras has made photography increasingly convenient and cheap; photographers now take pictures much more readily and frequently. As a result, today's photo-sharers are often faced with the challenge of choosing a few good photos from hundreds or even thousands of captured images. One of the key tasks in the photo sharing process is *selecting an appropriate collection of photos* to share based on the relevant criteria.

To address this problem, we present a novel approach for selecting collections of photographs from a personal photo library: our system provides *automatic suggestions* from *collection preferences* and *direct selection tools* at the level of individual photographs. In contrast to typical photo browsing interfaces that require a user to manually locate and select every photograph for a collection, our system allows users to create collections semi-automatically. First, the user specifies rough preferences for a candidate collection. For example, the user might ask for a collection of 30 photos, mostly of people, and with roughly the same number of photos of each person. We encode collection preferences as constraints and employ

a constraint solver to automatically generate candidate collections of photographs that match these preferences. Second, we provide a number of direct manipulation tools that enable users to ask for similar photos, replacement photos, or photos of a specific place or person. Users can alternate between asking the system to automatically generate new candidate collections and directly selecting photos of interest. One key difference between our work and existing systems is that our system can be used in a manner that ranges from fully automatic to fully manual, and that the user has a shareable collection at every stage in the process. He can stop after the initial collection suggestion and share, spend a few minutes selecting favorite photos, or spend hours creating that perfect shareable collection. The effort required to select a shareable collection scales with the size of the collection, not the size of the library.

As discussed in Chapter 1, we found that users typically approach the selection task with several *collection goals* that are semantically rich and influenced by the target audience and sharing medium. For example, one might want "equal numbers of Brian and Aidan photographs" or "some pictures from every place visited." In addition, users have *photo preferences* about particular photos. A photo might be chosen because it effectively captures a person or situation; other photos may be excluded because they are deemed "unflattering" or "boring."

We conducted a user study of our system with nine people and their personal photo libraries and found that they did indeed use the strategies and criteria we uncovered in our field studies. Our evaluation results suggest that the components of our interface—designed to help users apply both collection and photo selection criteria—are useful for selection tasks.

To summarize the main contributions of this chapter, we present:

- a novel semi-automatic interaction technique for selecting collections of personal photographs,

- field work that deepens the understanding of personal strategies for sharing photographs, and,

- design recommendations for tools that support sharing of personal photographs.

2.2 RELATED WORK

Research on personal photo libraries has focused on creating more effective browsing and searching interfaces through improved display and layout [Bed01, DWR+04, HDBW05, HBB07], improved annotation [KPC+99, WDS+01, SK00, GAW04], and automatic clustering [Pla00, HNS+04]. Elements such as zoomable grid layouts, annotations, and metadata are now part of many commercial photo applications such as Apple's iPhoto and Google's Picasa. Rich metadata with semantic information allows for sorting and filtering mechanisms that enable users to browse their collections in a more directed fashion [KS03, GSW+08] and search for specific photographs more easily [FBA00, KPT+08]. However, merely adding metadata to a photo library is not sufficient. Even with rich metadata, such as tags for people and places, users must still manually scan all of their photos and select the ones they want to share. Furthermore, when selecting photos to share, users are often interested in telling a story or applying subjective criteria which can be hard to encode in a query. In our system the user does not have to formulate a query, but can rather specify their goals for their collection loosely in terms of attributes of their photographs.

The process of selecting photos to share is often called *triage* as it involves culling a large library of photographs into a much smaller collection. Kirk et al. [KSRW06] call this the *pre-share stage* of photowork and find that it is one of the most common and time-consuming parts of working with photos. The Pixaura system [SKL+08a] was specifically designed for the triage process and introduced support for "tentative" decisions. However, the selection process in Pixaura remains manual as with all previous systems.

In our work we create candidate collections automatically and allow the user to refine them—removing the need to look at every photograph in their library, but still providing local control so users can tell their stories. To automatically create sharable collections, we rely on metadata (time stamps, people and place tags) for

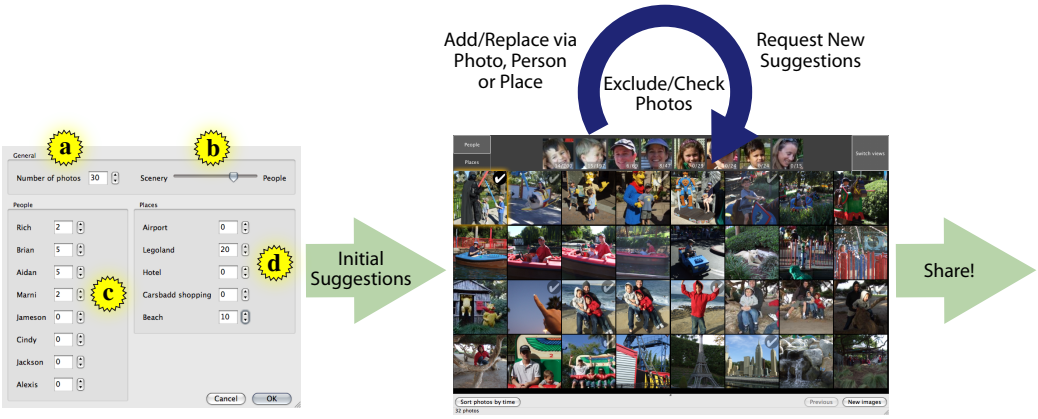


Figure 2.1: *Overview of our system.*

every photo in the library. We expect that this type of metadata will be automatically available soon with wider availability of face recognition and GPS technology.

Recently, Ritter and Basu [RB09] proposed "smart selection" for the file system which allows users to form complex selections by example. In their interface, the user trains a classifier by providing positive and negative examples of files to include in the selection. The system then uses the classifier to suggest other files to add to the selection. For our design, we chose to work with a constraint system instead of a classifier to allow users to express hard constraints such as such as excluding certain people or places. In addition, a goal of our system is to provide a sharable collection immediately; this is difficult to reconcile with a training phase.

2.3 SYSTEM DESIGN

We present a system that enables the user to specify their collection goals and choose individual photos based on the strategies seen in our home studies. Our system (Figure 2.1) is comprised of three parts: an engine that automatically suggests photos that fit the user's collection goals, a set of direct selection tools to add/replace/exclude photos, and statistics that provide feedback on the composition of the current collection. The main components of the user interface include a preferences dialog box for specifying collection goals (left of Figure 2.1), a view

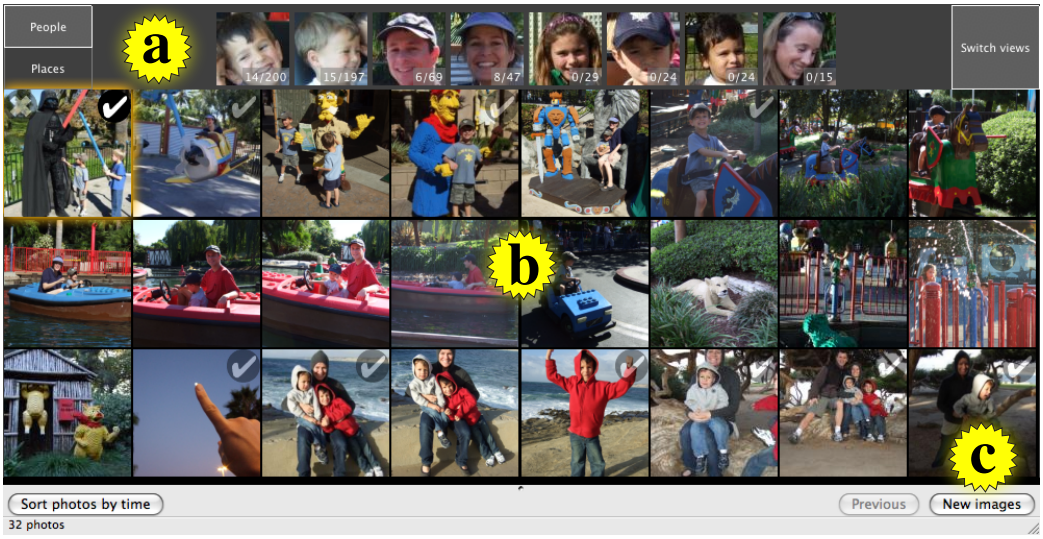


Figure 2.2: *Our interface with current collection of photos in the center and people/places at the top.*

of the current collection of images that provides access to the direct selection tools (Figure 2.2b), representative images of all the people and places in the library (Figure 2.2a), and a *new images* button that asks the system to suggest more photos (Figure 2.2c). The current collection of photos automatically resizes to fit the available window size, and full-screen views are available by double-clicking on a photo.

2.3.1 A user example

To illustrate how the various components of our system work together, consider Rich, who just returned from a trip to California with his wife and two young children. He wants to share the highlights of his trip with his extended family on his photo blog. First, he specifies the general parameters of the collection he is looking for using the preferences dialog box. He chooses to share 30 photos (Figure 2.1a) and decides to emphasize photos with people over scenery shots (Figure 2.1b). He enters zeros for non-family members (Figure 2.1c) and emphasizes Legoland and the beach over the airport and other places (Figure 2.1d). When Rich is satisfied, our system computes an initial collection of photos that match his specified criteria

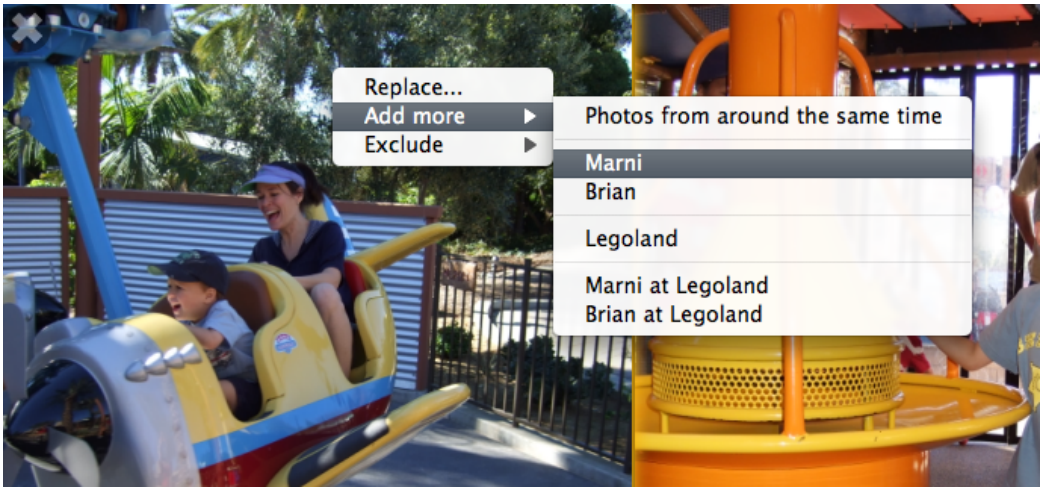


Figure 2.3: *Query-by-example options.*

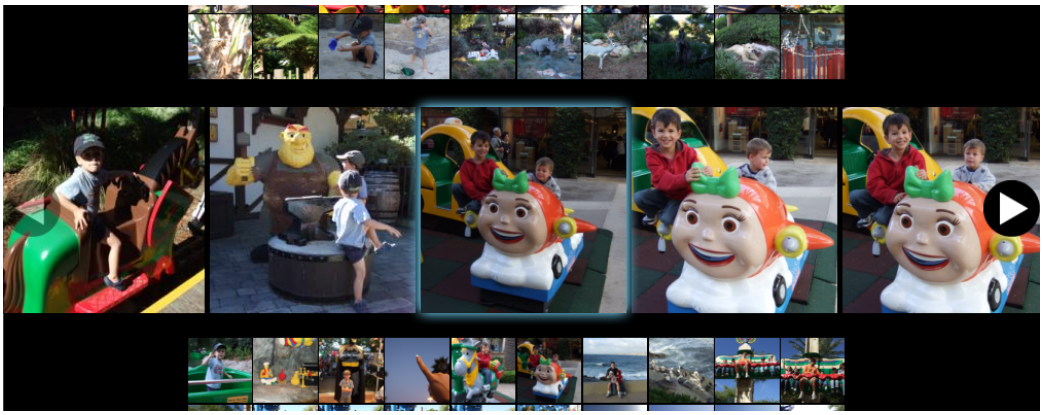


Figure 2.4: *The photo tray for choosing additional photos or replacements.*

(Figure 2.2). Rich immediately sees several photos that he likes and several that he doesn't. Rich *checks* photos he wants to keep and removes photos he doesn't like by clicking on their *exclude* buttons. To get new candidate images, Rich clicks on the *new images* button (Figure 2.2c), which replaces all unchecked photos with new photos and restores the collection to the initial size of 30. Rich can also navigate back and forth between candidate collections using the *previous* and the *next* buttons.

After iterating on his collection, Rich decides that he would like more pictures

of his son Brian. He clicks on one of the pictures that include Brian and selects the *add more* menu item, which offers him several options, all based on the photo he clicked (Figure 2.3). He selects *Brian at Legoland* and a tray of additional photos appears (Figure 2.4). Rich checks several good pictures of Brian and dismisses the tray; the system adds the photos to the end of his collection. To set up the whole story, Rich decides to include one of the pictures of the family at the airport. Since he initially told the system he didn't want any photos from the airport, his current collection contains no such photos. Rich clicks on the *places* button at the top of the interface (Figure 2.2a) and sees representatives for each place in his collection. He clicks on the airport and asks for more photos, analogous to Figure 2.3. The tray of additional photos appears and he picks his favorite. Finally, Rich uploads his finished collection to his photo blog.

The rest of this section discusses the various parts of our system in detail.

2.3.2 Automatic suggestions from collection preferences

Our system allows users to express the following collection goals via the preferences dialog box:

Collection size. The number of desired photographs in the output collection (Figure 2.1a). This preference is typically informed by the sharing method to be used. For instance, many people share fewer images via email than they do via online sharing sites.

People/scenery proportions. The overall proportion of "people" photographs, containing at least one person, versus "scenery" photographs, containing no people (Figure 2.1b).

Individual proportions. The proportion of people photos that a particular individual will appear in (Figure 2.1c). The number for each person represents the desired count of photographs containing that person, but the numbers do not have to add up to the correct total; they are interpreted by the system as proportions relative

to the total number of desired photos. By default the system attempts to balance the number of photos of each individual in proportion to how often they appear in the library. The user can zero-out an individual to remove them entirely from generated collections, or change the defaults to change the balance of individuals, for example, "No pictures of Rich and equal numbers of Brian and Aidan."

Place proportions. The proportion of photos associated with a particular place (Figure 2.1d). These preferences operate indentically to those for individuals.

Generating collections

To generate collections of photos that meet the user's goals, we construct an *integer set constraint problem* and use an off-the-shelf constraint solver [Gec10] to generate solutions, each of which corresponds to a single suggested collection of photos. The constraint solver operates on sets of integers; we first randomly map photos to integers to reduce sequences of neighboring photos in the solutions. The primary tool we use to specify the constraint problem is the *intersection constraint*, in which the intersection of the solution set and some target set is constrained to have a certain size. For example, if the user specifies that there should be 10 photos of Rich, then we construct the set of all Rich photos and specify that the intersection of that set and the solution set should have size 10. Intersection constraints handle the individual people and place proportions and the people/scenery proportion. Note that if there are many solutions that satisfy the constraints, then we pick one arbitrarily. See Section 2.5 for a discussion on how we might bias the system towards better solutions.

Conflicting constraints

It is possible for the user to specify conflicting constraints to the system; we prevent those that we can predict beforehand (e.g. asking for more photos of Rich than exist in the library), but many are more difficult to detect, such as asking for all photos of Rich but none of Marni when Rich and Marni are always photographed together. If the user creates a problem with no solutions, we loosen some of the constraints to

allow the solver to find a solution. For example, we can replace the constraint that a particular person appears in 10 photos with a constraint that the person appears in 10 ± 1 photos, 10 ± 2 photos, etc. Since the original preferences are not exact—the user generally only has a rough idea of the desired make-up of the collection—this approach produces reasonable results. The user can always use one of the direct selection tools to fine-tune the collection. We currently allow a maximum range of ± 3 and loosen all of the people/place constraints simultaneously. We could implement a more intelligent algorithm that loosens the "tightest" constraint first, but we haven't found this to be necessary. Another approach is the use of soft constraints, which is discussed in Section 2.5.

2.3.3 Direct selection tools

We designed several tools that allow the user to add, replace, and exclude photos both individually and as a batch, using the *photo tray* as a common interface. Figure 2.4 shows the photo tray for selecting an addition. The surrounding photos move out of the way of the tray, which is then inserted as a new row as close as possible to the target photo to maintain context with the rest of the collection. The tray acts as a scrollable list, allowing images to be viewed full-screen or selected for replacement/addition.

Addition tool

For a given target photo, we suggest possible additions to the collection by exploring the various known aspects of the target photo. For example, Figure 2.3 shows the options available for a photo that contains Marni and Brian at Legoland:

1. Temporal neighbors of the target photo,
2. Photos that contain Marni,
3. Photos that contain Brian,
4. Photos of Legoland
5. Photos of Marni at Legoland, and,
6. Photos of Brian at Legoland.

Of the possible combinations of time, people and places, we chose these because they were a good match for the kinds of operations users were performing in our initial studies. If the user selects an option, then they are presented with a selection mechanism (Figure 2.4) that allows them to add any number of photos of that type to their collection. In this way we provide a simple version of "query by example" that allows the user to use any photo in the current collection as an intuitive way to get suggested additions. Added photos are automatically checked.

Exclude tool

Similar to the addition tool, the user can use any photo in their collection to exclude entire types of photos from consideration. We generate the same types of photos as with the addition tool, with the exception of the photos adjacent in time. If the user selects a particular option from the list, then every matching photo is excluded: they are removed from the current collection (unless checked), and they will not show up in any future suggestions.

Replacement tool

For a given target photo, the replacement tool suggests replacements from the temporal neighborhood of the target photo. The user can choose one of the suggested photos, which replaces the target photo in the collection and is automatically checked. Initially our prototype offered the same options for photos as with the addition tool, but we found in our pilot studies that users generally wanted to use the replacement tool to choose between alternatives of a specific photo, for example, the best photo from a rapid series of shots. Based on this observation, we simplified the replacement tool to just use the temporal neighborhood.

2.3.4 People and places browsing

We use a facet-like interface for the individual people and places in the library (Figure 2.2a). For each person/place, the system provides a representative image, which the user can click on to access an addition tool identical to that direct selection

addition tool. The people/place representatives are always available and provide a method for returning to images that might have been previously excluded.

2.3.5 Displaying statistics

To assist users in making sure that preferences such as "equal number of Brian and Aidan photos" are satisfied even after the user has forced the system by excluding and checking photos, we added a display that shows the number of photos of each person and each place in the current collection. We overlay these statistics on top of the people and places options (see Figure 2.2a). The display also includes the total number of available photos for each person and place.

2.4 EVALUATION

To determine the effectiveness of our approach, we conducted an exploratory evaluation of our selection interface. Our aim was to investigate three main questions:

1. Do users find our automatic selection tools helpful for creating collections of photos that match their collection preferences?

2. Do users find our direct selection tools helpful for finding specific images that match their photo-level preferences?

3. Are there specific situations in which users like our selection tools more/less than traditional selection tools?

We tested our interface on nine participants (five men, four women); all were employees with a variety of roles at a large software company and between 35–54 years of age.

We asked participants to provide two or more sets of personal photos that they had taken roughly within the last year. For each participant, we chose two sets with similar characteristics (i.e., similar number of photos covering a similar number of different people and places) to use in the study. The uploaded sets ranged in

Figure 2.5: *The reference implementation with scrollable events in the center, photo tray on the bottom and people/places at the top.*

size from 83 to 646 photos, and the average size was 343. We conducted a phone interview with each participant to find out the names of the people and places in their photos, and then we manually tagged their images using this information. Finally, we conducted in-person interviews during which participants used both our interface and a reference interface to create selections from their two collections of provided photos.

We designed the reference interface to include the most common features found in commercial photo organization applications. The interface presents users with an overview of all photos in the library organized into a timeline, clustered into events using the temporal clustering algorithm of [PCF03] and displayed as a series of grids ordered by photo capture time (Figure 2.5). Hovering over the representative image for a person or place at the top of the display highlights the appropriate photos in the main timeline view. Users check photos in the main timeline view to add

them to the current selection in the photo tray at the bottom. As with our interface, double-clicking on an image causes it to be displayed in a full screen view. Note that in our interface, we intentionally disabled the timeline display to ensure that participants would use and evaluate the novel components of our system, despite that previous work has shown overviews to be useful for selection tasks [CMS99].

For each interface, we conducted a brief training session with a test photo library, and then we asked participants to perform two tasks using their library. First, we asked them to select a *co-worker collection* of 15 photos to be shared with their colleagues in a somewhat public form (e.g., an office calendar). We then asked them to select a *family collection* of 30 photos to be shared with family members or close friends. In choosing these tasks, we wanted participants to consider how they would create selections that would be shared with two very different types of recipients. We gave participants 8 minutes to complete each task. At the end of the interview, we conducted a 15-minute debriefing session during which we asked participants which interface they preferred and what specific features of each interface they liked and disliked. We randomly chose which collection was assigned to which interface, and we alternated the order in which participants used the two interfaces.

2.4.1 Findings

Apart from video and application logs, we obtained detailed verbal and written feedback about our system from the nine participants in our study. *All of the participants mentioned specific features of our interface that they liked or found useful.* Furthermore, one third of the participants preferred our interface (by itself) over the reference interface (by itself). Of the remaining six participants, three wanted a combination of the features in both interfaces, one said that he would "gravitate towards" our interface given more time to get used to the features, and two preferred the reference interface. Overall, the results suggest that the components of our interface—designed to help users apply both collection and photo preferences—are useful for selection tasks. The feedback also indicates some limitations with our current design and suggests directions for improvement. Here, we describe the key specific findings in greater detail.

People and places are important. All of our participants took people and places into account when selecting collections of photos. For example, the photos they selected for the *co-worker collection* typically contained far fewer people than the ones they selected for the *family collection*. In several cases, participants selected more photos from specific places that they remembered being particularly interesting or noteworthy. In addition, a few participants tried to at least roughly balance out the number of photos of different people when creating the family collection. This behavior reinforces some of the findings from our field study and helps validate our high-level approach to design an interface that makes it easy to navigate and filter photos based on people and places.

Positive reaction to direct selection tools. Six out of the nine participants said that they liked the direct selection tools in our interface, with the *replace* and *add more* features being the most popular. We observed a variety of usage patterns for these tools. Many participants used the *add more by person* feature to find more (or better) photos of an individual who appeared in the selected collection. One participant used the *add more around this time* feature extensively to find and compare similar photos of a specific action or event. Another participant used the *add more by place* feature to look for better representative images of the cities he visited while on vacation. A few participants who seemed relatively unfamiliar with their libraries enjoyed using the direct selection tools to explore their photos along specific dimensions. In general, participants liked the ability to see and focus on a set of candidate photos that were sorted and filtered based on semantically meaningful characteristics. One participant said she "liked that you could sort by people/places and that you had multiple ways to extract/add the preferred data." Others described the direct selection tools as "powerful" and "natural." The positive reaction to this component of our interface helps validate the design of our direct selection tools and suggests that sorting/filtering can help users apply the kind of photo-level selection criteria necessary to create satisfying collections of photos.

Mixed reaction to automatic selection tools. Five of the participants used our automatic selection tools in the manner that we expected. Namely, they used the preferences dialog to specify the desired collection preferences of the selection at the beginning of each task, and then they periodically used the *new images* button to ask the system for more candidate photos. Three of these participants said that they found the automatic selection to be useful because it helped them find a good mix of photos that covered the range of people and places that they wanted in the collection. One participant mentioned that the automatic tools made the selection task seem "like a game," and he described the *new images* feature as "playful." Furthermore, participants who were less familiar with their libraries seemed to like the fact that the system was able to automatically suggest photos for them to consider. Finally, three participants said that the automatic selection feature would be helpful for quickly creating a representative selection of images because they would not have to consider every photo in the library. This range of positive feedback suggests that at least some users will appreciate the ability to specify collection preferences and would likely use our automatic selection tools.

On the other hand, several participants did not like or understand the automatic selection tools. One participant felt that the photos suggested by the system were "random," and another complained that the *new images* feature was too "unexpected." It is possible that users with very strong photo-level preferences will typically not like the photos that the system automatically suggests. For example, one participant who did not like the automatic selection tools had very specific photo-level preferences about the framing and composition. In addition, two participants mentioned that they did not want to specify selection criteria in the preferences dialog before seeing any photos. One potential approach for addressing this issue would be to allow users to specify or change preferences at any point during the selection process.

Statistics were useful. Two of our participants said that they found the statistics about the various people and places represented in the current selection to be a useful piece of information. They liked that they could see the distribution of people and places at a glance. This result follows logically from the observation that people

and places represent important criteria for most selection tasks.

Overview of photos is important. As mentioned above, we intentionally disabled the timeline display in our interface for testing purposes. Not surprisingly, many participants mentioned certain situations in which they wanted an overview of all their photos when using our system. In particular, users wanted an overview for finding specific photos they remembered and browsing through all their photos in an unstructured manner.

2.4.2 Summary

The results of our evaluation provide some answers to the questions we hoped to address and suggest a few directions for improving our interface. There is strong evidence that our direct selection tools help users find satisfying images based on their photo-level preferences. The feedback suggests that sorting and filtering candidate photos based on semantically meaningful properties is a useful and intuitive operation for the set selection task. We also found some evidence that our automatic suggestion tool can be helpful for enforcing collection preferences. Although the response to this feature was somewhat mixed, the results suggest that many users would be willing to let their photo software automatically suggest pictures for them to share. However, some of the feedback suggests that the next iteration of our design should allow users to adjust collection preferences at any point during the selection task and not just at the beginning. Finally, regarding the specific set selection situations in which users would prefer our interface, the feedback indicates that our tools are well suited for exploring relatively unfamiliar photos and for quickly generating representative sets from large image libraries. On the other hand, more traditional overview displays make it easier for users to find specific images that they remember and to browse through all of their photos. Thus, one area for future work is to determine how best to combine overviews with the novel components of our interface into a single system that works well in all situations.

2.5 CONCLUSION

Considering the studies discussed in Chapter 1, we present several *design considerations* and *a novel semi-automatic interface* for selecting collections of photographs. Our user feedback lead us to the conclusion that the novel components of our system are useful in selecting photographs, and that adding our interface components to a traditional photo browser would likely provide a powerful and satisfying user experience. We are in the process of designing such a hybrid system, and look forward to evaluating it in future work.

We are considering several improvements to our system that would speed the selection process. Some of our users asked for a way for the system to identify blurry or dark photos, or identify photos in which all the subjects' eyes are open. Although most of the desirable qualities of a photograph require deep semantic knowledge, there are a few that are amenable to algorithmic assistance (See Chapter 1 for more discussion of the various photo qualities we found in our studies). We would like to explore integrating work from the computer vision community to create a ranking over the photos which could be then used to bias the system towards more desirable photos. This might reduce the time required for selecting a collection by removing highly-undesirable photos from consideration.

There are several other interesting heuristics to explore that could improve the quality of our automatic suggestions. Users commonly take a rapid series of photos when something interesting occurs—such a series might well be a "highlight" and should probably be included, but only once. Similarly, posed group photos are almost always orchestrated with some effort which seems to indicate that they should be given some preference in the selection. Also, a balance between group shots and individual shots could be useful, since a selection with only one or the other tends to lack interest.

Our constraint system uses hard constraints, which means that there is no way of specifying that some solutions are more desirable than others without excluding the other solutions entirely. If the user's constraints are not too tight, then the system returns a very large space of solution collections that are equivalent with

respect to the constraint problem, but not necessarily equivalent from the user's perspective. There may be reasons to prefer some of these collections over others: a better distribution of photos in time, better individual photo properties (e.g. focus, lighting), etc. In future work we would like to extend our system to use soft constraints to handle these types of preferences. The challenge is to choose the better solution collections in a reasonable amount of time: the system must remain interactive.

Currently, users specify collection preferences in our system by entering parameters directly into a dialog box, but this is a fairly low-level interface to our constraint system. In future work, we plan on exploring a *template* system that will free users from having to specifying each parameter directly. For example, a user might be able to choose from a "family vacation" template, a "wedding" template, or a "landscapes" template. The template would contain not only the prescription for setting various parameters, but also key people and events. For example, the "wedding" template would know that a wedding typically contains a bride and groom and that major events include the bride walking down the aisle, the exchange of rings, etc. An important aspect of such a system would be an interface for the user to specify their own templates, as well as learning templates from a given hand-made collection of photos.

Finally, our evaluation focused primarily on single events; it would be interesting to evaluate our system on larger, multiple-event collections.

Chapter 3

Creating collections with automatic suggestions and example-based refinement

3.1 INTRODUCTION

Personal media libraries are an important part of daily life, as users amass music, photos, and videos. One common user task is collecting items from such libraries for a specific purpose. For example, people create music playlists for parties, exercise, and work. And when sharing photos from a recent trip, people often select a handful of good photos from hundreds or in some cases thousands of potential candidates. If we consider the term *collection* to refer to a subset of items from a larger set (the *library*), then the challenge across the above-mentioned situations is creating a relatively small collection from a potentially very large personal library.

Today, there are two types of interfaces for creating collections: manual and automatic. Faceted filtering interfaces [YSLH03, Hea09] have made it easier to manually select items from large libraries by allowing users to narrow down possibilities and focus on a specific group of items. For example, if users are interested in a playlist of Michael Jackson songs from the album Thriller, they can sort or filter their entire music library along those two facets and create the playlist. However, users often want collections that vary along multiple criteria, such as playlists that include multiple artists and genres [HG09]. To create more varied playlists, users must query or filter by individual selection criteria, such as artist, and manually select songs, for the collection. Automatic solutions are typically either fully random, or example-based [App10, FTKW08, RB09]. In the latter case, the user provides one or more example items and a collection is defined automatically based

on item properties, such as genre, artist, year, etc. Although automatic interfaces can be effective for specifying goals that are hard to describe in words (e.g. give me more songs with this kind of melody), they give the user much less control over the final collection, as the user can only give examples and can't express goals, such as "I only want rock music."

To better understand user needs for working with collections, we surveyed previous work and carried out contextual-inquiry interviews, as described in Chapter 1. We found that when working with photographs and music, users have concrete goals in mind. For example, a playlist for work should have ambient music with few words, while a playlist for a party should have energy but include at most one or two songs from any specific artist. But it turns out that users are often satisficing, i.e., they are not looking for a specific item and will often stop when they find a "good-enough" photo or song instead of continuing to search for the "best" item [BMH06]. They also frequently get sidetracked by specific photos or songs (often those with personal significance) that inspire them to browse and add other related items. As a result, creating collections is typically an iterative process where users successively refine a set of candidate items until they are satisfied with the final collection. During this process, users sometimes change their minds and end up with something totally different than what they initially had in mind.

Learning about user needs brought us to the realization that both automatic and manual approaches are necessary. On the one hand users are satisficing and in many cases are not too concerned about which specific items end up in their collections. This suggests the need for automatic methods that eliminate the manual effort of selecting individual items. On the other hand, users want to refine their sets based on the relationships between items, which indicates the importance of more user-directed refinement tools. Furthermore, since the selection criteria for some items is often highly subjective and deeply personal, it is hard to imagine how any fully automatic algorithm would be able to reliably create personally useful collections, even with multiple examples.

We propose a semi-automatic interface for creating collections that combines freeform text input with example-based iterative refinement. With our approach,

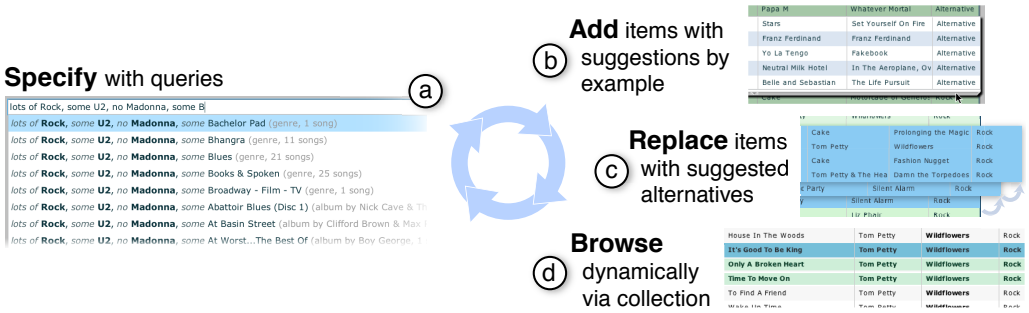


Figure 3.1: *Our approach to creating collections of items such as music playlists and photograph albums is to specify the general parameters of the collection with a keyword query interface (a), and to provide tools for adding, replacing, and browsing items by example (b–d).*

users can create collections, such as music playlists, automatically by specifying high-level preferences about their collection with keywords, such as "lots of rock, some U2, no Madonna, 2 hours" (Figure 3.1a). User preferences are transformed into constraints and plausible collections are generated with an off-the-shelf constraint solver [Gec10]. We chose text input over traditional graphical user interfaces with checkboxes and sliders because media libraries often include an overwhelming number of facet values. For example, a typical music collection may have over 500 different artists. With text, users can describe the criteria they care about without being overwhelmed by all the possibilities. The disadvantage of text input is that the user may not know what to type. To remedy this challenge, we offer autocomplete feedback, similar to Inky [MCB+08].

Since users want to iteratively refine their collections based on examples, we propose three different collection editing techniques. First, we introduce an in-place *suggestion widget* for adding new items to the collection (Figure 3.1b) based on item metadata. Second, we allow users to explore *replacement alternatives* for one or more items in a collection by automatically suggesting replacement items (Figure 3.1c). Finally, we link the user's library to the collection and enable users to dynamically sort and scroll the library relative to any item in the collection (Figure 3.1d). This *linked view interface* allows users to pivot into their library and easily add new

related items.

Our contributions are as follows:

- A framework for iterating between automatic suggestions and example-based refinement,
- A simple textual query language for constructing collections of items,
- Example-based user interface elements for
  - Replacing items with equivalent alternatives,
  - Adding related items, and,
  - Browsing the library relative to one or more selected items,
- A constraint-based implementation expressed as integer constraint solver.

We present two applications, SongSelect, which we use to explain our approach and perform user testing with, and PhotoSelect, which we employ to demonstrate the adaption of our approach to a different domain. Early user feedback on SongSelect is positive. Users like the text interface and suggestion widget and are happy to see automated playlist creation with more user control. We conclude the chapter with a discussion of the design considerations for collection-oriented tools.

3.2 RELATED WORK

Hansen and Goldbeck [HG09] pose creating collections as a recommendation problem, namely how can we build recommender systems that can suggest collections, not just single items. They propose three key aspects that must be considered when building systems for recommending collections: the value of the individual items, co-occurrence interaction affects, and order effects including placement and arrangement of items. In this work we focus on the interface for working with collections and propose a set of interaction techniques that together present the user with a semi-automated approach to building collections. Our text query interface allows users to specify co-occurance preferences, and the constraint solver resolves item

interaction effects and satisfies collection-level goals. Our current design does not consider order effects.

Unlike today's faceted- and keyword-search interfaces, which require the user to specify selection criteria item-by-item in order to build a collection that varies across facets, early information retrieval systems returned sets, or collections, of items using boolean logic queries [Tun09] expressed in languages, such as Structured Query Language (SQL). Some modern applications still support boolean logic queries via form-based interfaces (e.g., iTunes Smart Playlists). Boolean logic offers richer control over constructing a collection than simple keyword queries and can enable selecting sets of items that vary along multiple axes. However, aside from the difficulty in learning database languages, databases are optimized for generating collections by applying filters on an item-by-item basis. This focus on items makes it difficult to implement constraints that apply to the collection as a whole. A more natural approach is to treat the problem as a constraint-satisfaction problem on integer sets, where items are mapped to integers and constraints of all types can be specified directly [Kum92]. This is the approach we take, as described in the Implementation section.

Our text interface is inspired by sloppy command interfaces like Inky [MCB+08], CoScripter [LLC+07], and Chickenfoot [BWR+05], but our keyword commands are simpler than those required to be transformed into executable code [LM06]. With SongSelect users type commands like "mostly rock, some U2, no Billy Idol, less than 3 hours" and since most terms in the command contain facet values, the parsing becomes a matching task. There remain ambiguities, however, and we use a scoring scheme very similar to that of Inky to choose a single, unambiguous interpretation of the user's input [MCB+08].

Automatic example-based algorithms for creating collections continue to be improved [RB09, dEW06] and there are a number of domain-specific approaches [CA09, KPSW06, PVV08, PW05, RBH05, Pla00]. However, as we discuss in the next section, users want to iteratively refine collections and certain items can have deep personal meaning, which makes it challenging to completely automate the process.

Quantifiers	*all*, *lots of*, *some*, *a couple*
Prepositions	*by*, *from*, *at*
Durations	*< 30 min*, *about two hours*
Counts	*20 songs*, *more than five songs*
Sizes	*less than 250 Mb*, *2 gigs*
Time	*before*, *prior to*, *after*

Table 3.1: *Examples of modifiers accepted by SongSelect in addition to titles, artists, albums and genres, with selected examples (not an exhaustive list).*

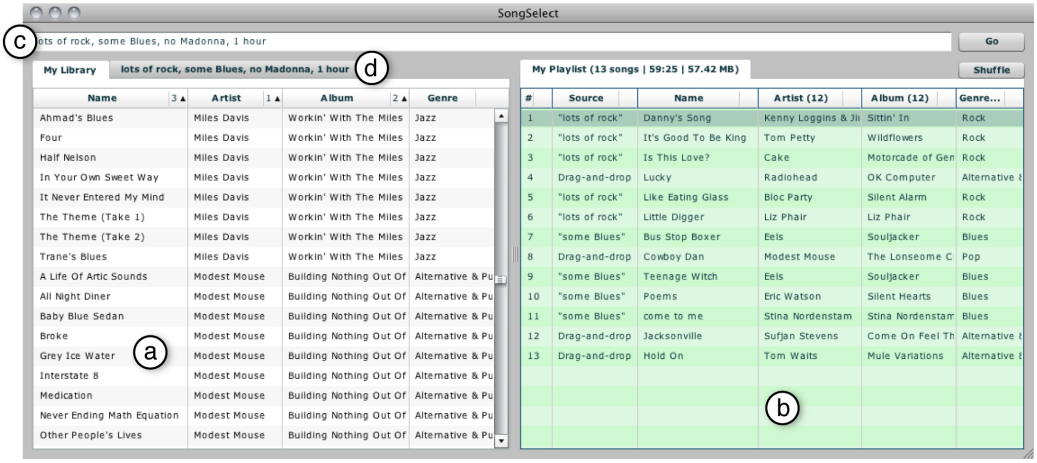


Figure 3.2: *The interface for SongSelect include the text box for specifying queries (c), the browsing pane for browsing the library (a) and query results (d), and the user's current playlist (b).*

3.3 USER INTERFACE

We propose that tools for working with collections need to support both automatic suggestions and interactive refinement. We first describe our interface design in the context of the music playlist application, SongSelect, and then discuss how this design applies to photo collections.

3.3.1 SongSelect

SongSelect has three areas, a text box at the top for specifying queries and two tabbed panes (Figure 3.2). The pane on the left displays the user's music library and query history as tabs (Figure 3.2a), while the pane on the right displays playlists (Figure 3.2b). The user can start making playlists in the traditional way, by selecting songs from the library and dragging them over to the playlist. Or the user may ask SongSelect for suggestions. To get suggestions, the user specifies his preferences with keyword queries, such as "lots of rock, some U2, no Madonna." Given a user query (Figure 3.2c), SongSelect creates a playlist and stores it as a tab next to the user library in the left pane (Figure 3.2d). Note that these keyword queries are closer in flavor to the early set retrieval interfaces than today's keyword search interfaces, which retrieve lists of ranked documents. Also, a SongSelect playlist is analogous to a database view, although some user preferences can have relationships making them more complex to satisfy with traditional database query languages, such as "only one song per artist." We chose a freeform text interface over a traditional form-based GUI interface in order to allow users to specify only the things that are relevant without being overwhelmed by all the available options. However, other interfaces are possible, for example, a visual layout interface such as Musicovery [Mus10] would be an interesting alternative to the table-views in our interface.

To support interactive refinement, we designed with examples in mind. SongSelect supports three example-based interaction refinement techniques. First, the user can select one or more items in a playlist and look for alternatives by simply pressing the keyboard arrow keys. The selected songs are replaced in-place with alternatives derived from the user queries (Figure 3.3). Second, to better access the user's intent behind an example, we present a new widget, *the suggestion widget*, which allows users to flexibly add new songs to the playlist using song metadata, such as artist, album or genre. For example, if the user wants to add five more pop songs to his playlist, he can drag on "Pop" in the genre column (Figure 3.4). Finally, to support library browsing as part of collection refinement, we link the library view to the playlist view and allow users to use any song as a pivot for finding related

songs in the library (Figure 3.5).

Keyword queries. Since users are familiar with keyword search interfaces, which can take any string as input, we designed the playlist query text input interface to be as flexible as possible. Users can type as much or as little as they like. SongSelect will do its best to infer user intent and return a reasonable playlist. We define a user query as a list of criteria, such as "some rock, a lot of U2, no Alternative," that is specified as comma-separated *phrases*. Each phrase corresponds to a user criteria. Phrases can be as simple as the name of an artist, album, genre, or song. For more precision users can add modifiers, such as "mostly," "some," "a lot," "no." When a modifier is not specified, we assume the most general "some" modifier. Complex phrases include two or more facet values, such as "rock by U2" and "Michael Jackson before 1990." Phrases can also describe criteria about the playlist length, such as "max 2 hours, no more than 40 songs." Table 3.1 shows examples of the types of the modifiers SongSelect supports. If a phrase fails to parse properly, SongSelect alerts the user and asks him to correct or remove the phrase.

To aid the user in making queries, we designed an autocomplete widget that recognizes phrases and provides information about items in the library (see Figure 3.1a). The autocomplete widget includes genres, artists, albums, and songs. It lists the number of available items next to each autocomplete item. We found this critical for setting user expectations. If they only have one hip hop song, they should not expect to get a whole playlist of hip hop songs. Additionally songs and album entries include the artist name.

Users can also add phrases to the query by browsing their library and double-clicking on songs. The generated phrase depends on the column where the user double-clicks. To generate a song phrase, users can double-click on the song name. To generate an artist phrase, such as "some U2", users can double-click on the artist.

Finally, user queries are stored as tabs in the left pane (Figure 3.2d) so that user may go back to previous queries. This is useful as a history mechanism, but it also enables users to merge multiple playlists together or add new songs through keyword queries.

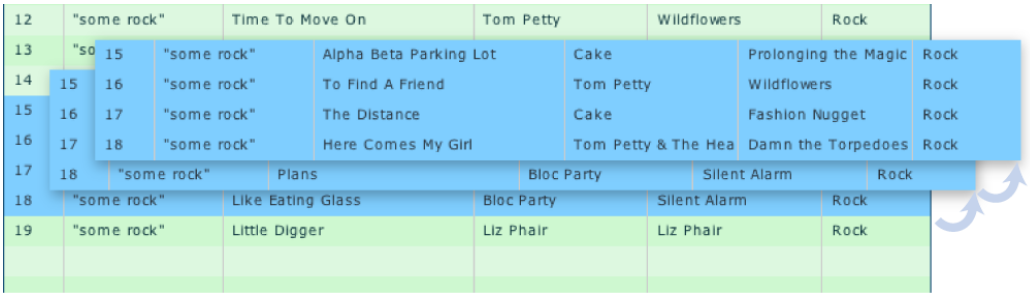


Figure 3.3: *Successive alternatives for songs generated from the phrase "Some rock."*

Exploring alternatives. The user can browse through alternatives for the music playlist by selecting all or parts of the playlist and cycling through suggestions with the left and right arrow keys, as shown in Figure 3.3. Alternatives are generated through the phrases specified by the user. So if the user selects a few songs that were suggested because he asked for "some rock," he will see other rock songs that are not already in the playlist. If the user selects a song that was added to the playlist manually, through drag and drop, then SongSelect defaults to using the artist name and gives alternative songs by that artist. Songs that are added to the playlist through the suggestion widget are given a virtual phrase that corresponds to the basis for the suggestion. For example, if the user dragged the "rock" suggestion widget, the songs that are added are assigned the "rock" phrase.

Suggestion widget. The suggestion widget allows users to grow a collection in-place, removing the need to move back and forth between the library and the collection. The suggestion widget is visible to the user as a thumb and is available for all cells that are associated with more than one item. Since song names are typically unique, cells with song names do not include a suggestion widget, while an artist, album, and genre are not unique and can therefore be used to add more items. The user clicks on the thumb and drags down to add songs. SongSelect limits the size of the expansion with the number of available songs. Users can cycle through alternatives in the suggestion widget in the same manner they explore alternatives in the playlist. This can be useful when the user wants to add only one or two songs

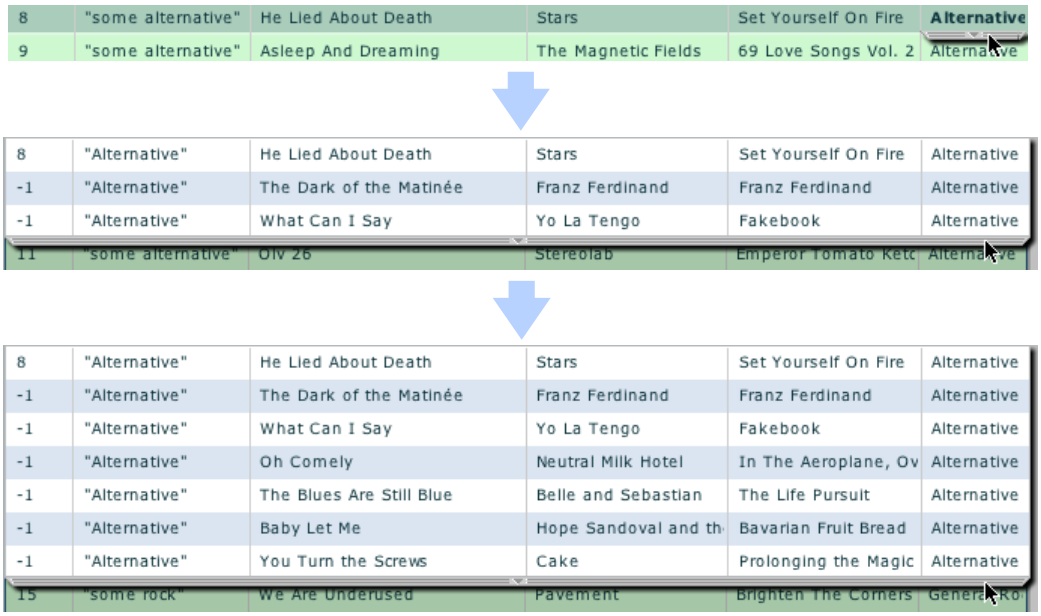


Figure 3.4: *The suggestion widget invoked on the facet "Genre" with value "Alternative," the progressive suggestions as the user expands the widget.*

(Figure 3.4).

The suggestion widget presents a novel way to add items to collections and may be used independently from the keyword interface. It allows users to create playlists by example, but constrains how that example is used.

In SongSelect we implemented the suggestion widget as a tray that appears on top of the playlist. One limitation of this approach is that as the user expands the tray and adds new songs, he hides some of the songs already in the playlist. In PhotoSelect the suggestion widget expands in place and moves the items following the selected item. This approach does not hide any elements but can cause extra shuffling, which may be distracting to the user.

Browsing libraries with linked views. Our interviews and prior research show that users want to be able to browse their library when they are creating a collection. To enhance this back and forth behavior, we designed a two-panel interface with a panel for the library on the left and a panel for playlist on the right. The two panels

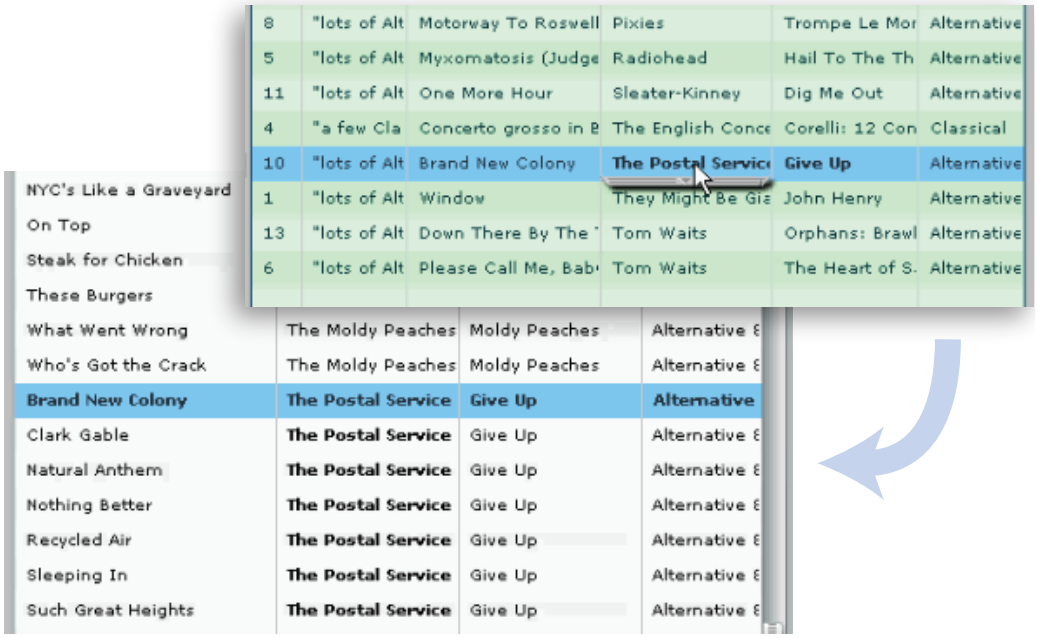


Figure 3.5: *Selecting a song by* The Postal Service *in the playlist sorts the library by artist and scrolls to show other related songs.*

are linked and the library is sorted and scrolled dynamically with respect to the playlist. When the user clicks on any song in the playlist, the library panel is sorted and scrolled to the song the user selected (Figure 3.5). The sorting is based on the column the user clicked. If the user clicked on the name of the song, the library is sorted by song name. For better orientation, we support secondary and tertiary sorting. If the user clicked on the genre of a song, the library is sorted first by genre, then by artist, and then by song. We specify default sorting rules for any metadata dimensions. Since not all song metadata is always visible, SongSelect includes a context menu that allows users to pivot on any of the metadata associated with a song, such as rating, number of times played, year released, etc.

Rich feedback. Providing effective feedback was a key design goal, since users can be confused by automatic suggestions. SongSelect offers a number of rich feedback features. First, the autocomplete widget sets user expectations, so that if a library

has only one rock song, the user should not expect to create a playlist of only rock songs. Second, the playlist includes a column, the "source" column, that describes why a song is in the playlist (see Figure 3.2). It lists the phrase that is responsible, the facet value that was used with the suggestion widget, "drag-and-drop" for manually added items, and "auto fill" for items that were added to satisfy a length or duration criteria. The playlist grid also shows statistics about the playlist. The tab displays the number of songs, duration, and size of the playlist. The columns also list the number of unique items in that column, so that the user can quickly see how varied or uniform the playlist is.

Since users are often concerned about how the items in a collection relate to one another, we added brushing to the playlist view. As the user moves the mouse over the playlist related items are highlighted. When the user puts the mouse over an artist name, such as "Cake," all "Cake" songs are highlighted. When the user moves the mouse over a genre, such as "rock," all rock songs are highlighted. This type of feedback allows users to assess the playlist and decide whether they need to add new songs for balance.

3.3.2 PhotoSelect

PhotoSelect demonstrates the adoption of our general approach to a new domain and thus repurposes some of the UI features of SongSelect (see Figure 3.6a). The interface has a search box at the top with two panes. The library appears on the left and the collection to share appears on the right. To gather up photos to share, users can type queries such as, "mostly landscapes, none from the hotel, a few group shots," or they can scroll through their library and drag and drop photos into the collection. The three collection refinement techniques are also available. Users can scroll through alternatives, use the suggestion widget to add new photos, and dynamically sort and scroll their photo library. Since photos have an inherent time-based ordering, which is also preferred by users [GGMPW02], we select time as the default sorting criteria. PhotoSelect uses people and place metadata, and timestamps. Although not all of this metadata is part of all personal photo collections today, with advances in GPS technology and face recognition, we expect this metadata to

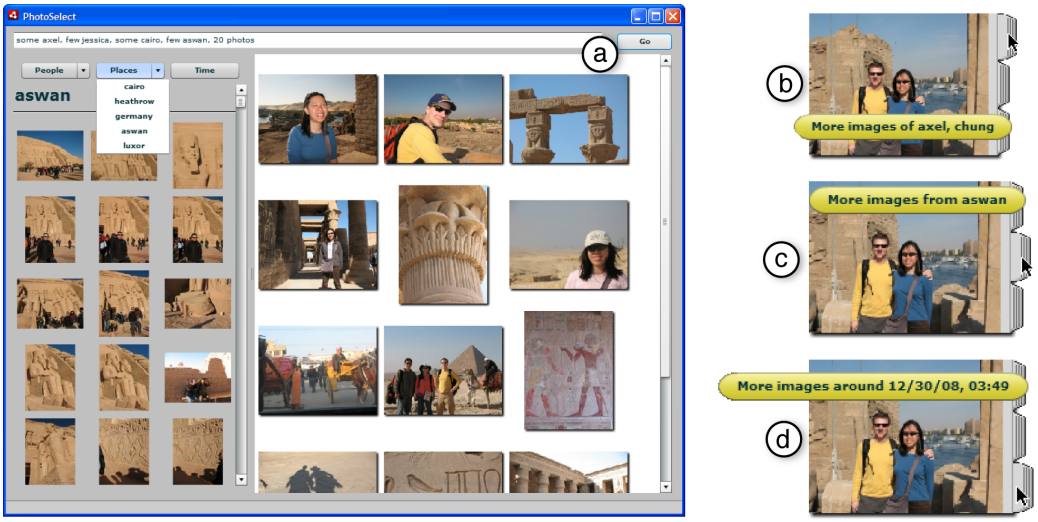


Figure 3.6: *(a) The interface for PhotoSelect includes a text box for specifying queries (top), a browsing pane for browsing the library (left), and the user's current collection (right). Suggestion widget tabs and tooltips for people (b), places (c), and time (d), respectively.*

become a ubiquitous part of personal photo collections.

There are some key differences between PhotoSelect and SongSelect. First, photographs are displayed in a grid instead of a list, which means that metadata incorporated with an image is not readily visible. Since the suggestion widget in SongSelect uses visible metadata, we redesigned it for PhotoSelect (Figure 3.6b–d). When the user selects a photograph, three thumbs appear. The thumbs correspond to people, places, and time. The user can add more photos based on the people, place, or time of any photo in the collection.

We added additional keywords to our parser in order to support domain-specific keywords. For example, users may want to say "mostly landscapes" or "a few group shots." We expect that every new domain will have some number of specialized keywords.

3.4 IMPLEMENTATION

In order to support different domains, we separate the implementation into a client-server model, where the client contains domain knowledge and the server is a generic domain-independent constraint solver. The client transforms user input into a set of constraints using its domain knowledge, passes those constraints to the server, which then returns solutions to the constraint problem. The advantage of translating domain-specific user constraints into a domain-independent form is that we can use powerful off-the-shelf solvers for a variety of applications. The server is implemented in C++ as a web server and solves constraint problems using the Gecode constraint-solving toolkit [Gec10]. For rapid prototyping and iteration, the user interfaces are written in ActionScript using Adobe Flex 3 [Ado10b] user interface elements and the Adobe Integrated Runtime (AIR) application framework [Ado10a].

Many simple queries can either be fully satisfied without querying the server, such as single-phrase queries ("some rock"), or partially satisfied, such as phrases that include or exclude items ("no Madonna"). However, most non-trivial queries do require the services of the server, such as queries whose phrases' domains overlap ("some Madonna, a few pop"), or phrases that specify a constraint on the set as a whole ("less than 30 minutes").

3.4.1 Translating queries into constraints

As described in the user interface section, queries are comprised of a series of comma-delimited phrases, each of which produce a constraint on the items in the solution set.

Item constraints

The most common type of constraint specifies a class of items in the library and the size or proportion of that class in the solution set. For example, "lots of Madonna" specifies that the class of songs by Madonna should be 50% of the solution set. And "no rock by U2 less than 3 minutes" translates into zero songs that that contain "U2" as artist and "rock" as genre and are less than 3 minutes long. Table 3.2 lists how we

Quantifier	**Proportion**
all, *everything*, *every*,	100%
most, *mostly*, *most*	75%
lots, *lots of/from/at*,	50%
some, *some of/from/at*,	25%
few, *a few*, *a little*	10%
couple, *a couple*,	2 items
one, *one of/from/at*,	1 item
none, *nothing of/from/at*,	0 items

Table 3.2: *We transform quantifiers into proportions in order to issue constraints.*

translate different types of quantifiers. Users could also directly specify proportions or number of items. In our pilot study we found that users didn't often know exactly how many items they wanted in their collection and preferred using more vague query words.

Before passing the constraints to the server, the client goes through all constraints to normalize the proportions and check to see if each constraint can be trivially rejected. So, for example if the user asked for "some U2" but he only has two U2 songs, the constraint is modified to specify two U2 songs, instead of 25% of what could be a 2 hour collection. All proportions are normalized to add up to 100%. This ensures that the constraint solver does not return items the user did not request. For example, the query "mostly rock, some U2, some Madonna, no Michael Jackson, a few Billy Idol" becomes 55% rock, 19% U2, 19% Madonna, and 7% Billy Idol for our sample library.

Set constraints

Apart from constraints on classes of items, the user can set constraints on the solution set as a whole, such as "some rock, < 90 min". If a duration, song count or size is specified by itself in a phrase, it is understood to apply to the entire set. The constraint is constructed by summing the values from individual items, for example, the phrase "< 90 min" is translated to be "the sum of song lengths < 90 min". If

no size or length constraints are specified, SongSelect and PhotoSelect default to a length of 20 songs or photos.

Parsing

The user queries are parsed after every keystroke and the results are used to populate the auto-complete widget. When the user is finished entering a query, the parsing results are the first step in transforming the query into the constraints that are sent to the server.

The language for describing collection preferences is inherently ambiguous. Does the word "all" refer to the quantifier from Table 3.1, or does it refer to a (possibly-partial) title, album, artist or genre? What if the library contains both an artist and an album named "all?" We use a non-deterministic parser which generates all possible interpretations of the user's query and applies a simple scoring function to rank the interpretations. The autocomplete widget uses this ranking in its display. For each parser token like *title* or *duration*, the scoring function assigns a numeric score based on the semantics of the domain. The score of the entire phrase is simply the sum of scores for each token. In the music domain, we score tokens in the following decreasing order: genre, artist, album, title, anything else. The interpretation with the highest summed score wins.[1] We choose this ordering (general to specific) because we expect the user to use the queries to summarize the general properties of the solution set, not to pick out individual items. In the case where "all" matches both an album and an artist, the parser favors the artist, which results in more songs in the final solution set. In the photo domain, there is much less metadata, so ties are not very common. The parser scores people higher than places and places higher than arbitrary tags.

3.4.2 Implicit constraints

When users create collections, they have some implicit criteria that apply to most if not all collections. One such criterion is that they want their collections to be diverse

[1] In the case of a tie between two interpretations, the most specific interpretation wins.

(e.g. playlists should include multiple artists and photos collections should include photos from the entire library, not just a small subset). To create diverse collections, we configured the server to explore the subspaces of constraint-matching sets at random, ensuring that a query of "some rock" would not return a playlist with only songs from the first rock artist in the library, for example. While this simple heuristic works fairly well, it does not ensure diversity. In the future, we plan on exploring different strategies for ensuring diversity. Photo collections, for example, often include bursts of photos taken at approximately the same time when users captured several images in an effort to get "the best" one. It seems likely that users would only want one photo from such events and this could be encoded as an implicit constraint. Photo and music quality could also be part of the implicit criteria used to generate a collection.

3.4.3 Limitations

Perhaps the biggest limitation of our approach is that the constraint solver sometimes fails to find a solution and gives no feedback for the reasons for this failure. The SongSelect and PhotoSelect applications do a fair amount of processing in order to pass along constraints that are satisfiable but sometimes slight changes to the query could yield a better result. To improve our approach, we plan to explore techniques that allow for soft and hard constraints, and fail more gracefully. Since users are satisficing, returning a "good enough" collection is generally better than returning no result at all.

Our approach also does not consider ordering effects within a collection, but such preferences could be expressed as constraints in our current implementation. An interesting future direction is to explore adding weights to constraints so that users can express queries such as, "prefer rock over pop music" or "pick outdoor scenes over indoor shots."

3.5 USER FEEDBACK

We report on user feedback from a pilot study of SongSelect with four participants. We plan a comparative evaluation with manual and automated approaches in the future. Each session lasted an hour and included an introduction into the participant's existing music collection and playlists, a training task with SongSelect, and a playlist-creation task of their choice. The participants were between 24 and 40 years of age, two male and two female. All of them use Apple's iTunes software and have over 5000 songs in their libraries (one participant has over 70,000 songs). All of them reported making playlists, but their playlists had very different characteristics, and they varied in the amount of time they spent making them. Some made playlists as a way to put songs on their mobile devices, while others made themed playlists for fun or as part of an exercise class. Two of the participants were very concerned with the order of songs in their playlist, while the other two didn't care about the order at all (to the extent of deliberately shuffling the order of their playlist). All of them reported listening to music every day. All of the participants were aware of iTunes' Genius playlist creation tool, but perceived it to be biased towards popular music and not applicable to their personal libraries. Two participants mentioned using Pandora, an online streaming music service that creates streams of music similar to examples provided by the user. Two participants explicitly said that they enjoyed making playlists and enjoyed the results, but found the process tedious and wished that there were faster ways of generating quality playlists.

Overall, the participants reacted positively to SongSelect. Three commented that they liked the automated-but-malleable nature of SongSelect playlists. All four commented that they would like more automated tools for creating playlists but that they were concerned with the lack of control with existing systems for automatically creating playlists.

When asked about their favorite part of SongSelect, all participants agreed that the in-place suggestion widget was a great way to add new items and that being able to look at the library side-by-side with the playlist was useful. Two of the participants requested that a modal suggestion widget so that they could add just the songs they

wanted and not all suggestions. Three participants liked the ability to look through alternative item suggestions with the arrow keys but requested more control. Two participants wanted to be able to change the way SongSelect generated alternatives. One wanted to constrain the suggestions to a specific artist, while another wanted to change the genre of the songs.

All our participants were able to use the text-based interface to generate an initial playlist. One participant said he preferred SongSelect's interface to iTunes' SmartAlbums interface because he didn't have to move the mouse, he could just type. All of the participants wanted to be able to use more metadata dimensions in the text box, in particular the album year, the dates the songs were added, and other, more subjective dimensions such as the "energy" of a song. Three participants remarked on the inaccuracy of the genre metadata in their libraries, and noted that the quality of the query results depended on the accuracy of the metadata.

The participants who struggled with the text interface had two main challenges. One was confused by the results returned by the constraint server and felt that SongSelect was hiding things from her. This was likely due to a software bug and could be improved. The other participant thought of SongSelect's interface as a search interface not as a playlist creation interface and as a result was confused when SongSelect produced only 20 songs and not every matching song. This may be a more serious issue and one that requires careful design. By converting the search box into a query command interface SongSelect takes away the ability to search the library. This could be remedied, by explicitly including a Find feature that could be triggered with common key commands (Ctrl-F). However this confusion may speak to a larger problem: that users have expectations of what appear to be search text boxes. That is, queries in text boxes are for searching for specific items, not entire collections of items with interdependencies. This type of interface may require more training and a longer term field study to evaluate its effectiveness.

3.6 DISCUSSION OF DESIGN CONSIDERATIONS

Creating playlists and sharing photos are just two situations among many in which users work with collections. For example, when planning a weekend trip, users

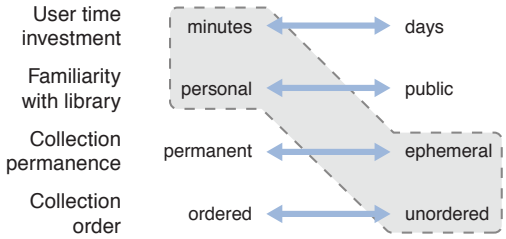


Figure 3.7: *Design space for tasks with collections. SongSelect and PhotoSelect are designed for quickly creating ephemeral collections from personal libraries (shown in grey). SongSelect does allow users to manually order songs in the playlist but order is not used in generating playlists.*

are often faced with putting together an itinerary with many different activities such as meeting friends, visiting museums, and attending events. When selecting investments, users are often trying to pick a good mix of stocks while satisfying some high-level objectives and risk tolerance. When remodeling a house there are many decisions that must be done in concert. For example, the choice of cabinets affects the counters and appliances.

Although in all of these situations, users iteratively create collections while managing multiple constraints, it is important to consider their differences. We propose a design space for discussing the differences and similarities between these problems, illustrated in Figure 3.7.

User time investment

Selecting stocks is a very different kind of activity than creating a music playlist. Although users are trying to make collections in both situations, the high cost of making a bad investment decision means that users are willing to spend days working on their collection. In contrast, creating a music playlist usually takes an hour or two, and some users are only willing to spend minutes on it. With SongSelect and PhotoSelect we designed for shorter tasks, although some of our participants mentioned that they would use SongSelect to edit playlists over time. Automated tools can shorten the time it takes to complete a task, thereby lowering the barrier

to entry. In our interviews many users expressed wanting to create more playlists but being held back by the time investment.

Familiarity

Users approach personal libraries differently than less familiar public libraries, because they know what is available. Similarly, in the two domains that we considered, users can be expected to be familiar with the kinds of metadata available, such as artist names, genres, places, etc. Both SongSelect and PhotoSelect were designed for use with personal libraries, but we found that some participants acquired music at such a high rate (10 new albums per month) that their libraries were a mix between very familiar and unfamiliar songs. Although we believe that the interaction techniques we present in this chapter could apply to unfamiliar content, further investigation is necessary.

Collection permanence

Users create collections for different reasons. Music playlists created for parties are often transient and get dated quickly. Investment and remodeling decisions, on the other hand, tend to be more permanent. SongSelect and PhotoSelect's designs focused on the collection creation aspects rather than longer-term maintenance aspects. We suspect that long-lived collections may require additional tools that help users track how a collection has changed over time.

Item order

For many collections the order of items is important. Although SongSelect lets users manually order items, it does not consider ordering effects when it generates collections. We expected this to be a common complaint among our participants, however we found that many listen to playlists in shuffle mode.

3.7 CONCLUSIONS AND FUTURE WORK

We present a semi-automatic interface for creating collections of items from personal media libraries. Our interface combines free-form text input with example-based refinement. We present a *keyword query interface* that lets users create collections by specifying collection characteristics and propose three different example-based refinement techniques. First, we introduce the *suggestion widget* for in-place addition of new collection items that allows the user to flexibly add content using example metadata. Second, we allow users to *automatically scan through alternatives* for one or more collection items while retaining any user specified collection characteristics. Finally, we use a *two-pane linked interface* to let users dynamically sort and scroll their libraries relative to a collection item. We demonstrate our approach in two applications—a music playlist creation application, SongSelect, and a photo set selection application, PhotoSelect—but we believe these interaction techniques are applicable to other domains. Initial user feedback confirms the need for semi-automated tools that let users direct automatic collection creation.

In future work, we plan to explore working with multiple collections. For example, when users create photo books or calendars, they look at previous photo collections they may have created and shared. Today's interfaces do not let users compare collections and easily see which items are included in multiple collections. More generally, today's folder-based interfaces assume and enforce that items are only present in one location. There are many situations in which this is not optimal, and users make many copies that are hard to keep up-to-date.

SongSelect and PhotoSelect allow users to create collections from their personal libraries, but users are often looking to make collections of non-personal content, like when they build travel itineraries, select stocks, or select furniture for their house remodel. Non-personal content is often scattered throughout multiple websites and part of the user's task is to find all necessary information. A tool for helping users build non-personal collections will have to be able to manage and integrate metadata from multiple locations.

Chapter 4

Perceptual models of viewpoint preference

4.1 INTRODUCTION

What makes for a good view of a 3D object? Different views are not equally effective at revealing shape, and people express clear preferences for some views over others [BVBT99]. Object recognition and understanding depends on both view-independent properties [Bie87] and view-dependent features [KD79]. Researchers have proposed a variety of measures for view point preference, for example viewpoint entropy [VFSH01], silhouette stability [GRMS01], mesh saliency [LVJ05], and symmetry [PSG+06]. We combine attributes like these into an overall measure that we call the *goodness* of views, based on human preference data. While similar approaches based on machine learning algorithms have been previously described, in one case the method only incorporates a single measure [LN08], and in another case the system is designed to produce a very specialized measure trained on a small data set generated by one or few people [VBP+09].

Our goodness measure relies on weights determined via a large user study, in which, given two views of the same object, hundreds of subjects were asked to select a preferred view. The resulting dataset covers each of 16 models with 120 pairs of views, and each pair was evaluated by 30 to 40 people. From this data we can combine attributes together in various predictive models and reliably predict new view selections not used in the training.

Moreover, our methodology offers several benefits beyond the specific goodness measures recommended herein. First, we gain insight by examining the relative effectiveness of various components of our goodness measure, as well as other

methods proposed in the literature. For example, we find that the mesh saliency approach described by Lee *et al.* [LVJ05] is not as effective at describing our data as projected area, a much simpler model. The optimization procedure can incorporate any attribute or combination of attributes *a posteriori* and also evaluate any proposed model for viewpoint preference using only the data acquired in our user study.

This work also considers several straightforward applications for the goodness measure for views. Supposedly simple tasks such as orbiting around a 3D shape (with the implied goal of *understanding* it) are often reduced to a "turntable" with the camera rotating above the equator [Goo10], even where another path might reveal the shape more effectively. We offer several tools motivated by this observation. The first kind of tool automatically selects either good individual viewpoints or an orbit around an object designed to pass through good views as much as possible. Likewise, a second class of tool helps the user navigate in camera space by gently nudging the camera towards good views and away from bad views.

The main contributions of this chapter are:

- an evaluation of 14 attributes from the literature that encode a range of desirable properties w.r.t. understanding and exploring a 3D shape,
- a user study and methodology by which we evaluate and optimize combinations of these attributes,
- a set of simple recommended measures for practical applications,
- a large dataset resulting from the study, publicly available for download by other researchers,
- an optimization tool that finds good individual views of an object or a smooth orbit around an object that passes through good views, and
- an interface that allows users of various skill levels to navigate around 3D shapes using a commodity 2D input device such as a mouse or touch screen, or even 1D input widget such as a scroll-bar.

4.2 RELATED WORK

Our work concentrates on finding good views of a single object. Full 3D camera control is therefore outside of our scope, and we refer to the excellent taxonomy of general methods for camera control in 3D by Christie and Olivier [CON08].

Several researchers have investigated ways of evaluating the goodness of a view. For example, Kamada and Kawai [KK88] attempt to minimize the number of degenerate faces in orthographic projection. Plemenos and Benayada [PB96] describe a measure for goodness of views based on projected area of the model, while Roberts and Marshall [RM98] compute multiple views that, combined, cover the entire surface of the model as well as possible, using an approximation of the aspect graph [KD79]. Scene visibility is also used for camera placement in the work of Fleishman et al. [FCoL99]. Blanz *et al.* [BVBT99] perform studies that show what attributes are important for determining canonical views for humans, following the seminal work by Palmer *et al.* [PRC81] that first introduces the *notion* of canonical views as well as the first such study. In the work of Gooch *et al.* [GRMS01], an optimization process adjusts camera parameters to produce more "artistic" compositions by causing silhouette features to match known compositional heuristics such as "the rule of fifths." Vazquez *et al.* [VFSH01] coin the term *viewpoint entropy*, inspired by Shannon's information theory [Sha48] and based on relative area of the projected faces over the sphere of directions centered in the viewpoint. Sokolov and Plemenos [SP05] use dihedral angles between faces and discrete gaussian curvature. Also inspired by information theory, Lee *et al.* [LVJ05] describe *mesh saliency* and show how it can be used to optimize viewpoint selection. Yamauchi *et al.* [YSY+06] partition the view sphere based on silhouette stability [WW97], and then use mesh saliency to determine the best view in each partition. Finally, Podolak *et al.* [PSG+06] describe a goodness measure based on object symmetries.

In principle, the tools we present in this chapter could make use of any of these goodness measures, and indeed we investigate and combine several of them. To the best of our knowledge, Polonsky *et al.* [PPB+05] were the first to explore a number of different attributes, which they call *view descriptors*. They suggest the

possibility of a combined measure, but leave this as future work. Most closely related to our own is the work of Vieira *et al.* [VBP+09]. They train a support vector machine classifier using a small set of tuples, one for each view, where each consists of a vector containing concatenated goodness values, and a user-provided binary preference for this specific view. They compare to individual goodness measures, and show that using a combination better fits to a wider range of models—an inspiration for our work. Their approach is designed for user interaction on a small set of models with similar objectives, and they leave a validating user study for future work. In contrast, our motivation is to find a goodness measure designed for a broad range of models and applications, and is based on a large user study.

The tools presented in this chapter are designed to work for both static and moving cameras, either under guided user control or as a path designed to offer a good overall view of the object. Barral *et al.* [BDP00] optimize paths for a moving camera so as to provide good coverage of an overall scene, but the resulting paths appear to be unpleasantly jerky. In follow-up work to [YSY+06], Saleem *et al.* [SSBS07] show how to compute smooth animation paths that connect the best viewpoints, and adjust zoom and speed according to the goodness along the path. In comparison, the paths computed by our method are smooth, but additionally optimize the integral of goodness along the path. Recent work by Kwon and Lee [KL08] shows how to optimize a camera path given animated character motion as input, where the goal is to optimally cover the space swept out by the motion.

4.3 MEASURING VIEW GOODNESS

In this section we describe our process for obtaining a measure for the goodness of views. First, we describe a set of view-dependent attributes that we will later combine to form various overall goodness metrics. Next, we present the results of a large user study in which we gather information about subjects' preferences among pairs of nearby views. Finally, we use the data from the study to optimize the weights of our combined goodness measures, and also evaluate the relative contributions of the different attributes.

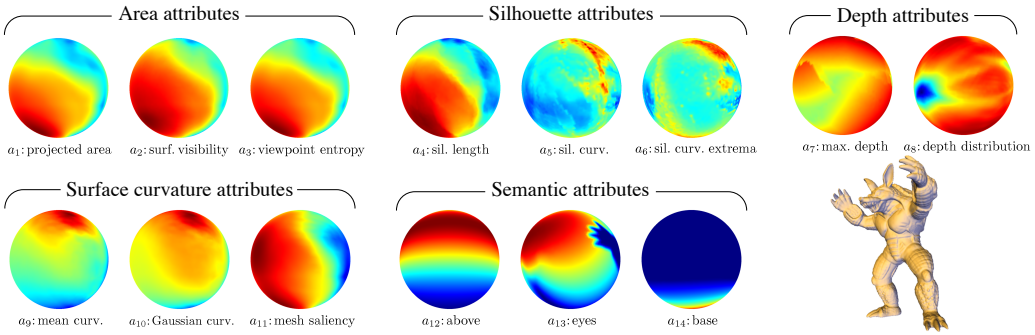


Figure 4.1: *Five groups of attributes, visualized over the sphere of viewing directions, for the armadillo model shown in the lower right. Color values range from blue (low) to red (high).*

4.3.1 Attributes of Views

As reviewed in Section 4.2, the literature offers many attributes of views that may contribute to the overall goodness. However, previous efforts have typically considered just one attribute in forming a goodness measure. Obviously no single measure taken alone fully characterizes what people consider good, and one would expect different measures to combine with differing relative impact overall. In this section we present a group of attributes with the hope that they may combine to form a more accurate measure than any one or a few measures taken alone. These measures are visualized over the sphere of viewing directions for one model in Figure 4.1. Each of the attributes we selected is taken directly from the literature or inspired by previously described attributes. In the presentation below, the attributes are organized into five categories relating to different aspects of a view, for example, surface area or silhouettes.

Area attributes

Area attributes are related to the area of the shape as seen from a particular viewpoint.

a_1: Projected area. This attribute is the projected area of the model in the image plane as a fraction of the overall image area. Introduced by Plemenos and

Benayada [PB96], this measure is generally maximized by non-degenerate views.

a_2: Surface visibility. Plemenos *et al.* [PB96] define the surface visibility as the ratio of visible surface area in a particular view to the total surface area of an object. Maximizing surface visibility should reduce the amount of hidden surface of an object.

a_3: Viewpoint entropy. Introduced by Vázquez *et al.* [VFSH01], this attribute converts projected areas of mesh faces into a probability distribution and measures the entropy of the result. Since the original viewpoint entropy method employed a spherical camera, we use the extension to perspective frustum cameras due to Vázquez and Sbert [VS02].

Silhouette attributes

Silhouette features are believed to be the first index into the human memory of shapes [HS97], and, as a direct result of Shannon's information theory, it was known as early as the 1950s that edges and contours contain a wealth of information about a 3D shape [Att54, KD79].

a_4: Silhouette length. The overall length a_4 of the object's silhouettes in the image plane, expressed in units of the average dimension of the image plane. This attribute is correlated with the appearance of holes and protrusions such as arms or legs.

a_5: Silhouette curvature. Vieira *et al.* [VBP+09] introduce silhouette curvature as an attribute, defined as

$$a_5 = \int |\kappa(\ell)| d\ell$$

where the curvature κ is parameterized by arc length ℓ.

a_6: Silhouette curvature extrema. While silhouette curvatures will capture general complexities in the silhouette of an object, we are often interested in sharp features

such as creases or the tips of fingers. To that end, we introduce a simple measure that emphasizes high curvatures on the silhouette:

$$a_5 = \int \kappa(\ell)^2 d\ell.$$

However, through experimentation we found it best to drop the curvatures found at depth discontinuities (i.e. T-junctions) as these areas sometimes contribute high curvatures without obvious connection to visual interest or features.

Depth attributes

Depth attributes are related to the depth of the shape as seen from a particular viewpoint; similar to area attributes, depth attributes can help avoid degenerate viewpoints.

a_7: Max depth. The maximum depth value of any visible point of the shape is used to avoid degeneracies in [SS02].

a_8: Depth distribution. Since the maximum used in a_7 is noisy, we also introduce an attribute that is designed to encourage a broad, even distribution of depths in the scene:

$$a_8 = 1 - \int H(z)^2 dz$$

where H is the normalized histogram of the depth z of the object, sampled per pixel. This measure becomes small when H is "peaky" (most of the object is at a single depth) and is maximized when H is "equalized" (a range of depths are visible). Thus, a_8 encourages objects with largely planar areas to take oblique rather than head-on views, conforming to a human preference observed by Blanz *et al.* [BVBT99].

Surface curvature attributes

Geometric surface curvatures of the shape are assumed to be related to the shape's semantic features and are easily computed.

a_9: Mean curvature. We compute the mean curvature on the surface of the object using [MDSB02]. We consider curvature magnitudes to be relevant (not generated by noise) if they could be generated by a feature larger than 1% of the object's size. We then linearly map the curvature values into $[0, 1]$ and compute the mean value visible at a particular viewpoint:

$$a_9 = \frac{1}{A_p} \int_{x \in A_p} \left[\frac{|h(x)|}{h_{\max}} \right]_{01} dA.$$

Here, A_p is the projected screen area occupied by the object, $h_{\max}$ is the absolute value of largest relevant curvature, and the operator $[\ * \ ]_{01}$ clamps its argument to the range $[0, 1]$. We use the absolute value of the curvature to avoid cancellations in the integration.

a_{10}: Gaussian curvature. Gaussian curvature is also used in previous work ([PKS+03, PPB+05]); we compute Gaussian curvature on the surface of the object using Meyer *et al.*'s angle defect formula [MDSB02]. We treat the computed Gaussian curvatures analogously to the mean curvatures:

$$a_{10} = \frac{1}{A_p} \int_{x \in A_p} \left[\frac{|k(x)|}{k_{\max}} \right]_{01} dA.$$

a_{11}: Mesh saliency. The final surface curvature attribute is *mesh saliency*, introduced by Lee *et al.* [LVJ05]. Mesh saliency is constructed from the mean curvature of the surface at multiple levels of detail; Lee *et al.* apply this attribute for both mesh simplification and viewpoint selection. As defined by Lee *et al.*, the attribute a_{11} is the *total sum* of mesh saliency visible from a viewpoint. We note that this confounds two factors—average mesh saliency and the projected area measure a_1 described above. While it would probably be wise to decorrelate these two factors, we use a_{11} as described in the literature for easier comparison.

Semantic attributes

Much of the previous work in automatic viewpoint selection has avoided the use of semantic features preferring that view goodness can be fully computed from the

geometry of the object. We include semantic features because we believe that they are important in human preference, and we will be able to measure this importance in Section 4.3.3.

a_{12}: Above preference. Blanz *et al.* [BVBT99] observe that people tend to prefer views from slightly above the horizon. Based on their observation, Gooch *et al.* [GRMS01] initialize their optimization for "artistic" compositions from such a view. Thus attribute a_{12} favors these views with a smooth falloff towards the poles:

$$a_{12} = \mathcal{G}\left(\phi; \frac{3\pi}{8}, \frac{\pi}{4}\right),$$

where ϕ is the latitude with 0 at the north pole and $\frac{\pi}{2}$ at the equator, and $\mathcal{G}(x, \mu, \sigma)$ is the non-normalized Gaussian function $\exp(-(x-\mu)^2/\sigma^2)$. The a_{12} attribute peaks at $\frac{\pi}{8}$ above the equator and is minimal at the south pole. For objects with no inherent orientation, such as the heptoroid and rocker arm models (Figure 4.6), we simply set this term to zero. Nevertheless, typical computer graphics models generated by CAD or acquisition processes do indeed have a stored up-direction, and it is also possible to use techniques like those of Fu *et al.* [FCODS08] to determine the orientation of man-made objects with unknown up directions.

a_{13}: Eyes. When the object of interest is a creature with eyes or a face, we observe that people strongly prefer views where the eyes can be seen [Zus70]. Thus, attribute a_{13} measures how well the eyes of a model can be seen, when appropriate. In our system we mark the eyes by hand, by annotating a central vertex on the surface with a tiny radius. We note that just as technology for automatic face detection in images and video has matured, we expect that analogous algorithms for 3D models will become robust in the future and will obviate this manual task. To measure a_{13}, we simply sum this "eyes" surface value for all visible pixels. Most pixels do not contribute, so the behavior of this attribute is roughly that of a delta function for visibility attenuated by a cosine term for oblique views. For objects without eyes, this attribute is set to zero.

a_{14}: Base. Just as people tend to prefer seeing eyes, they tend to avoid views from directly below for objects that have an obvious base on which they sit. The attribute a_{14}, measures the amount that the hand-marked base is visible, using the same strategy as for eyes. While we mark these features by hand, for many models they could be found using the automatic method of Fu *et al.* [FCODS08]. Note that we distinguish the base from the eyes because we expect their behaviors to be anti-correlated, and because some models will have eyes, some will have base, some will have both, and some neither.

Our implementation uses an image-based pipeline to avoid dependencies on mesh representation. To compute the attributes described above we render the object into an ID, a depth and several color images, and then use image processing to compute projected area, silhouettes, and so forth. For mesh saliency, we use the implementation of Lee *et al.* to assign a value at every vertex as a preprocess. Similarly, we compute the mean and Gaussian curvatures of the surface using the method of Meyer *et al.* [MDSB02]. For any particular view we render a "color" buffer containing these values interpolated across faces and compute the appropriate quantities on the resulting rendered images. The eyes and base attributes are computed in a similar way.

4.3.2 Collecting human preferences

Here we describe a study we performed in order to collect data about the relative goodness of views according to human preferences. In the next section we will use this data to train a model for view goodness that combines the attributes described above. It will also allow us to remark on the relative importance of the individual attributes in forming a combined measure of goodness. In order to design a study to meet these goals, a number of issues need to be addressed.

The first concern is what models to use for the study. Researchers performing perceptual studies often resort to models of abstract shapes like "blobbies" or geons. However, for several reasons we prefer to use models that are more recognizable. First, the resulting data will better characterize the kinds of models that we work with in computer graphics. Second, it is easier for people to express view preferences

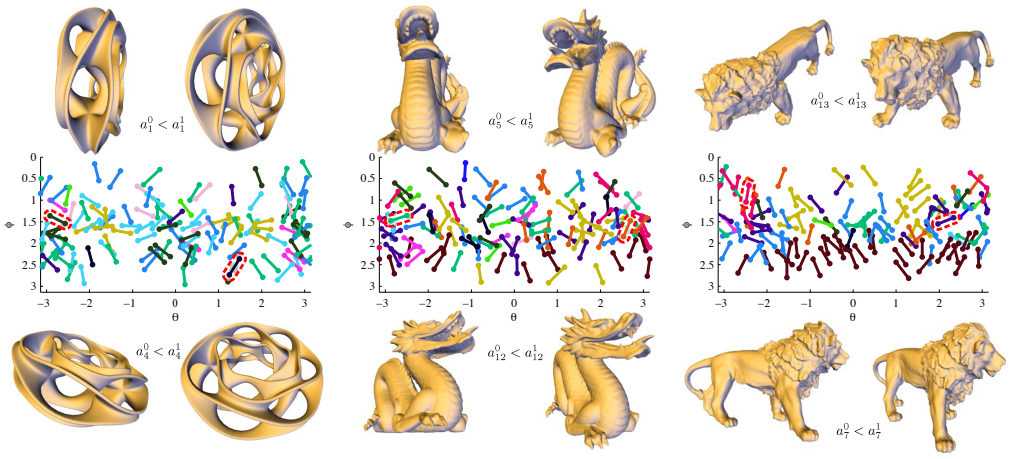


Figure 4.2: *Distribution of the pairs of selected views for three objects used in our study (120 pairs per object). In the middle the pairs are plotted in the $\theta \times \phi$ domain, and colored by the attribute in which they vary most. To illustrate typical image pairs, we highlight pairs that were particularly dominant in a specific attribute—top-left: projected area, bottom-left: silhouette length, top-middle: silhouette curvature, bottom-middle: above preference, top-right: eyes, bottom-right: max. depth. In each plot, the highlighted pair on the left side of the plot corresponds to the image pair at the top, and the pair on the right corresponds to the image pair on the bottom.*

when they understand what they are seeing, so we believe the data will be both more meaningful and less noisy. Nevertheless, we would like the models to represent a broad range of shapes and objects. We selected 16 models, some scanned from real objects and some modeled via software, and they may be seen in the upper four rows of Figure 4.6. Eight of the models will be familiar to graphics researchers (armadillo, dragon, etc.) and the other eight are selected from separate categories in the Princeton Shape Benchmark [SMKF04]. All but one of the objects are recognizable shapes even for people who have never seen them before. The heptoroid model is more abstract but still easy to understand from most or all views.

Our goal is to learn by asking people which views they prefer and by how much. Unfortunately there is no absolute scale for this kind of preference, since one person's "pretty nice" is another person's "okay." Moreover such judgements are not quantitative. A standard strategy in such scenarios is to use the method of

paired comparisons [Dav63], which asks people a simpler question: "Which of these two views do you prefer?"—a two-alternative forced choice experiment (2AFC). In principle, by asking many people this question for many pairs of views it will be possible to establish an overall ranking for all views. Standard practice would be to ask this question for either all pairs or many random pairs. However, in designing our study we found it to be more effective to ask the question only for pairs of *nearby* views. In particular we found that an angular separation of $\frac{\pi}{8}$ radians provides a nice balance in that the views are sufficiently far apart that the difference between the images is obvious, and yet similar enough that it does not feel like an "apples-to-oranges" comparison. We additionally fix the orientation of any rendered view such that the up vector of the model is aligned with the up vector in the image plane, so nearby views typically have similar orientations.

Regardless of how similar or different the views are, the instructions given to the subjects are critical. We want people to consider the shape of the object when choosing a view, but "shape" means different things to different people, especially graphics non-experts. Therefore, inspired by language in the study of Blanz *et al.* [BVBT99], we provide the following instructions to our subjects:

> *Which of the two views of the object shown below reveals its shape better? For example, suppose that you had to choose one of these two pictures to appear in a magazine or product advertisement. Do not worry if neither of them is ideal. Just click on the one that you think is better.*

The next issue to address is how to choose the particular pairs of views to be used in the study. The natural goal is to choose pairs randomly but roughly uniformly distributed over the sphere. We use rejection sampling with a probability distribution that includes a term that discourages choosing views that are close to previously selected views. In addition, we include two more criteria for rejection. First, in order to avoid pairs where the model appears to have rotated substantially in image space, we include a term that discourages pairs from varying much in the longitudinal direction near the poles. This term falls off with latitude so that at the horizon pairs vary equally in θ and ϕ, as can be seen for the sets of pairs shown

in Figure 4.2. Second, if we sample the sphere uniformly, it is possible that many pairs will not exhibit strong differences in the attributes and that for a particular attribute a_j we may not get a range of variation among the pairs. Therefore, we also include a rejection policy that favors pairs of views in which attributes are varying. Specifically we use the sum of absolute differences in each attribute between the pairs, where each attribute is scaled in terms of standard deviations (because they have different ranges of values). Using this strategy we selected 120 pairs of views for each of the 16 models (3,840 images in total). Figure 4.2 shows the pair distributions for three models, colored by the attribute by which each pair varies the most. Of course all attributes vary somewhat for every pair, but it is easy to see that across all pairs every attribute has substantial variation. (See Figure 4.3 for the color coding.)

We ran our study on the Amazon Mechanical Turk (AMT), a service that allows researchers (and others) to provide small jobs for anonymous workers over the Internet for a small amount of money. The use of AMT is increasing in computer graphics and human-computer interaction research, see, for example, [HB10], [DHSC10] and [CSD+09]. In our study, each job ("HIT" in Mechanical Turk terminology) was to make a choice for each of 30 images. For each person, the pairs were presented in random order, and for each pair the two images were randomly shuffled left-right. To filter out careless subjects, we resort to the strategy of Cole *et al.* [CSD+09] in which every pair is shown to the worker twice (show 30 pairs, shuffle, then repeat) and we only retain the data from HITs where the two answers were reasonably consistent. Specifically, we require that 22 or more of the 30 answers were answered the same way the second time. This affords us reasonable confidence ($p > 0.99$) that the worker was not picking randomly, and also keeps data where the user could not make up their mind a substantial fraction of the time.

After discarding the inconsistent data we have between 30 and 40 people expressing a choice in each pair, for a total of 2,119 HIT assignments and 127,140 choices overall. This data was collected from 524 unique workers. Many subjects did multiple HITs, but they were never presented the same pair in more than one HIT. The most active subject worked on 51 HITs, and the histogram falls off roughly with a power law shape where most workers did just one HIT.

4.3.3 Modeling viewpoint preferences

The study produces a set of data that we can now use to evaluate predictive models of viewpoint preference, and to fit models of viewpoint goodness. The data is as follows: each pair i was shown twice to $n_i/2$ people, meaning there were n_i opportunities to pick one image or the other. Let's say that view v_i^0 was picked k_i times, and v_i^1 was chosen $n_i - k_i$ times. If we have a probabilistic model that predicts how often a user would choose v^0 over v^1, call it $P(v^0, v^1) \equiv P(v)$, then we can compute the likelihood that such a model explains our observed data.

Likelihood of observing our data

Out of the n_i people who see pair i, the probability that exactly k_i will choose view v_i^0 is given by the binomial distribution:

$$\binom{n_i}{k_i} P(v)^{k_i} (1 - P(v))^{n_i - k_i}$$

Thus, the likelihood $\mathcal{L}$ of seeing these observations, over all M pairs is:

$$\mathcal{L}\left[P(v)\right] = \prod_i^M \binom{n_i}{k_i} P(v)^{k_i} (1 - P(v))^{n_i - k_i}.$$

As usual when dealing with probabilities, the log-likelihood form is more convenient:

$$\begin{aligned} \mathcal{L}^*\left[P(v)\right] &= \sum_i^M \ln\binom{n_i}{k_i} + \ln(P(v)^{k_i}) + \ln((1 - P(v))^{n_i - k_i}) \\ &= c + \sum_i^M k_i \ln P(v) + (n_i - k_i)\ln(1 - P(v)) \end{aligned}$$

Interpreting likelihood values

To gain some intuition about the range of likelihood values, we can compute the likelihood of two basic predictive models: a *naïve* model that randomly guesses and an *oracle* that has perfect knowledge. Consider first the naïve model that predicts either image with equal likelihood; $P_{\text{naïve}} = 1/2$ and its value of $\mathcal{L}^*_{\text{naïve}}$ is -33203. On

the other end of the spectrum, we can consider an oracle with complete knowledge of the users' selections. When the subjects select view v^0 over v^1 k out of n times, the oracle predicts a probability of k/n for that selection; $P_{\text{oracle}} = k/n$ and $\mathcal{L}^*_{\text{oracle}}$ is −3688 on our data. The naïve predictor and the oracle provide a convenient frame of reference for *any* model's performance: we express the model *fitness* as $F[P] = (\mathcal{L}^*[P] - \mathcal{L}^*_{\text{naïve}})/(\mathcal{L}^*_{\text{oracle}} - \mathcal{L}^*_{\text{naïve}})$. The closer a model's fitness approaches unity, the better it predicts the response of our users.

Goodness functions for single viewpoints

The preceding discussion focussed on models that operate on *pairs* of viewpoints; while this matches our collected data, for practical applications we would prefer a model that predicts the goodness of a *single* viewpoint. Given goodness values for single viewpoints $G^0 \equiv G(v^0)$ and $G^1 \equiv G(v^1)$, we need a prediction of how often a user would choose v^0 over v^1. There are many possible models for this response and, generally, this method of paired comparison has been an active area of research since the 1920's [Dav63]. The well-studied Bradley-Terry model [BT52] characterizes the probability P that a person will choose v^0 over v^1 as the sigmoid-shaped logistic function of the difference in their inherent goodnesses: [1]

$$P(G^0, G^1) \quad = \quad \frac{1}{1 + e^{-\sigma(G^0 - G^1)}}$$

Examining Figure 4.3, note that many of the attributes already have sigmoid-like shapes, indicating some weak explanatory power, even individually.

Given a particular model of viewpoint goodness G such as a single attribute, or a weighted combination of various attributes, we can compute the probability $P(G^0, G^1)$ that a user would select the first view, then evaluate the likelihood $\mathcal{L}^*[P(G^0, G^1)]$ that this model explains our data, and finally assign a fitness value $F[G]$ relative to the performance of the oracular and naïve models. Schematically, we have:

$$G \xrightarrow{\text{B-T model}} P \xrightarrow{\text{user data}} \mathcal{L}^* \xrightarrow{\text{oracle \& naïve}} F$$

[1] Though we refer the reader to [BT52] for the details, the Bradley-Terry model follows from the intuitive idea that if view 0 has goodness G^0 and view 1 has goodness G^1, then the probability that v^0 is picked over v^1 is $G^0/(G^0 + G^1)$.

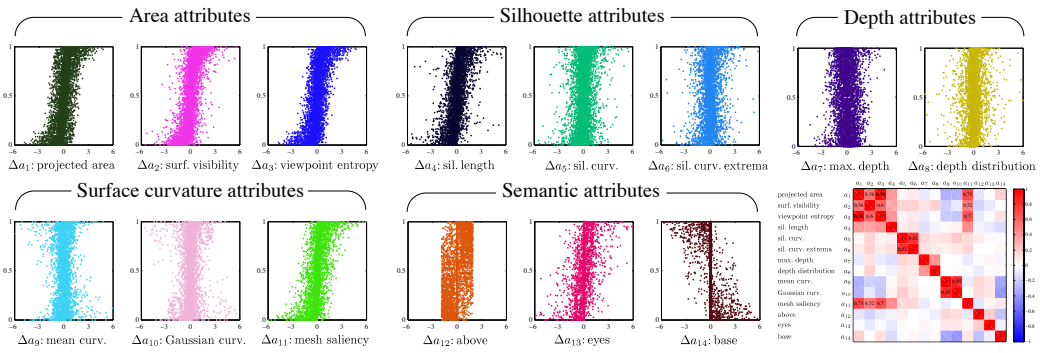


Figure 4.3: *Plots of the difference in each attribute value in a viewpoint pair versus the user's preference for the first image in the pair. The horizontal axes are in units of standard deviations of the differences in attribute value across all pairs of images. The matrix in the lower-right shows the linear correlation coefficients between pairs of attributes, with the values of highly-significant correlations marked. Note the three strong clusters: area attributes (projected area, surface visibility, viewpoint entropy and mesh saliency), silhouette curvature attributes (silhouette curvature and silhouette curvature extrema), and surface curvature attributes (Gaussian curvature and mean curvature).*

We now explore various models of viewpoint goodness, with the intent of discovering important attributes and providing the practitioner with practical models.

Single-attribute models of goodness

As described in Section 4.3, each viewpoint is associated with $N = 14$ attributes. Figure 4.3 shows raw fitnesses of the individual attributes. We first explore the simplest models of viewpoint goodness: goodness is given by a single weighted attribute: $G_i = a_i$. We can fit the value of σ in the Bradley-Terry model to the data by any convenient 1D optimization procedure. To avoid over-fitting the data we use 100 trials of random sub-sampling validation: in each trial the weights were trained on a random subset of half of the objects and tested against the other half. The final weights are the means of the 100 trials, and are shown in Table 4.1. We can see that surface visibility plays the single strongest predictive role; on the other hand, silhouette curvature does not appear to do as well on its own. However, when

Table 4.1: *Fits of individual attributes to our study data. Fitnesses that are listed as exactly zero are statistically indistinguishable from zero at a significance level of $p = 0.05$, the rest are all significant.*

Model	F	$\sigma(F)$
Oracle	**1**	0
Surface visibility	**0.38**	0.072
Viewpoint entropy	0.37	0.092
Projected area	0.28	0.110
Mesh saliency	0.26	0.100
Sil length	0.20	0.120
Above	0.16	0.064
Base	0.13	0.087
Eyes	0.09	0.033
Sil curvature	0.01	0.040
Sil curvature extrema	0	0.032
Depth distribution	0	0.020
Max depth	-0.02	0.021
Abs mean curvature	-0.15	0.260
Abs Gaussian curvature	-0.15	0.180
Naïve	**0**	0

combined with other attributes, it may perform quite differently: see Section 4.3.3.

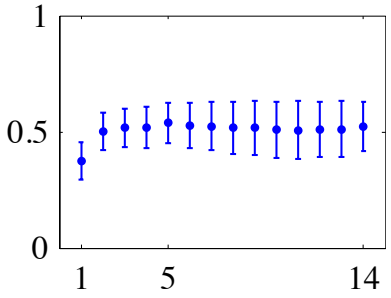
Linear models of goodness

The natural extension to the single-attribute models are the *linear-K* models that combine K attributes to form a goodness value:

$$G(v) = \sum_{j \in S} w_j a_j$$

where v is a viewpoint and S is the set of indices of attributes used in the particular model ($|S| = K$). Since σ in the Bradley-Terry model is made redundant by w_1, we fix it to be one. We can then optimize the values of the weights by maximizing the value of $F[G_i]$, which is nonlinear in the unknown weights w_i and the known

attributes a_i of the views. However, the function is quite smooth and the downhill simplex method of Nelder and Mead [NM65] find the optimal weights consistently and quickly for our data. There are $\binom{14}{2} = 91$ linear-2 models, $\binom{14}{3} = 364$ linear-3 models, etc., for a total of 16383 possible linear models. We separately trained and tested all 16383 models using, as before, 100 trials of repeated random sub-sampling. The result is a distribution of fitnesses for each potential linear model, sampled 100 times. Given the statistical nature of the sampling, it is inappropriate to, say, simply select the model with the highest mean energy for some particular K: another run of 100 trials of training and testing might result in a slightly different ranking. Instead, we use Tukey's "honestly significant difference" (HSD) method, a multiple-comparison procedure [Hsu96], to identify the *pool* of models that perform statistically indistinguishably from the top-performing model. Shown on the right is the mean and standard deviations of the fitnesses of the pool of top-performing models for $K = 1 \ldots 14$. Note that using more than five attributes does not improve the performance of the linear models.



Given that there are several top-performing linear models with K attributes, we recommend the following single-attribute, linear-3 and linear-5 models, spanning the useful range of K (Table 4.2). In each class K, these recommended models are in the pool of highest-performing models, and are chosen with an eye towards computational simplicity of the component attributes. The simplest model is simply a_2, surface visibility, and if more computational resources are possible, then we add a_{12} and a_4, the above preference and silhouette length, respectively. Still better performance is possible by adding a_1 and a_7, projected area and max. depth. If it is possible to include marked features such as eyes or base, we also suggest an alternative model, *linear-5b*, which swaps a_7 for a_{13}, the eyes attribute. Table 4.3 lists the performance of our recommended models. We use the linear-5b model for the rest of the discussion that follows and in Figures 4.4, 4.6, 4.7, 4.8 and 4.9.

Figure 4.4 shows how the optimized weights of the linear-5b model fit the

Table 4.2: *Weights of viewpoint attributes for our recommended models of viewpoint goodness. Note that, since each attribute is scaled differently, the absolute values of weights are meaningless and comparing weights* across attributes *is not possible.*

	a_2	a_{12}	a_4	a_1	a_7	a_{13}
Single	23					
Linear-3	18	2.8	0.51			
Linear-5	14	2.7	0.46	14	2.5	
Linear-5b	15	2.6	0.42	13		670

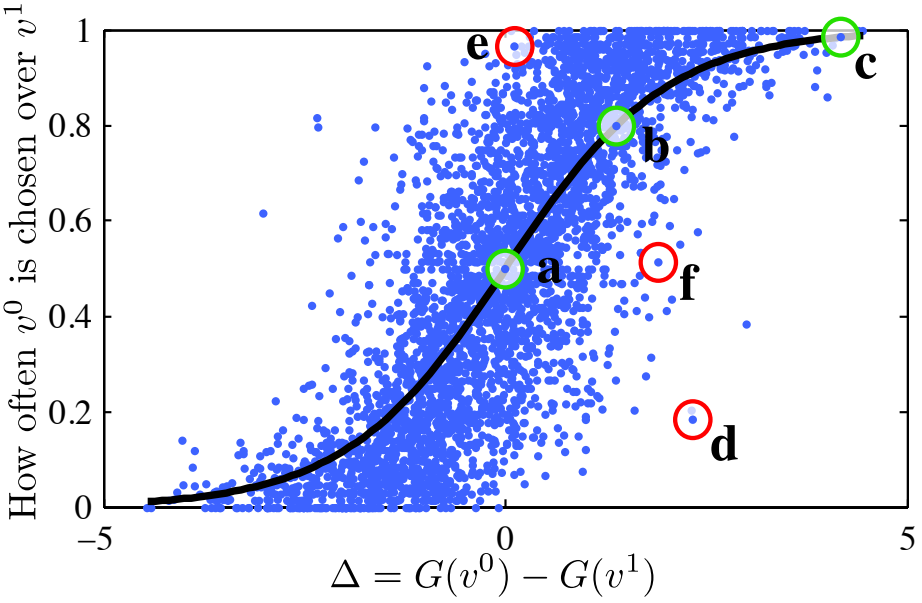


Figure 4.4: *Fit of differences in goodness values in the linear-5b model to observed user selections for 1920 pairs of viewpoints across 16 models. $G(v)$ is a linear combination of viewpoint measures fit to the probability model of Section 4.3.3. The points highlighted in green are particularly well-predicted by the model, while the points in red are not; the labels correspond to viewpoints in Figure 4.5.*

user's preference data. Overall, the sigmoid shape of the logistic curve appears to fit the shape of the data well. While the "slope" of the curve appears to be slightly shallower than that of the data, this can be explained by the nature of the probabilities associated with the binomial distribution—that "errors" near $\Delta = 0$ are more easily explained, in a probabilistic sense, than those out in the tails of the curve. Examples of successes and failures of the model are marked in Figure 4.4 and the corresponding images are shown in Figure 4.5.

Quadratic models of goodness

We note that there is an obvious gap between the performance of our linear-5b model and that of the oracle. It is natural to wonder whether some other model might perform better. Perhaps the oracle is unapproachable—after all, it is unrealistic because *by definition* the oracle knows the answer for any user preference! One strategy would be to consider attributes other than the 14 that we have evaluated for the views used in our dataset, which we leave for future work. However, we can ask if there are other models that would use our attributes to better fit the data.

Table 4.3: *Performance of predictive models. All models were tested using 100 trials of random sub-sampling validation: in each trial the models were trained on a random subset of half of the objects and tested against the other half. The resulting mean and standard deviation of test fitnesses are reported.*

Model	$\langle F \rangle$	$\sigma(F)$
Oracle	**1**	0
10-NN	0.77	0.020
Quadratic	0.63	0.090
Linear-5b	**0.58**	0.060
Linear-5	**0.58**	0.060
Linear-3	0.55	0.062
Single	0.38	0.072
Naïve	**0**	0

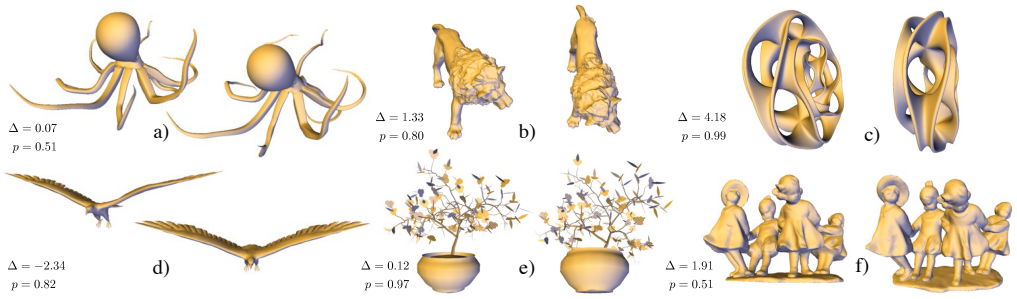


Figure 4.5: *Selected viewpoint pairs corresponding to the labelled points in Figure 4.4. In cases a) and f), the users expressed no clear preference, while in the other cases they preferred the right image. The upper row (a, b, c) shows example pairs of images that are well-predicted by our model, ordered by increasing values of predicted and actual goodness. The lower row shows pairs of images that are not well-predicted by our model. Our model predicts that users will select the left-most image of pair d) less than 20% of the time, but it was actually selected in over 80% of the trials; an anti-correlation. Our model predicts no preference for either view of pair e), yet users nearly always selected the left view. Finally, pair f) is overwhelming predicted to choose the left-most image, but users chose both images equally. The right-most image allows the second child's head to be more clearly seen, a quality that our attributes do not capture.*

A potential concern of a linear model is that it cannot characterize correlations between the attributes (see Figure 4.3 for evidence of their correlations). Therefore we consider a quadratic model as follows:

$$G(v_i^m) = \sum_j^N w_j a_{ij}^m + \sum_{jk}^N w_{jk} a_{ij}^m a_{ik}^m$$

This model has 14 + 105 = 119 weights, and, when fit using the same procedure as before, has a mean fitness of $F = 0.63$ (Table 4.3), a moderate improvement over the various linear models.

Non-parametric models of preference

We also experimented with non-parametric fitting of the viewpoint preference data. In particular, we applied the *K-nearest-neighbors* model for various values

of K [FH51]. We again randomly split the data into a training set and a test set by partitioning the data associated with a random subset of half the models. To find the likelihood of each viewpoint in the testing set, we first computed the attribute deltas for all 14 attributes, converted to units of standard deviations, then found the K-nearest neighbors in the training set using Euclidean distance. Once the K-nearest neighbors are found, the likelihood is computed in the same way as with the oracle, but averaged over the K-nearest neighbors. The performance of this model is shown for $K = 10$ in Table 4.3, averaged over 100 trials (similarly to the linear-K models, values of K greater than 10 did not improve performance). While the performance of this model is better than the linear or quadratic models, it is onerous to compute: to predict the preference for one view in a viewpoint pair, all 14 attributes must be computed for the two viewpoints and the minimum distance must be computed to the training set of 960 viewpoint pairs. In addition, the K-nearest-neighbors model does not directly provide the goodness for a single viewpoint, only the preference in a viewpoint pair.

Of course, any number of other models from machine learning might perform even better, but at the expense of both computational complexity and loss of intuition. Thus, for applications and results described in the remainder of the chapter, we employ the linear-5b model. For example, based on this goodness model we show in Figure 4.6 the best view for each of the 16 shapes used in our study, and for 4 shapes not used in our study.

4.4 APPLICATIONS

In lieu of virtual hands and clay, most complex tasks in three dimensions, such as shape modeling or camera navigation, are generally performed in two dimensions when using contemporary computer hardware. True 3D input devices [Zha98] are not widely deployed, perhaps because of the simplicity and commercial success of commodity 2D input devices such as the mouse or, more recently, multi-touch panels. At the same time, the complexity of 3D modeling packages [Aut10b, Aut10a] has grown to a point where one requires significant training and practice to carry out even the most mundane tasks.

Figure 4.6: *The best view according to the six attribute model for each of the 16 training models (rows 1–4) and four additional model (last row).*

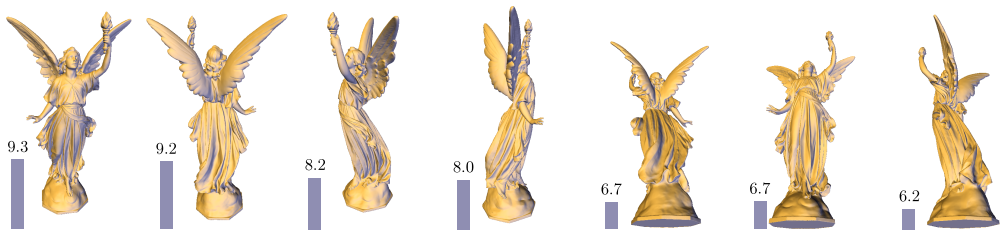


Figure 4.7: *The seven best views of the Lucy model, selected using the mean shift algorithm on the linear-5b model. Using the mean shift algorithm on the viewing sphere, the number of views do not need to be pre-selected. The goodness value of each view is displayed on the left: note the clear distinction between the various types of views.*

In the following, we propose several applications that use our goodness measure (or any other) to assist a user in finding good views and navigating around a 3D model.

4.4.1 Finding the N-best views

A straightforward application that uses our goodness measure is finding the N-best views of a model, as proposed and demonstrated in [PPB+05, YSY+06, VBP+09]. Instead of picking a fixed number of best views, we have decided to find those views that are most representative, and therefore employ *mean-shift clustering* [CM02] to find the dominant peaks in our goodness function. As a result, depending on the model chosen, we obtain varying numbers of best views. As can be seen in the distribution of good views over the viewing sphere, low frequency goodness functions result in only few representative views (Figure 4.7), whereas high frequency functions require a larger number of views to reveal all important features of the shape.

4.4.2 Periodic orbits and scrubbing

Closed-loop viewpaths provide a convenient 1D user interface to viewing the salient features of a 3D model, removing the complexities of the standard trackball interface. The user can *scrub* the viewpoint along the viewpath by means of a standard scroll

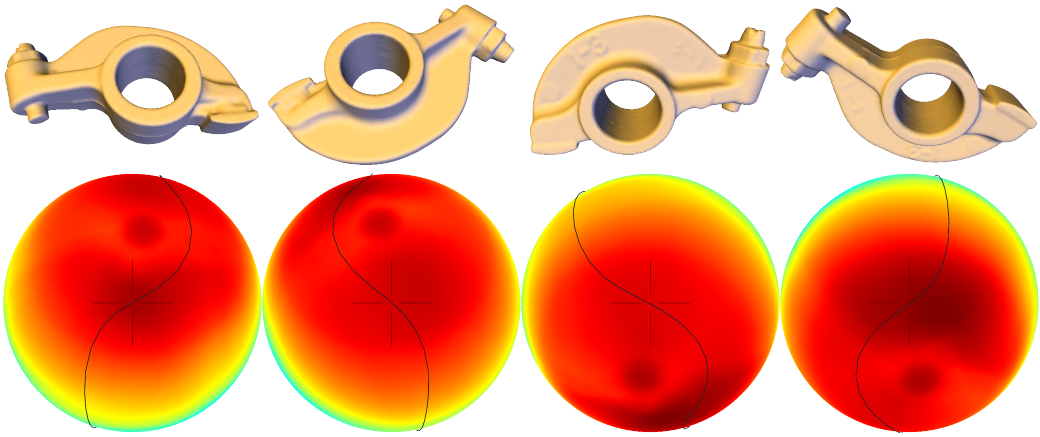


Figure 4.8: *A closed, periodic camera orbit passing through the best views of the rocker arm object.*

bar widget or by linear mouse drags. This simplified 1D interface is appropriate for applications where the user might want to quickly preview the features of a model, for example, when browsing a large database of 3D models. In this scenario, models are often explored simply by rotating around the equator [Goo10].

The method presented by Barral *et al.* [BDP00] computes a greedy path that suffers from visibly jaggy motion. Saleem *et al.* [SSBS07] correct this by smoothly connecting stable and salient viewpoints. We go a step further, and compute a path such that the integral of goodness along the path is maximized. Similar to [SSBS07], we initialize a closed loop such that it passes through regions of high goodness using our N-best views algorithm described above. The path optimization then proceeds by optimizing a snake [KWT88] on the view sphere, guided by the gradient of our goodness measure. To ensure that the path is of controllable length and smoothness, we use the common energy terms for stretch and bending. The result is a closed, periodic loop that passes through all good views, and can either be animated, or explored with a 1D scrub bar (Figure 4.8).

4.4.3 Trackball extensions

Finally, we can further enhance the traditional trackball interface with a goodness-based force model that gently guides the user towards good viewpoints and tries to keep them away from bad viewpoints. We term the two modes of the trackball *grab*, which is the mode of holding the mouse button down and moving the model, and *throw*, which is the animation path the model camera describes after the mouse has been released. These two modes of interaction are treated separately: while strong guidance can be perceived as a significant disturbance during the grab operation, the throw path can be adjusted with a larger force.

Grab. As the user rotates the trackball, we apply a *nudge* force in the direction of and proportional to the gradient of the goodness function, but only use the component which is orthogonal to the current direction of motion. The summed force is scaled down to ensure that the nudged camera travels the same distance than the original camera would without the nudge. This adds a small resistance to motions that would travel to bad viewpoints and eases motions towards good viewpoints. The nudge force is small in all cases, but increases with the speed of the user's input, similar to how mouse acceleration depends on input speed in current desktop systems. The expectation is that when a user applies large, fast, coarse motions, they will end up at a good viewpoint more often than not. The results of this guidance force can be seen in Figure 4.9 (top). Note that the adjustments are locally subtle, but add up to large displacements in the direction of good viewpoints.

Throw. In the standard trackball interface, once the mouse button is released, the animation around the model proceeds linearly, with the last rotation speed applied during grabbing. By adding our nudge force described previously, we can guide the animations towards better views. We furthermore add in some friction force that is inversely proportional to the goodness of the current viewpoint, so as to slow down near good viewpoints. To avoid overshooting good viewpoints, we pre-compute the entire animation path directly after switching from grab to throw, and search for the first point along that path where the camera would turn π away from the

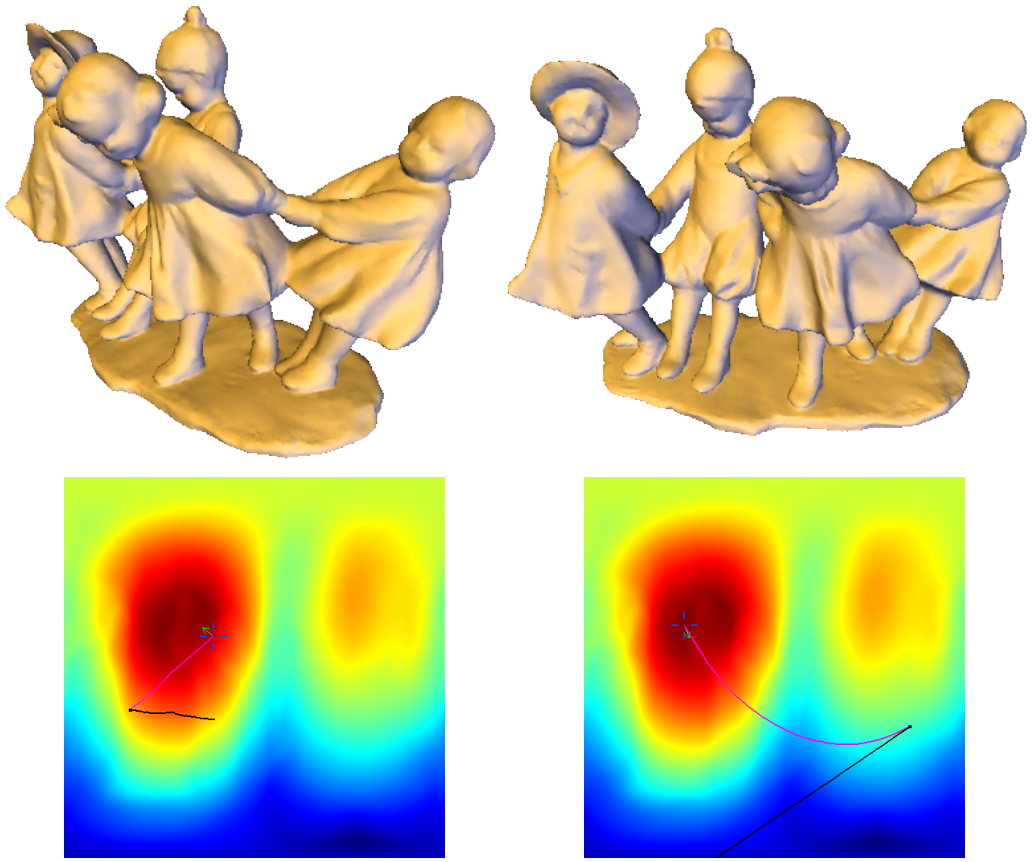


Figure 4.9: *Top: the trackball gentle nudges the viewpoint towards better views while the user is dragging with the mouse. Bottom: if the user "throws" the trackball, the path is attracted by nearby high-quality viewpoints. The black line shows the original path without nudging, while the pink path shows the nudged viewpoint path experienced by the user.*

initial throw direction (the turn-around point). From there we perform gradient descent to find the nearest off-path local maximum of our goodness function, and warp the path, from the turn-around point to the end, such that it ends in the best view. The adjustment to the animation path is more pronounced when compared to the modification during grabbing (Figure 4.9, bottom), but we can guarantee that the animation comes to a stop in good views. We expect that this will especially beneficial for model exploration on devices with constrained touch input, such as Apple's iPhone or iPad devices.

4.5 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

We have presented a perceptual model of viewpoint preference, and while our model performs well for the task carried out in our user study, we see this as the starting point for a much larger research effort. The method as proposed does have some limitations, and these generally identify interesting areas for future work.

We can only make quantitative claims about the 16 models for which we have collected data. While we picked models that have a range of visual features and qualities, there are certainly more classes of models to be explored. Furthermore, we are only searching over two camera parameters, which surely benefits the optimization procedure of Section 4.3.3. So, in addition to evaluating over a larger set of models, we also hope to search over more camera parameters, such as field-of-view, up vector, velocity, and the camera position in complex scenes [CON08].

There are many potential attributes of viewpoint goodness described in the literature, and we did not include every such attribute. In ongoing work, we hope to include broader classes of attributes. For example, Podolak *et al.* [PSG+06] observe object symmetries can play an important role in selecting viewpoints. But also attributes such as lighting variation, occluding contours, texture and others could be included. It is important to note though, that the fact that silhouette curvature is surprisingly unimportant after model fitting, and that surface visibility is by far the most influential metric, points to some significant redundancies in *any* combined measure of viewpoint goodness. As a result, we eventually discarded some measures that initially appeared promising, such as "number of disconnected silhouette loops"

which are heavily correlated with measures such as silhouette length. Thus, a more robust methodology for investigating of the correlations between various attributes is merited.

And finally, while our fitted model appears to be reasonable, it is error-prone and we have identified some of the most egregious error modes in Figures 4.4 and 4.5. This does point to the possibility that special-case combinations of measures will create better combined goodness measures for classes of models.

Conclusion

We have presented the *selection problem* in user interfaces and described several useful techniques and interfaces for easing the burden posed to users. Chapter 2 lays out our ethnographic research into the selection of photo highlights with home users and suggests a semi-automated system for assisting the user. Chapter 3 expands on Chapter 2 by examining a second domain, the creation of musical playlists, and extends our recommendation system with a simple query language for creating collections. Finally, Chapter 4 considers a different, rather more general, domain: selecting camera viewpoints for 3D objects. In addition to the future work discussed in the conclusion sections of Chapters 2–4, we believe that exploring other domains (such as selecting a stock portfolio or generating vacation plans) will be fruitful.

Bibliography

[Ado10a] Adobe AIR application runtime library.
http://www.adobe.com/products/air/, 2010.
Retrieved June 21, 2010.

[Ado10b] Adobe Flex web application softare development.
http://www.adobe.com/products/flex/, 2010.
Retrieved June 21, 2010.

[App10] Apple iTunes Genius.
http://www.apple.com/itunes/features/#genius, 2010.
Retrieved March 29, 2010.

[Att54] Fred Attneave.
Some informational aspects of visual perception.
Psychological Review, 61(3):183–193, 1954.

[Aut10a] Autodesk 3ds Max.
http://www.autodesk.com/3dsmax, 2010.

[Aut10b] Autodesk Maya.
http://www.autodesk.com/maya, 2010.

[BDP00] P. Barral, G. Dorme, and D. Plemenos.
Visual understanding of a scene by automatic movement of a camera.
Proc. Eurographics 2000, 2000.
Short paper.

[Bed01] Benjamin B. Bederson.
PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps.

In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Papers: Information visualization, pages 71–80, 2001.

[Bie87] Irving Biederman.
Recognition-by-components: A theory of human image understanding.
Psychological Review, 94:115–147, 1987.

[BMH06] Frank Bentley, Crysta Metcalf, and Gunnar Harboe.
Personal vs. commercial content: the similarities between consumer use of photos and music.
In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 667–676, New York, NY, USA, 2006. ACM.

[BT52] R.A. Bradley and M.E. Terry.
Rank analysis of incomplete block designs, i. the method of paired comparisons.
Biometrika, 39:324–345, 1952.

[BVBT99] V. Blanz, T. Vetter, H.H. Bülthoff, and M.J. Tarr.
What object attributes determine canonical views?
Perception, 24:575–599, 1999.

[BWR+05] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller.
Automation and customization of rendered web pages.
In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 163–172, New York, NY, USA, 2005. ACM.

[CA09] Zeina Chedrawy and Syed Sibte Abidi.
A web recommender system for recommending, predicting and personalizing music playlists.

In *WISE '09: Proceedings of the 10th International Conference on Web Information Systems Engineering*, pages 335–342, Berlin, Heidelberg, 2009. Springer-Verlag.

[CJJ04] Sally Jo Cunningham, Matt Jones, and Steve Jones.
Organizing digital music for use: an examination of personal music collections.
In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR 2004)*, 2004.

[CM02] D. Comaniciu and P. Meer.
Mean shift: A robust approach toward feature space analysis.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5):603–619, 2002.

[CMS99] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors.
Readings in information visualization: Using vision to think.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[CON08] Marc Christie, Patrick Olivier, and Jean-Marie Normand.
Camera control in computer graphics.
Computer Graphics Forum, 27(8):2197–2218, 2008.

[Cra09] Craigslist.
`http://newyork.craigslist.org/`, 2009.
Retrieved March 25, 2009.

[CSD+09] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh.
How well do line drawings depict shape?
In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, volume 28, August 2009.

[Dav63] H. A. David.
The Method of Paired Comparison.
Hafner Publishing Co., 1963.

Bibliography

[dEW06] Marie desJardins, Eric Eaton, and Kiri L. Wagstaff.
Learning user preferences for sets of objects.
In *ICML '06: Proceedings of the 23rd international conference on machine learning*, pages 273–280, New York, NY, USA, 2006. ACM.

[DHSC10] Julie S Downs, Mandy B Holbrook, Steve Sheng, and Lorrie Faith Cranor.
Are your participants gaming the system?: Screening mechanical turk workers.
CHI '10: Proceedings of the 28th international conference on Human factors in computing systems, pages 2399–2402, 2010.

[DWR+04] Steven M. Drucker, Curtis Wong, Asta Roseway, Steven Glenner, and Steven De Mar.
MediaBrowser: Reclaiming the shoebox.
In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 433–436, New York, NY, USA, 2004. ACM Press.

[FBA00] Adam M. Fass, Eric A. Bier, and Eyton Adar.
PicturePiper: Using a re-configurable pipeline to find images on the web.
In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 51–62, New York, NY, USA, 2000. ACM.

[FCODS08] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer.
Upright orientation of man-made objects.
ACM Transactions on Graphics, 27(3):42:1–42:7, August 2008.

[FCoL99] Shachar Fleishman, Daniel Cohen-or, and Dani Lischinski.
Automatic camera placement for image-based modeling.
Computer Graphics Forum, 19:12–20, 1999.

[FH51] E. Fix and J.L. Hodges.

Discriminatory analysis, nonparametric discrimination: Consistency properties.
Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
Unpublished technical report; see [SJ89] for a re-printed version with commentary.

[FKP+02] David Frohlich, Allan Kuchinsky, Celine Pering, Abbe Don, and Steven Ariss.
Requirements for photoware.
In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 166–175, New York, NY, USA, 2002. ACM Press.

[FTKW08] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder.
CueFlik: Interactive concept learning in image search.
In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 29–38, New York, NY, USA, 2008. ACM.

[GAW04] Andreas Girgensohn, John Adcock, and Lynn Wilcox.
Leveraging face recognition technology to find and organize photos.
In *MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 99–106, New York, NY, USA, 2004. ACM.

[Gec10] Gecode generic constraint development environment.
http://www.gecode.org/, 2010.
Retrieved March 29, 2010.

[GGMPW02] Adrian Graham, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd.
Time as essence for photo browsing through personal digital libraries.

In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 326–335, New York, NY, USA, 2002. ACM.

[Goo10] Google 3D Warehouse and SketchUp.
http://sketchup.google.com/3dwarehouse, 2010.

[GRMS01] B Gooch, E Reinhard, C Moulding, and P Shirley.
Artistic composition for image creation.
Eurographics Workshop on Rendering, pages 83–88, 2001.

[GSW+08] Andreas Girgensohn, Frank Shipman, Lynn Wilcox, Thea Turner, and Matthew Cooper.
MediaGLOW: Organizing photos in a graph-based workspace.
In *IUI '09: Proceedingsc of the 13th international conference on Intelligent user interfaces*, pages 419–424, New York, NY, USA, 2008. ACM.

[HB10] J Heer and M Bostock.
Crowdsourcing graphical perception: Using mechanical turk to assess visualization design.
Proceedings of Computer Human Interaction (CHI 2010), 2010.
ACM CHI 2010 Best Paper Nominee.

[HBB07] O. Hilliges, D. Baur, and A. Butz.
Photohelix: Browsing, sorting and sharing digital photo collections.
In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pages 87–94, 2007.

[HDBW05] David F. Huynh, Steven M. Drucker, Patrick Baudisch, and Curtis Wong.
Time Quilt: Scaling up zoomable photo browsers for large, unstructured photo collections.
In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1937–1940, New York, NY, USA, 2005. ACM Press.

[Hea09] Marti A. Hearst.
Search User Interfaces.
Cambridge University Press, New York, NY, USA, 2009.

[HG09] Derek L. Hansen and Jennifer Golbeck.
Mixing it up: recommending collections of items.
In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1217–1226, New York, NY, USA, 2009. ACM.

[HNS+04] Susumu Harada, Mor Naaman, Yee J. Song, Qianying Wang, and Andreas Paepcke.
Lost in memories: Interacting with photo collections on PDAs.
In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 325–333, New York, NY, USA, 2004. ACM Press.

[HS97] Donald D. Hoffman and Manish Singh.
Salience of visual parts.
In *Cognition*, volume 63(1), pages 29–78, 1997.

[Hsu96] Jason Hsu.
Multiple Comparisons: Theory and Methods.
Chapman and Hall/CRC, 1996.

[KD79] J.J. Koenderink and A.J. van Doorn.
The internal representation of solid shape with respect to vision.
Biol. Cybern., 32:211–216, 1979.

[KK88] T. Kamada and S. Kawai.
A simple method for computing general position in displaying three-dimensional objects.
Comput. Vision Graph. Image Process., 41(1):43–56, 1988.

[KL08] Ji-yong Kwon and In-Kwon Lee.
Determination of camera parameters for character motions using motion area.

The Visual Computer, 24(7):475–483, July 2008.

[KPC+99] Allan Kuchinsky, Celine Pering, Michael L. Creech, Dennis Freeze, Bill Serra, and Jacek Gwizdka.
FotoFile: A consumer multimedia organization and retrieval system.
In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 496–503. ACM Press, 1999.

[KPSW06] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer.
Combining audio-based similarity with web-based data to accelerate automatic music playlist generation.
In *In Multimedia Information Retrieval*, pages 147–154, 2006.

[KPT+08] Sean Kandel, Andreas Paepcke, Martin Theobald, Hector Garcia-Molina, and Eric Abelson.
PhotoSpread: A spreadsheet for managing photos.
In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1749–1758, New York, NY, USA, 2008. ACM.

[KS03] Hyunmo Kang and Ben Shneiderman.
MediaFinder: An interface for dynamic personal media management with semantic regions.
In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 764–765, New York, NY, USA, 2003. ACM Press.

[KSRW06] David Kirk, Abigail Sellen, Carsten Rother, and Ken Wood.
Understanding photowork.
In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 761–770, New York, NY, USA, 2006. ACM Press.

[Kum92] Vipin Kumar.
Algorithms for constraint-satisfaction problems: A survey.
AI Magazine, 13(1), 1992.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos.

Snakes: Active contour models.
International Journal of Computer Vision, 1(4):321–331, 1988.

[LLC+07] Greg Little, Tessa A. Lau, Allen Cypher, James Lin, Eben M. Haber, and Eser Kandogan.
Koala: Capture, share, automate, personalize business processes on the web.
In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 943–946, New York, NY, USA, 2007. ACM.

[LM06] Greg Little and Robert C. Miller.
Translating keyword commands into executable code.
In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 135–144, New York, NY, USA, 2006. ACM.

[LN08] Hamid Laga and Masayuki Nakajima.
Supervised learning of salient 2D views of 3D models.
The Journal of the Society for Art and Science, 7(4):124–131, 2008.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs.
Mesh saliency.
In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 659–666, 2005.

[MCB+08] Robert C. Miller, Victoria H. Chou, Michael Bernstein, Greg Little, Max Van Kleek, David Karger, and mc schraefel.
Inky: A sloppy command line for the web with rich visual feedback.
In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 131–140, New York, NY, USA, 2008. ACM.

[MDSB02] M Meyer, M Desbrun, P Schröder, and A Barr.
Discrete differential-geometry operators for triangulated 2-manifolds.

VisMath '02 Proceedings, Jan 2002.

[Mus10] Musicovery.
http://musicovery.com, 2010.
Retrieved June 22, 2010.

[NM65] J. A. Nelder and R. Mead.
A simplex method for function minimization.
Computer Journal, 7:308–313, 1965.

[PB96] Dimitri Plemenos and M Benayada.
Intelligent display in scene modeling: New techniques to automatically compute good views.
Proceedings of GraphiCon (1996), 1996.

[PCF03] J. C. Platt, M. Czerwinski, and B. A. Field.
PhotoTOC: Automatic clustering for browsing personal photographs.
In *Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing*, volume 1, pages 6–10 Vol.1, 2003.

[PKS+03] D Page, A Koschan, S Sukumar, B Roui-Abidi, and M Abidi.
Shape analysis algorithm based on information theory.
International Conference on Image Processing (ICIP 2003), 1:29–32, 2003.

[Pla00] J.C. Platt.
AutoAlbum: Clustering digital photographs using probabilistic model merging.
In *Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries*, pages 96–100, 2000.

[PPB+05] Oleg Polonsky, Giuseppe Patane, Silvia Biasotti, Craig Gotsman, and Michela Spagnuolo.
What's in an image: Towards the computation of the best view of an object.

The Visual Computer, 21(8-10):840–847, 2005.

[PRC81] S Palmer, E Rosch, and P Chase.
Canonical perspective and the perception of objects.
Attention and Performance IX, pages 135–151, 1981.

[PSG+06] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser.
A planar-reflective symmetry transform for 3D shapes.
ACM Transactions on Graphics (Proc. SIGGRAPH), 25(3), July 2006.

[PVV08] Steffen Pauws, Wim Verhaegh, and Mark Vossen.
Music playlist generation by adapted simulated annealing.
Inf. Sci., 178(3):647–662, 2008.

[PW05] Steffen Pauws and S. Van De Wijdeven.
User evaluation of a new interactive playlist generation concept.
In *In Proceedings of the ISMIR International Conference on Music Information Retrieval*, pages 638–643, 2005.

[RB09] Alan Ritter and Sumit Basu.
Learning to generalize for complex selection tasks.
In *IUI '09: Proceedings of the 13th international conference on intelligent user interfaces*, pages 167–176, New York, NY, USA, 2009. ACM.

[RBH05] R. Ragno, C. J. C. Burges, and C. Herley.
Inferring similarity between music objects with application to playlist generation.
In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 73–80, New York, NY, USA, 2005. ACM.

[RM98] D.R. Roberts and A.D. Marshall.
Viewpoint selection for complete surface coverage of three dimensional objects.

In *In Proc.of the Britsh Machine Vision Conference*, pages 740–750, 1998.

[Rod99] Kerry Rodden.
How do people organise their photographs?
In *Proceedings of the BCS IRSG Colloquium*. Electronic Workshops in Computing, 1999.

[Sha48] Claude E. Shannon.
A mathematical theory of communication.
Bell System Technical Journal, 27:379–423, 1948.

[SJ89] BW Silverman and MC Jones.
E. Fix and JL Hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on Fix and Hodges (1951).
International Statistical Review/Revue Internationale de Statistique, 57(3):233–238, 1989.

[SK00] Ben Shneiderman and Hyunmo Kang.
Direct annotation: A drag-and-drop strategy for labeling photos.
In *IV 'oo: Proceedings of the International Conference on Information Visualisation*, pages 88–96, 2000.

[SKL+08a] Lalatendu Satpathy, Saara Kamppari, Bridget Lewis, Ajay Prasad, Yong W. Rhee, Benjamin Elgart, and Steven Drucker.
Pixaura: Supporting tentative decision making when selecting and sharing digital photos.
Proceedings of HCI 2008, September 2008.

[SKL+08b] Lalatendu Satpathy, Saara Kamppari, Bridget Lewis, Ajay Prasad, Yong Woo Rhee, Benjamin Elgart, and Steven Drucker.
Pixaura: supporting tentative decision making when selecting and sharing digital photos.

In *BCS-HCI '08: Proceedings of the 22nd British CHI Group Annual Conference on HCI 2008*, pages 87–91, Swinton, UK, UK, 2008. British Computer Society.

[SLF+] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen.
Perceptual models of viewpoint preference.
In submission to ACM Transaction on Graphics, 201–.

[SM09] Robin Sease and David W. McDonald.
Musical fingerprints: collaboration around home media collections.
In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 331–340, New York, NY, USA, 2009. ACM.

[SMKF04] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser.
The Princeton shape benchmark.
In *Shape Modeling International*, June 2004.

[SP05] Dmitry Sokolov and Dimitri Plemenos.
Viewpoint quality and scene understanding.
In *VAST, Eurographics Symposium Proceedings*, pages 67–73, 2005.

[SS02] S Stoev and W Straßer.
A case study on automatic camera placement and motion for visualizing historical data.
Proceedings of the conference on Visualization '02, Jan 2002.

[SSBS07] Waqar Saleem, Wenhao Song, Alexander Belyaev, and Hans-Peter Seidel.
On computing best fly.
In *Proceedings of the 23rd Spring Conference on Computer Graphics*, pages 143–149. Comenius University, 2007.

[SWLD10] Adrian Secord, Holger Winnemöller, Wilmot Li, and Mira Dontcheva.

Creating collections with automatic suggestions and example-based refinement.
In *UIST '10: Proceedings of the 23rd annual ACM symposium on User interface software and technology*, New York, NY, USA, 2010. ACM.

[Tun09] Daniel Tunkelang.
Faceted Search.
Morgan and Claypool Publishers, 2009.

[VBP+09] Thales Vieira, Alex Bordignon, Adelailson Peixoto, Geovan Tavares, Hlio Lopes, Luiz Velho, and Thomas Lewiner.
Learning good views through intelligent galleries.
Eurographics 2009 (Computer Graphics Forum), 28(2):717–726, 2009.

[VFSH01] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich.
Viewpoint selection using viewpoint entropy.
In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 273–280, 2001.

[VGD+05] Amy Voida, Rebecca E. Grinter, Nicolas Ducheneaut, W. Keith Edwards, and Mark W. Newman.
Listening in: Practices surrounding iTunes music sharing.
In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 191–200, New York, NY, USA, 2005. ACM.

[Vie00] Fernanda B. Viegas.
Collections: Adapting the display of personal objects for different audiences.
In *Master of Science Thesis*. Massachusets Institute of Technology, 2000.

[VS02] P Vázquez and M Sbert.

Automatic keyframe selection for high-quality image-based walk-through animation using viewpoint entropy.
International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '02), Jan 2002.

[WDS+01] Liu Wenyin, Susan Dumais, Yanfeng Sun, Hongjiang Zhang, Mary Czerwinski, and Brent Field.
Semi-automatic image annotation.
In *In INTERACT2001, 8th IFIP TC.13 Conference on Human-Computer Interaction*, pages 326–333, 2001.

[WW97] Daphna Weinshall and Michael Werman.
On view likelihood and stability.
IEEE Trans. Pattern Anal. Mach. Intell., 19(2):97–108, 1997.

[YSLH03] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst.
Faceted metadata for image search and browsing.
In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 2003. ACM.

[YSY+06] Hitoshi Yamauchi, Waqar Saleem, Shin Yoshizawa, Zachi Karni, Alexander Belyaev, and Hans-Peter Seidel.
Towards stable and salient multi-view representation of 3D shapes.
In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, pages 265–270, 2006.

[Zha98] Shumin Zhai.
User performance in relation to 3D input device design.
SIGGRAPH Computer Graphics, 32(4):50–54, 1998.

[Zus70] L. Zusne.
Visual perception of form.
Academic Press, 1970.