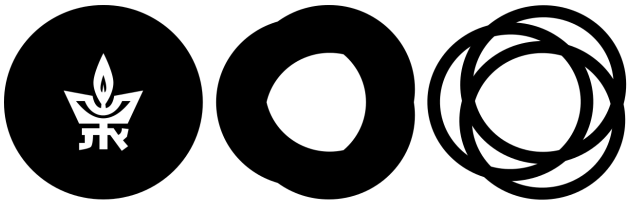


TEL AVIV UNIVERSITY

The Raymond and Beverly Sackler Faculty of Exact Sciences
The Blavatnik School of Computer Science

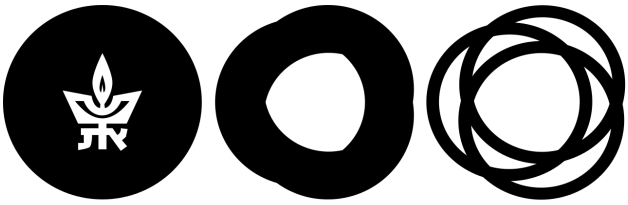
Semantic Similarity and Correspondence of 3D Shapes and Images

A thesis submitted toward a degree of
Doctor of Philosophy

by

Yanir Kleiman

August 2016



TEL AVIV UNIVERSITY

The Raymond and Beverly Sackler Faculty of Exact Sciences
The Blavatnik School of Computer Science

Semantic Similarity and Correspondence of 3D Shapes and Images

A thesis submitted toward a degree of
Doctor of Philosophy

by

Yanir Kleiman

This research was carried out in

The School of Computer Science

under the supervision of

Prof. Daniel Cohen-Or

Submitted to the Senate of Tel Aviv University

August 2016

Abstract

In this work we investigate the concept of semantic similarity between shapes and images. Similarities between elements can be used for numerous applications, such as clustering, categorization, visualization, retrieval, and browsing. For shape and image browsing, we present a novel exploration method which relies on the similarities between elements to produce a seemingly endless grid, which can be navigated in any direction like a regular map. This provides a smooth and intuitive browsing experience. Our method is efficient and highly scalable, and can be used for datasets that contain millions of elements.

For all of these applications, the quality of the solution depends strongly on the quality of the similarity measure. Similarities between elements must reflect the perceived similarity by humans, or the *semantic* similarity. This motivates further research to improve existing similarity measures, which often lack semantic context. First, we show how crowd sourced data can be used to deduce complex semantic similarities between shapes or images which are not possible to compute automatically without external knowledge and context. Such methods can be used to complement automatic methods and provide additional external context.

Next, we focus on the shapes domain. Shape segmentation, correspondence between shapes and semantic similarity are some of the pillars of shape analysis, and each of these problems enjoys extensive research of its own. However, these problems are linked together, and the output of one can be the input of another. Similarity of parts can be used to discover shape segmentation, segmentation can be used to compute shape correspondence, and correspondence can be used to compute semantic similarity between shapes. We present a similarity measure between shapes which is computed by segmenting and finding a correspondence between the segments. This similarity measure captures semantic relations such as shapes that belong to the same style, or have a similar function.

We further investigate the link between correspondence and segmentation of shapes, showing how segmentation of the shape can improve point-to-point correspondence. When matching shapes which contain symmetries, there are multiple contradicting solutions to the correspondence problem, which often cause instability of the final solution. To alleviate this problem we propose a symmetry aware correspondence, in which each segment can match several segments in the other shape. This enables us to find a matching between

symmetric segments in a very efficient manner. The symmetric matching between segments is less detailed but more accurate than non-symmetric methods, and it can be used to improve point-to-point correspondence even without providing a one-to-one mapping between the segments. Furthermore, resolving the symmetry can potentially be done as a post-process which is decoupled from the matching and thus less complicated.

Acknowledgments

I would like to thank my advisor, Daniel Cohen-Or. There are no words to summarize five years of working together. Danny always treated me as equal from the very beginning, pushing me to bring forward my own ideas rather than following the herd. He had enough patience for me to follow through on ideas that worked as well as some that didn't. He taught me the importance of collaboration and introduced me to many great people, some of which are mentioned below.

I thank all of my collaborators: Oana Sidi, Oliver van Kaick, Hao (Richard) Zhang, Noa Fish, Joel Lanir, Shmuel Asafi, Dov Danon, Yasmin Felberbaum, Olga Sorkine-Hornung, George Goldberg, Yael Amsterdamer, and Rachele Bellini. Special thanks to Oliver van Kaick, whose help was instrumental for some of the works detailed here.

A few people were gracious enough to host me in their labs, and show me the most fun part of doing research - traveling the world. My thanks to Olga Sorkine-Hornung, Tobias Ritschel, and Baoquan Chen. I would like to specifically thank Maks Ovsjanikov for hosting me in his lab during the last year of my PhD, and for his help and patience. Chapter 5 of this dissertation could not have been written without him.

Tel Aviv University will always feel like home to me; I spent about 9 years of my life between these walls. I would like to thank my labmates Noa Fish and Hadar Averbuch-Elor, for friendly advices, collaborations, and loads of fun, including McDonalds every Monday and a farewell t-shirt that I will never wear in public. I also thank Ronit Reitshtein for helping me with all the paperwork I needed help with.

My family supported me in every way throughout my studies, letting me skip holiday meals and birthdays for deadlines, helping with the kids when I was away, and even running errands in the university when I was abroad. I am grateful for everything.

Finally, I would like to thank Hedva, who held my head above water when everything seemed impossible. This work is as much yours as is it mine.

Table of Contents

List of Figures

List of Tables

1 Introduction

3D object repositories have gone from being small and scarce to large and abundant. Image repositories have seen similar growth but in a different scale; they have gone from being large to massive, and from abundant to omnipresent. The abundance of large collections increases the demand for efficient, reliable and intuitive ways to organize and explore shapes and images.

The main focus of this dissertation is the study of *semantic similarity* within large collections of shapes or images. We are interested in measuring semantic similarities between objects, as well as the applications of such a similarity measure. Detecting similarities between shapes or images is a core component of a broad range of tasks, such as retrieval, exploration, categorization, and classification. For shape and image exploration, we propose *Dynamic Maps*, a similarity based browsing method which enables intuitive navigation through large collections without relying on keyword search or filtering. This application, as well as the many tasks mentioned above, can greatly benefit from a similarity measure which reflects semantic information regarding the shape or image: the function of an object, its origin, its location, and more. However, many of the state-of-the-art similarity measures are feature driven, and do not reflect semantic information well.

Motivated by the necessity of similarity measures that incorporate semantic information, we present two methods to measure semantic similarities. The first relies on crowdsourced data, and suggests a querying scheme which gains a lot of relevant information from each query, thus reducing the necessary number of crowd queries. Using crowd queries enables us to capture semantic information that might exist outside of the image or 3D object, such as identifying paintings by the same painter or objects that belong in the same setting. The second method measures semantic similarity between shapes by comparing the parts that each shape is composed of. Using the correspondence between parts, we

can estimate which transformations are necessary to transform one shape into the other. We define the *shape edit distance* as the sum of these transformations. The shape edit distance is sensitive to the function of a shape and its origin.

The derivation of the shape edit distance demonstrates the relation between segmentation, correspondence, and similarity of 3D shapes. These three problems are some of the pillars of shape analysis, and each of them enjoys extensive research of its own. However, these problems are linked together, and the output of one can be the input of another. Similarity of parts can be used to discover shape segmentation, segmentation can be used to compute shape correspondence, and correspondence can be used to compute semantic similarity between shapes. We further investigate the link between correspondence and segmentation of shapes, and show how the segmentation of shapes can help the computation of a point-to-point correspondence between them. We introduce *symmetry aware correspondence*, in which each segment can match several segments in the other shape. We present an efficient and stable method of computing such correspondences, which alleviates the inherent instability of the optimization when symmetric shapes are being matched. We show that the symmetric matching of segments can be used to improve point-to-point correspondence even without providing a one-to-one mapping between the segments.

Finding correspondences between shape parts involves solving a graph matching problem. Graph matching is a heavily researched area, however most existing work focuses on one-to-one or one-to-many matching. In the methods above, we require a more complex structure of correspondence. Therefore, these works include notable contributions for the graph matching problem. We introduce two adaptations of spectral correspondence, a prevalent graph matching method. These adaptations are detailed in Sections 4.3 and 5.3.

In the rest of this chapter, we elaborate the discussion of semantic similarity and its applications, and of the contributions mentioned above.

1.1 What is Semantic Similarity?

We say that two images or shapes are *semantically* similar when the similarity between them is based on high-level concepts rather than low-level representations or features. Below we give examples of similarity measures with low semantic level and high semantic level. We categorize similarity measures to

three groups: non-semantic, basic semantic and highly semantic. However, this is not a mathematical concept which can be defined in exact terms, and other categorizations are possible. We claim that highly semantic similarity measures are more accurate in practice and more useful than similarity measures with low semantic level, as demonstrated in the following chapters of this dissertation.

Non-semantic similarity. Non-semantic similarity measures depend heavily on the low-level representation of the object. The similarity between images is measured by directly comparing their corresponding pixel values, and the similarity between shapes is measured by directly comparing their vertex positions. Thus, such measures can only capture similarities between images or shapes which are almost identical. In Figure 1.1 we show similar images for which the values of each pixel are quite different. Such images would not be considered similar by a non-semantic similarity measure.

Figure 1.1: *Similar images with low pixel-level similarity*

Basic semantic similarity. The basic semantic information in an image or 3D shape are what type of objects they depict: a flower, a bee, a human face, a mountain or a sunset on the beach. For images, image descriptors provide a higher level representation of the image, and can sometimes capture such semantic meaning. Simple descriptors can identify similarities in composition and color palette, while advanced descriptors can also estimate the context of a scene, or the type of content in the image. In Figure 1.2 we show images that have semantic similarities that can typically be captured by image descriptors.

3D shapes are less prevalent and less studied than images. Still, state-of-the-art techniques are mostly capable of identifying basic semantic information such as overall shape (comparable to the composition of an image) and the category the object belongs to.

Figure 1.2: *Images with basic semantic similarity*

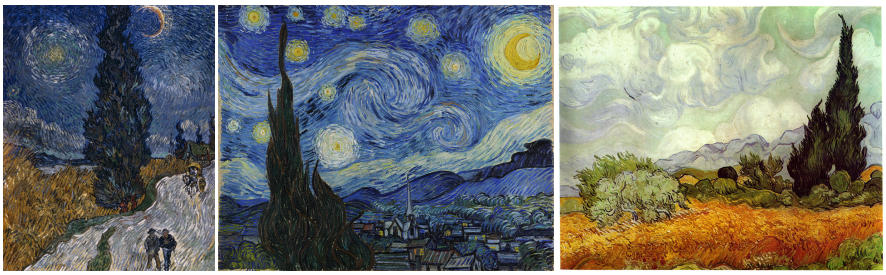
Specialized methods can also recognize semantic similarities in specific contexts. For example, identifying the similarity of different photos of the same person, photos of the same city, or the same 3D model in different poses.

Highly semantic similarity. Highly semantic similarity is derived from a broad semantic context, which may include elusive relations such as a similar emotion or sensation evoked by images (e.g., images that convey "fear" or "comfort"); shapes which are semantically related (e.g., different types of musical instruments); likeness between the photographed people; paintings by the same artist; or shapes that are modifications of the same base shape. These relations are quite often external to the image or shape itself; they come from common knowledge or experience, and thus cannot be deduced directly from computed features. In Figure 1.3, we show images that have external high level semantic relation to each other, as they were painted by the same artist.

Note that semantic similarity also depends on the context of the image, and on personal taste. Within a given context, for example in Figure 1.3, some people may decide that images (a) and (b) are more similar, while others may decide that images (b) and (c) are more similar. These decisions may change within a different context, such as the one given in Figure 1.4.

1.2 Applications of Semantic Similarity

In this section, we briefly describe some common applications of similarity between shapes or images, which motivate our search of highly semantic similarity measures.



(a) (b) (c)

Figure 1.3: *Images with high level semantic similarity*

Figure 1.4: *Images within context*

1.2.1 Image and shape retrieval

The most prominent applications of similarity are *image retrieval* and *shape retrieval*. The input of this task is a single image (shape), and a large collection of images (shapes). The goal is to retrieve the items in the collection which are most similar to the input. In Figure 1.5 we show examples of image and shape retrieval. The simplest way to perform retrieval is using a k nearest neighbors approach, and output the k elements in the collection which have the highest similarity to the input, according to the similarity measure. A non-semantic similarity measure can retrieve elements with low-level similarity to the input, if such elements exist in the collection. However, most often such elements do not exist. A semantic similarity measure can retrieve elements with high-level semantic similarity to the input as well, leading to more relevant query result, especially when k is large.

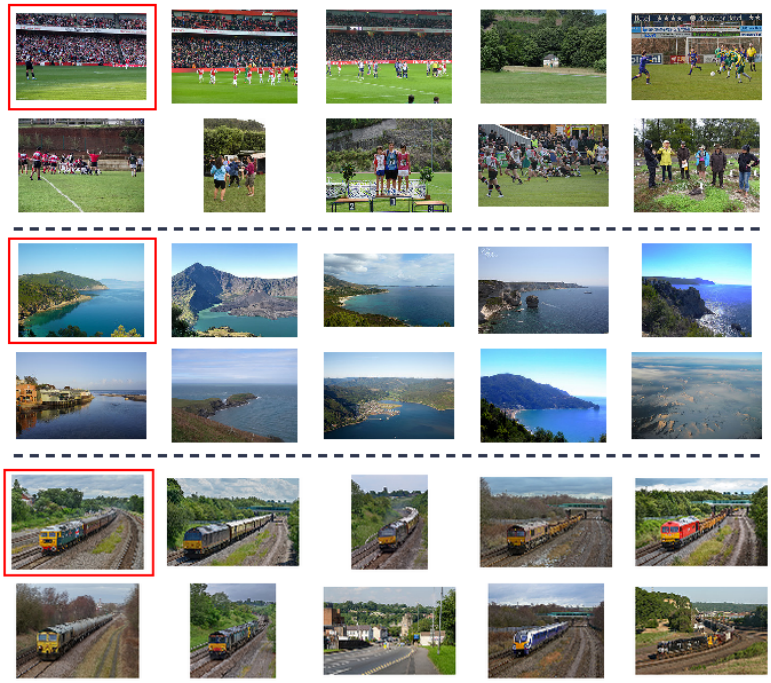


Figure 1.5: *Image retrieval using image descriptors. Each query image (marked in red) is displayed with its 9 closest matches, from left to right and top to bottom. The images are taken from a dataset of one million images.*

1.2.2 Relevance feedback

Since in many cases the distinction between a relevant and irrelevant result is user subjective, relevance feedback techniques were incorporated into many retrieval systems, allowing the user to guide the search according to personal preference and taste [Rui *et al.*, 1998; Leifman *et al.*, 2005; Cao *et al.*, 2006; Akgül *et al.*, 2010; Suditu and Fleuret, 2011]. Relevance feedback involves presenting the user with a set of suggested items. The user marks the preferred or relevant items, and presented with items which are similar to the selected ones. The process may then be repeated several times until the user is satisfied, often employing machine learning techniques in order to aggregate and refine previous selections. Still, the basis for learning is the core similarity measure between items in the collection, and a semantic similarity measure is more desired than a non-semantic one.

1.2.3 Embedding

Another important application that benefits from a reliable distance measure is mapping a set of shapes or images onto a low dimensional manifold. This mapping can be used for visualization or as the basis of many machine learning algorithms which involve dimensionality reduction. One of the most prevalent methods for dimensionality reduction is *multidimensional scaling*, or MDS [Sammon, 1969; Kruskal and Wish, 1978]. MDS takes as input the distances between elements and seeks an embedding in which the difference between the input distances and actual distances are preserved. The only input for the process are the distances between elements. Thus, for a useful embedding it is crucial to have an accurate similarity measure. An example of MDS embedding of a set of shapes is given in Figure 1.6.

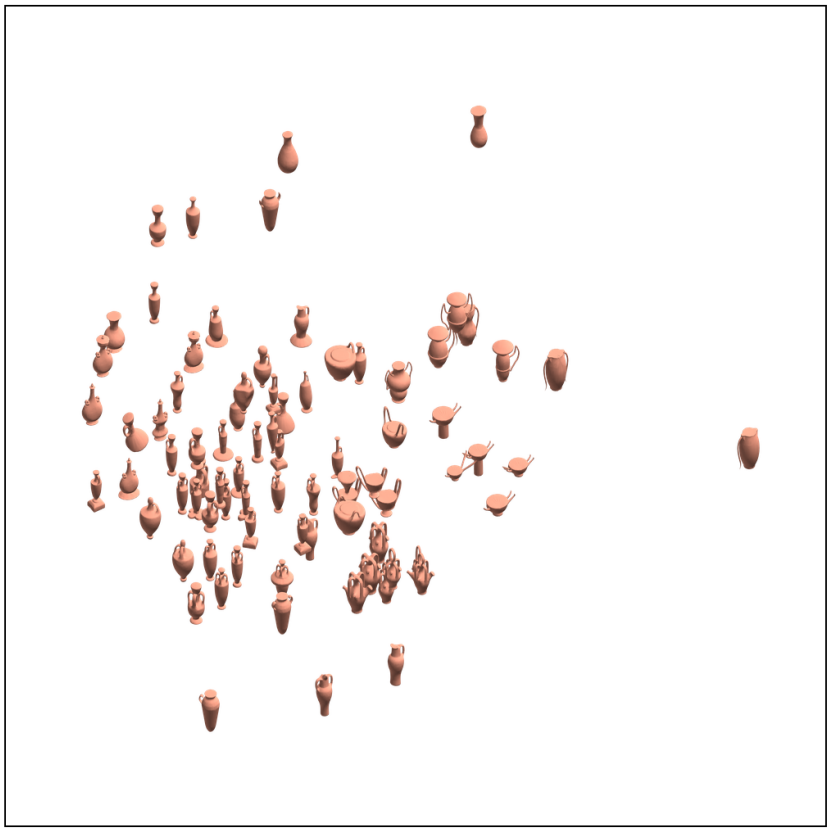


Figure 1.6: *MDS embedding obtained from SHED for a set of vases. Note how similar shapes are grouped together.*

1.2.4 Clustering

Clustering is a useful way to organize data, either for visualization or for further processing which relies on the common properties of the elements in each category. Most clustering methods rely on some similarity measure between elements in the collection. A well known method is *k-means* clustering, in which every element is classified to the closest of k centers, or the center with the highest similarity to the element. To measure this distance, a reliable similarity measure is required. Other advanced methods (for example, spectral clustering [Shi and Malik, 2000] or diffusion maps [Coifman and Lafon, 2006]) perform some manipulation or scaling of the original similarity space, but still can only be effective if the similarity measure is adequate. Figure 1.7 shows an example of clustering based on semantic similarity.

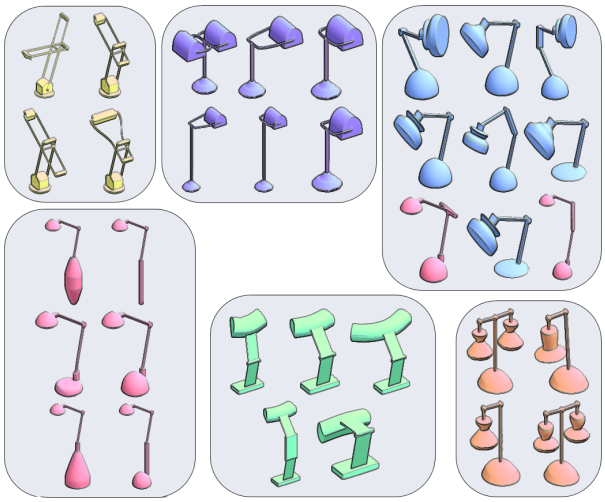


Figure 1.7: *Clustering of a set of lamps according to SHED.*

1.2.5 Categorization trees

By using hierarchical clustering, we can create categorization trees for a set of shapes or images. The resulting trees hierarchically organize the elements of the set and can be used for shape or image exploration. The trees can be generated automatically by using a non-parametric clustering method, in which the number of clusters in each set is selected automatically, such as Self-

Tuning Spectral Clustering [Zelnik-Manor and Perona, 2004]. An example of a categorization tree of shapes is presented in Figure 1.8.

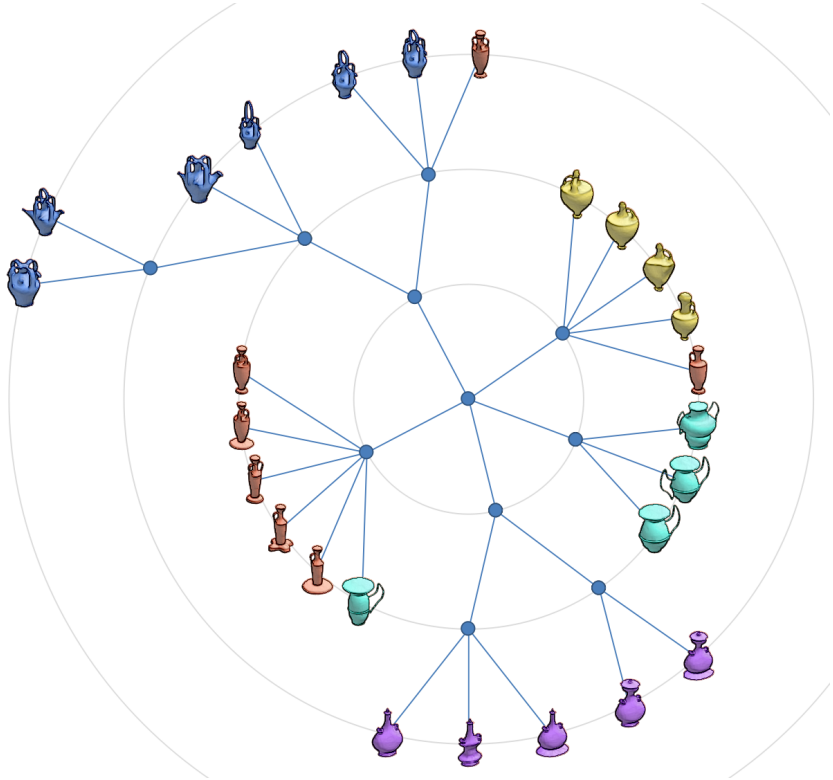


Figure 1.8: *Categorization tree of a set of vases according to SHED distances.*

1.3 Similarity Based Browsing with Dynamic Maps

For image and shape exploration, we introduce an intuitive browsing method which relies on the similarities between elements. Thumbnails are laid out on a grid which can be navigated like a geographic map, using pan and zoom operations. The grid is continuous and dynamic, with each patch generating at the moment it is needed. Images and shapes are high dimensional objects and the similarities between them cannot be accurately captured in a two dimensional space. Thus, any global mapping of images or shapes onto a two dimensional grid is bound to have discontinuities. However, the dynamic generation of the grid, which does not produce a global mapping, ensures every patch is locally smooth while the map remains coherent. This provides a smooth browsing experience for the user. An additional advantage of dynamic maps

is their ability to adjust to the direction of browsing, thus better reflecting the user's interest. Dynamic maps are efficient to generate and are very scalable. Thus, they can provide an on-line experience and support on-line changes to an underlying dataset of millions of elements. For a detailed discussion and evaluation of dynamic maps, see Chapter 2.

The work in this chapter was published in the following two papers:

- [Kleiman *et al.*, 2013] **Dynamic maps for exploring and browsing shapes.** Yanir Kleiman, Noa Fish, Joel Lanir, and Daniel Cohen-Or. *Computer Graphics Forum (SGP), 2013.*

- [Kleiman *et al.*, 2015a] **DynamicMaps: Similarity-based browsing through a massive set of images.** Yanir Kleiman, Joel Lanir, Dov Danon, Yasmin Felberbaum, and Daniel Cohen-Or. In *Proceedings of the SIGCHI conference on Human factors in computing systems, 2015.*

1.4 Semantic Similarity from Crowdsourced Clustering

In this work, we aim to compute a similarity measure which is completely semantic, driven by the context of a shape or image, as well as external information and even the emotion it evokes. Such context cannot be discovered by automated tools. Instead, we propose to gather information from a crowd using a crowdsourcing technique. Our goal is to quickly converge to an accurate similarity measure, while minimizing the cost, which depends on the number of necessary queries and their complexity. There are two main challenges in developing such a crowdsourcing technique. First, how to compare images in an efficient and useful way, i.e. what type of queries should be asked. Second, which images to include in each query, and how to use previous queries to better construct future queries.

We answer these two questions and present a method based on clustering queries. Users are given the task to cluster n images into k bins. We then use the clustering results to gradually improve the similarity measure by embedding the objects in a low-dimensional space. This embedding resolves conflicts in the query results and consolidates all of the provided information into a single coherent space. Then, new queries are created based on the embedding and the process is repeated. The resulting similarity measure can complement descriptor

based methods, for example to compute semantic similarities for a fraction of the dataset and propagate them to visually similar images. The method and its evaluation are discussed in details in Chapter 3.

The work in this chapter was published in the following paper:

- [Kleiman *et al.*, 2016] **Toward semantic image similarity from crowd-sourced clustering.** Yanir Kleiman, George Goldberg, Yael Amsterdamer, and Daniel Cohen-Or. *The Visual Computer, 2016*.

1.5 Semantic Shape Similarity Using Shape Edit Distance

In the shapes domain, current similarity measures mainly focus on distinguishing shapes from different classes, and give little attention to intra-class similarity. State of the art techniques are based on the appearance of the shape as a whole and capture low level similarities between shapes. These methods do not capture well similarities between articulated shapes, partial shapes, or shapes with changes of scaling of the parts; they lack the necessary semantic level to identify shapes of similar function or style.

We develop *shape edit distance*, a similarity measure which captures the fine details of the shapes, as well as the overall structure of the shapes. We aim to measure the amount of effort necessary to transform one shape into the other. To this end, we segment each shape into approximately convex parts [van Kaick *et al.*, 2014], and find a matching between the shape parts. Using the matching we estimate the magnitude of transformation each part went through. Thus, our method is capable of identifying shapes of similar function or style. The shape edit distance provides an intuitive similarity measure which is relatively close to human perception of similarity between objects. It is useful for identifying similar objects within the same class, as well as distinguishing between different classes of shapes.

A core part of our approach is computing correspondences between shape parts. Correspondence between graphs or feature points has been heavily researched in recent years, but mostly in the context of one-to-one correspondences [Leordeanu and Hebert, 2005; Berg *et al.*, 2005; Kezurer *et al.*, 2015] or one-to-many correspondences [Cour *et al.*, 2006; Leordeanu *et al.*, 2009]. These methods are less effective when a different correspondence structure is required.

For shape edit distance, we require a bidirectional one-to-many matching: a part in one shape can match many parts in the other shape and vice versa, but many-to-many relations are not allowed. Within these relatively flexible constraints, there are often competing solutions with high likelihood. In such cases some likely matches might belong to one solution and other likely matches belong to another. Therefore, we introduce an adaptive spectral correspondence technique, based on the popular spectral relaxation model [Leordeanu and Hebert, 2005]. Our technique iteratively improves the optimization based on previous selection of matches. The adaptive technique gives precedence to matches that belong to the same solution, which are more compatible with one another. The shape edit distance and its evaluation are discussed in details in Chapter 4. The details of adaptive spectral correspondence are discussed in Section 4.3.

The work in this chapter was published in the following paper:

- [Kleiman *et al.*, 2015b] **Shed: shape edit distance for fine-grained shape similarity.** Yanir Kleiman, Oliver van Kaick, Olga Sorkine-Hornung, and Daniel Cohen-Or. *ACM Transactions on Graphics (SIGGRAPH Asia), 2015.*

Approximately convex segmentation, a collaboration with the author which is used for shape edit distance but not detailed in this dissertation, was published in the following paper:

- [van Kaick *et al.*, 2014] **Shape segmentation by approximate convexity analysis.** Oliver van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or. *ACM Transactions on Graphics (TOG), 2014.*

1.6 Symmetry Aware Correspondence Using Shape Graphs

In this work, we follow a similar idea of segmenting shapes and finding a correspondence between shape parts, and use it to improve point-to-point correspondences and symmetry detection. We create a shape graph from the segmentation of a shape, where each node corresponds to a segment. This shape graph provides information about the structure of the shape, which can be used to stabilize the correspondence and rule out correspondences which are not coherent with the rest of the shape. In particular, we use the shape graphs to find a *symmetry aware correspondence* between the shape parts. Intuitively, the

symmetry aware correspondence allows segments to be matched to any number of symmetric segments. In practice, groups of segments of various size can correspond to groups of segments in the second shape, such that every segment in the first group corresponds to every segment in the second group.

Matching shapes that contain intrinsic symmetry is a particularly difficult correspondence problem, since there are multiple solutions which are equally likely to be correct. This non-convexity makes the optimization expensive and error prone, and often leads to inaccuracies in the final solution. Symmetry aware correspondences relax the one-to-one constraints and effectively reduce the search space. The new search space has a single stable solution which can be found very quickly and efficiently.

Note that for many-to-many correspondences, a solution that matches every part in one shape to all parts of the other shape is valid yet not desired. This is unlike one-to-one or one-to-many correspondences which do not allow benign solutions. This poses a problem when the optimization solution is not sparse, since there is no clear indication of how many matches to include in the final correspondence. Thus, we present an optimization method based on spectral correspondence which produces sparse solutions, where only matches that belong to the correspondence are associated with high values.

Converting a symmetry aware correspondence to a one-to-one correspondence requires additional steps; however, we show that the symmetry aware correspondence is useful even without producing a one-to-one correspondence, for example to improve point-to-point correspondence solutions. Our method has an additional application of symmetry detection, by matching a shape to itself. While one-to-one correspondences provide more information, our solution is significantly faster and provides more accurate correspondences than such methods. Our symmetry aware correspondence method is described in details in Chapter 5. The results of this work are pending publication.

2 Similarity Based Browsing with Dynamic Maps

In this chapter we present an intuitive tool for *similarity based browsing*, an open-ended exploration of shapes or images which relies on the similarities among them. Shape browsing and image browsing have different end results, however in practice they are performed in a similar manner since 3D models are typically viewed as a single rendered image during the browsing. Therefore, throughout this chapter we will mostly focus our discussion on image browsing, except the discussion of implementation details and evaluation.

2.1 Image Browsing

Commercial tools for image browsing or image search focus mostly on keyword search. The images are presented in a grid ordered by a relevance measure relative to the input keywords (similar tools exist for shape browsing). While text-based directed search can be effective for finding specific types of images, studies have shown that image search is often more exploratory in nature than Web search, and that browsing is an essential strategy when looking for images [André *et al.*, 2009; Markkula and Sormunen, 2000; Chew *et al.*, 2010]. Still, most commercial systems lack support for exploratory search and do not provide means for serendipity in the search process [Hearst, 2009; Markkula and Sormunen, 2000].

To address this gap, various research systems have looked into browsing as a complementary tool to text-based search methods [Combs and Bederson, 1999; Pečenović *et al.*, 2000]. One useful way of browsing through images is by using similarity. Users often look for images that are similar to a given image,

and browsing according to similarity between images has been shown to be useful [Rodden *et al.*, 1999; Liu *et al.*, 2004]. Relevance feedback (see Section 1.2.2) is often used to refine search results using a selection of preferred images [Zhou and Huang, 2003]. At each relevance feedback step, the user is presented with a new set of images based upon past selections. However, the navigation experience with this approach is not continuous and it requires the user to go over a large collection of images and select the relevant or irrelevant ones at each step.

A possibly more intuitive approach is to lay out the images on a continuous navigable two-dimensional grid such that similar images are displayed closely together. Navigating such a grid is similar to navigating a geographic map application, thus it has an intuitive and familiar user experience. Zooming capabilities can also be added to the navigation options to enhance the user experience. The continuous navigation eliminates the need for explicit relevance feedback on individual queries. Instead, the direction of navigation provides hints to the type of results to be displayed next. The challenge, however, is the generation of the continuous grid.

2.2 Dynamic Maps

Images and shapes are extremely high dimensional elements. Generating a cohesive global two-dimensional manifold that preserves similarity relations among all images or shapes is therefore challenging, if at all possible. However, when a user interactively navigates a map-like interface, only a small portion of the search space is displayed at a time. Our key idea is that for such navigation, global requirements can be relaxed. Navigation is done over a pseudo-map, where the data is dynamically organized into a local manifold, only in the region currently observed by the user. There are several advantages to generating a dynamic map on the fly. First, global constraints are relaxed and a locally continuous map can be generated, in which a pair of shapes are near in the embedding only if they are relatively close in the original high dimensional space. Note that the opposite is not necessarily true; a pair of shapes that are similar, i.e. close in the original high dimensional space, are not guaranteed to appear in the same patch. Their placement in the patch depends on the direction of browsing selected by the user. Second, the generated map can interactively change according to the user's interest and direction of browsing, thus providing

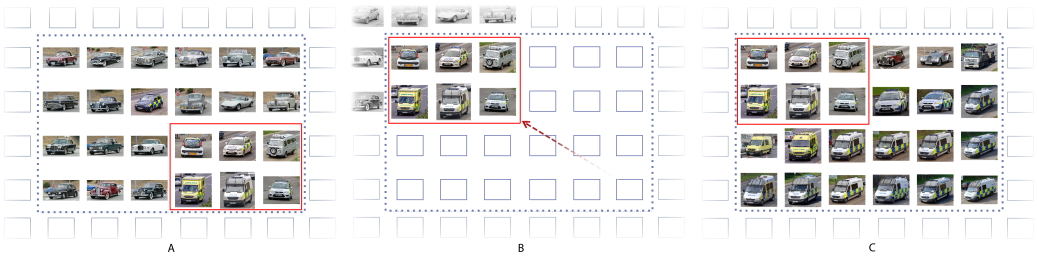


Figure 2.1: *Browsing images using a dynamic map. The map displays a region of images ordered by similarity (A). Dragging the map to the up left corner (B) reveals new images which are similar to shapes in the dragging direction (C).*

an effective browsing experience without intrusively querying the user. Third, a local region of the map can be generated quickly, and there is no need for a long computation time to generate the entire map. Thus, dynamic maps can provide an on-line experience and support on-line changes to the underlying dataset.

Figure 2.1 illustrates the navigation process in our solution. The user views a local subset of images, ordered such that similar images are next to each other. In this particular example, the user views a region of cars, and decides to navigate towards police cars. Another patch of the map is revealed, and instantly filled with similar images to the images framed by the red rectangle. The currently displayed map can be figuratively viewed as a window that shows a local patch of the pseudo-map. Figure 2.2 shows screenshots of our system during typical browsing sessions.

The challenge in generating such pseudo-maps is to create local manifolds that keep the sense of continuity. That is, the user pans over the pseudo-map while the manifold is perceived to be continuous. We present a technique of

Figure 2.2: *Screenshots of typical browsing sessions.*

embedding images onto dynamic pseudo-manifolds, where the relative positions of images respect only local high-dimensional relations. Relative distances among the displayed images are not necessarily preserved, allowing for an efficient usage of the display space and a spatially dense representation of the images domain. In contrast with common dimensionality reduction techniques (e.g. MDS), the end result of our method is not a global map which contains all images at once. The generated local pseudo-maps only exist temporarily within the viewport of the user; when the user navigates to reveal a new region of the map, only local relations to the previous map are maintained. Navigation over the pseudo-map enables a free-form exploration, where users can quickly and seamlessly direct the search towards relevant models of their choice.

The generation of local neighborhoods in the dynamic map is based on the assumption that for high dimensional data such as 3D models, short distances are more accurately measured than long distances. We thus use only the shortest distances between images in our dataset; only the distances to k nearest neighbors (with k being a small positive integer) of each image in the dataset are considered. A dense set is expected to have shorter distances, and thus more accurate, than a sparse set, hence our method is especially suitable for massive datasets.

2.3 Related Work

2.3.1 Image browsing

Images have several characteristics that makes image search different than text-based search. Unlike text documents, the content of an image can be grasped at a glance, and a large number of images can be presented to a user at once. In image search, often the user does not have an exact target in mind [Chung and Yoon, 2011]. Furthermore, images often lack textual cues and might have many different meanings embedded in a single image [Layne, 1994], making them difficult to support with only keyword-based search. For example, if the user is looking for a scenery image to add to a presentation, the user would not necessarily know how to phrase the search terms or even exactly which image he or she is looking for. Moreover, images presented in the first page of a text-based search result are not necessarily better than those presented in the following pages. Consequently, users have to sequentially scan these

results spending considerable effort finding relevant images. Still, most current systems focus on providing text-based image querying rather than navigational support even though studies have shown that image browsing can improve in achieving user's search needs [Combs and Bederson, 1999; Liu *et al.*, 2004; Pečenović *et al.*, 2000].

To address these needs, some research systems focus on supporting various browsing capabilities to enable navigating through images. For example, browsing specific clusters of images [Pečenović *et al.*, 2000], browsing hierarchies that are automatically built according to visual and semantic similarities [Jing *et al.*, 2012], or browsing along conceptual dimensions according to hierarchical faceted metadata [Yee *et al.*, 2003]. Similarly, some commercial systems added interactive visual contentbased search methods that allow browsing by similar shape and/or color. The "similar images" feature, allows users to search for images similar to a certain image, utilizing relevance feedback methods.

Laying out images on a large canvas allows users to browse the images according to some organization of their structure using pan and zoom interactions. In [Combs and Bederson, 1999], the results of an initial query can be browsed on a zoomable user interface (ZUI). In [Pečenović *et al.*, 2000], images were clustered into conceptual regions. The user can continuously pan across this plane and zoom in or out of any particular region. In JustClick [Fan *et al.*, 2009], a topic network is first generated and browsed through. Representative images of a topic are then organized on a 2D hyperbolic plane according to similarity.

In the works above, the images are laid out according to some measure of distance (in similarity) between them. However, when browsing images, there is no need for an accurate representation of the original distances between images. In fact, an even spread of images over the canvas can be more beneficial than an accurate representation of the original geometry [Rodden *et al.*, 1999], especially in cases where the original data includes very distinctive clusters which may appear too far apart for easy navigation. Indeed, the most common way to lay out a set of images is on a twodimensional grid. Studies have found that arranging a set of thumbnail images on a single-page grid according to their similarity can be useful for users in an image browsing task [Liu *et al.*, 2004]. Strong and Gong [Strong and Gong, 2008; Strong *et al.*, 2010] employed this idea and organized a collection of images based on similarity using an SOM-based algorithm. Users could browse the image collection using pan and zoom interactions. According to the authors their system could support browsing with

up to 10,000 images. Similarly, in PhotoMesa [Bederson, 2001], images are laid on a large 2D grid. Users can browse through a large collection of images, panning to browse horizontally or vertically through the image collection. Here, zooming out enabled seeing the photos semantically grouped into preorganized categories.

The systems mentioned above work with a limited number of images and are not scalable beyond several thousands of images. Thus, they are not suited for large repositories that exist in the Web today. Our work builds upon the idea of browsing images on a large 2D canvas, and the works in [Liu *et al.*, 2004; Rodden *et al.*, 2001; Strong and Gong, 2008] that present similar images together on a grid. However, we apply it to a dataset of virtually unlimited size, finding solutions for interacting in such a large image space.

2.3.2 Shape browsing

A common means to explore large shape repositories is by searching for similar shapes through a series of queries. The problem of searching for similar shapes to a given query object is known as "shape retrieval". During the last two decades a huge body of work in that area has focused on the development of various shape descriptors and signatures to facilitate retrieval. Among them are descriptors based on statistical moments [Elad *et al.*, 2002; Novotni and Klein, 2003; Kazhdan *et al.*, 2003], distance [Osada *et al.*, 2002], symmetry [Kazhdan *et al.*, 2004], volume [Zhang *et al.*, 2001; Shapira *et al.*, 2008]. For more information see a survey by [Tangelder and Veltkamp, 2004]. An alternative approach was introduced by Bronstein et al. [Bronstein *et al.*, 2011]. Instead of global shape signatures they compute local features such as Heat Kernel Signature (HKS) [Sun *et al.*, 2009], quantize them into geometric words, and use them in a bag of words manner to discover similarities between shapes.

Shape exploration is commonly carried out by interactively navigating through design galleries based on a parametric model [Shapira *et al.*, 2009]. Design galleries have been used for model suggestions based on part correspondence [Chaudhuri and Koltun, 2010; Kim *et al.*, 2012] and semantic context [Talton *et al.*, 2009]. Vieira et al. [Vieira *et al.*, 2009] utilize design galleries for learning descriptive views of 3D objects, where the user supplies the training data by selecting good and bad object positions. Another form of exploration is presented in Yang et al. [Yang *et al.*, 2011], where a shape space is characterized from an input mesh and a set of non-linear constraints is then

used for exploration and navigation of new designs that are aligned with the given constraints. Umetani et al. [Umetani *et al.*, 2012] present a method for shape exploration (in this case - furniture) constrained by physical requirements. The user is able to focus on the aesthetic side of the design while the system enforces physical soundness. Ovsjanikov et al. [Ovsjanikov *et al.*, 2011] extract a deformation model from an input shape to explore in a constrained manner the variability within a set of similar shapes.

2.3.3 Relevance feedback

Many recent search and retrieval systems, including both image and shape retrieval, utilize relevance feedback [Rui *et al.*, 1998; Leifman *et al.*, 2005; Cao *et al.*, 2006; Akgül *et al.*, 2010], a method to refine search results using selection of preferred elements. Suditu and Fleuret [Suditu and Fleuret, 2011] presented an image retrieval system that features iterative relevance feedback for a very large set of images. At each step, the user is presented with a set of images, and selects a single image that is the closest match to the desired query. Then a new set of images is displayed and the process is repeated.

While this process may be effective at filtering relevant images, the use of relevance feedback in commercial search interfaces is still relatively rare [Ruthven and Lalmas, 2003]. One possible explanation is that it requires users to make relevance judgments on each item, which is an effortful user task [Ruthven and Lalmas, 2003; Croft *et al.*, 2001]. Relevance feedback tends to work best when the user selects multiple objects as relevant as well as some objects as irrelevant. However, selecting multiple objects is cumbersome for most users. This is amplified in image search where extractable low-level features (e.g., color, texture, shape) may not necessarily match high-level perception-based human interpretation [Zhou and Huang, 2003]. Our method is inspired by that concept, but operates on the implicit feedback given by the user's advancement through the dynamic map.

2.3.4 Planar Mapping

Generating a two dimensional map of high dimensional elements is in essence a dimensionality reduction task. Common dimensionality reduction techniques such as multidimensional scaling (MDS) or locally linear embedding (LLE)

[Roweis and Saul, 2000] create a global manifold that aims to preserve the distances among the high dimensional data points, to the extent possible. Such global solutions are beneficial for applications such as clustering and classification, which rely on the underlying geometry or spread of data. However, our premise is that for browsing tasks, there is no need for an accurate representation of the original distances between shapes. In fact, an even spread of shapes over the map area can be more beneficial than an accurate representation of the original geometry of the search space, especially in cases where the original data includes very distinctive clusters which may appear too far apart for easy navigation.

Our dynamic map bears some resemblance to the self organizing map (SOM) [Kohonen, 1990], a popular dimensionality reduction method that produces a dense and intuitive grid-like structure. The grid preserves similarity between elements without preserving the distance. However, an SOM provides a global solution, in which local discontinuities may occur frequently. In addition, it entails a computationally intensive training process, which is applied globally as a pre-process, making it difficult to use on a very large dataset with frequent updates. Our technique is local and computationally inexpensive, which makes it a viable option for massive online datasets of images which are constantly changing.

A number of papers use dimensionality reduction techniques to map and then browse an images space according to the global relations among images [Chen *et al.*, 2000; Pečenović *et al.*, 2000]. In order to better organize the images, layout methods have been applied to MDS results to put them on a 2D grid [Rodden *et al.*, 2001]. Works such as [Sakamoto *et al.*, 2004; Lasram *et al.*, 2012; Strong and Gong, 2008; Strong *et al.*, 2010] use SOM to visualize a given small set of elements in a global cohesive map.

Such methods work well for small sets, however they are too computationally intensive and globally constrained to be effective for massive datasets. Perhaps more importantly, it is not possible to capture all of the high dimensional relations in a single global low dimensional map. Therefore, global dimensionality reduction methods cannot preserve continuity and local similarity everywhere on the map. This issue is the underlying motivation of our work. Since our method relies only on local relationships, our technique is locally smooth, computationally inexpensive, and highly scalable.

2.4 Map Generation

We provide the user with a dynamic grid-like map which is instantly and continuously generated during user interaction. The input to the map generation process is a precomputed nearest neighbors graph with a similarity score for each edge. The map can be seeded around a specific image or constrained by any number of images. As the user is navigating by panning the map, the map is extended locally to the region of interest, using previously placed images as constraints. The map is generated by iteratively filling in empty cells in the grid with the most compatible image for each cell. The compatibility of a image to a cell in the grid depends on the images that are already assigned to adjacent cells in the grid; each adjacent image votes for its nearest neighbors as candidates, and the scores of all candidates are accumulated to produce a majority vote.

Every image I in the dataset is associated with a list of nearest neighbors $I' \in Near(I)$, and their respective similarity scores $S(I, I')$. Each cell c in the grid is connected to a weighted list of adjacent cells c_i with respective weights w_i. For example, in our implementation each cell is connected to neighbors on the five by five grid centered at the cell in question, with weights that are inversely correlated with the Euclidean distance between the cells. We refer to existing images that occupy the adjacent cells of cell c as *reference images* or $R(c)$ where each filled cell c_i is associated with an image I_i. The compatibility score for placing an image I in cell c is defined as the weighted sum of similarity scores for each neighbor that appears in the list of reference images:

$$C(I, c) = \sum_{I_i \in R(c)} w_i \cdot S(I, I_i)$$

where $S(I, I_i) = 0$ when image I_i is not a nearest neighbor of image I. At each iteration, we choose a vacant cell c in the grid, and search for the image that maximizes the compatibility score,

$$I_c = \underset{I}{\operatorname{argmax}}\, C(I, c).$$

To reduce the search space, we only consider images which are nearest neighbors of the reference images. We exclude images that are already present on the map from the candidates list, to avoid repetitions. Since the number of adjacent cells is bound (depending on the grid size that is chosen), the computational cost of

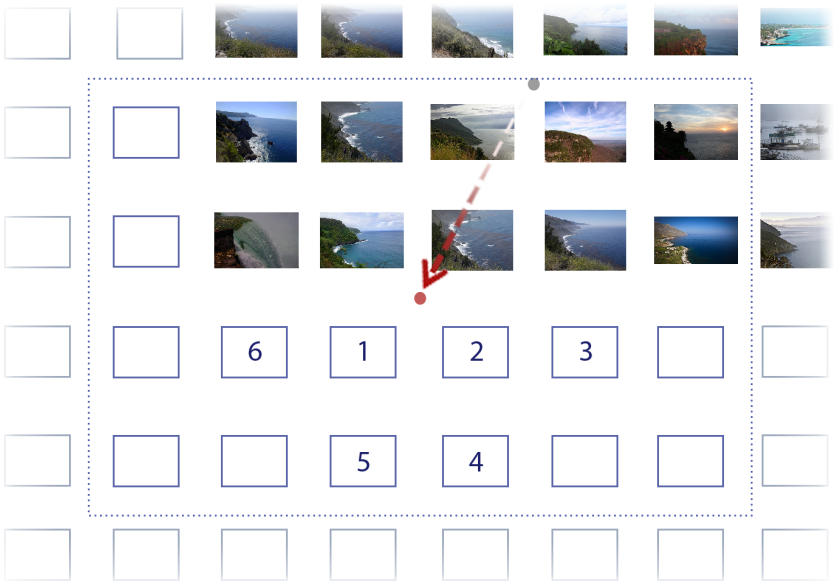


Figure 2.3: *The map is filled in an order relative to the direction of browsing. In this example, the user dragged the map two images up and one image to the right. The gray dot and red dot, respectively, mark the previous and new center of the viewport. The numbers state the order in which the first six cells on the map are filled.*

creating the map amounts to a small constant, independent of the dataset size. This allows creating the map on-the-fly, during user interaction.

The voting process gives precedence to cells that are filled early in the map generation process. We use this to further enhance the user experience, by selecting the vacant cells in accordance with the user actions. In general, we give precedence to cells that have the most filled cells which are direct neighbors in the 8-connected grid. However, since the map is a regular grid, often there will be ties and many cells will have the same number of reference images, for example along the edge of the previous region of interest. We break ties using the following process. We compute vectors from the previous center of the map to each cell, and to the new center of the map. We then select the cell with smallest angle between the map's center vector and the cell vector. This causes the grid to start growing from the user's focus area on and outwards into the rest of the map.

Figure 2.3 illustrates the order in which empty cells in the grid are filled. The user drags the map two cells up and one cell to the right. The center of the user's viewport thus moves on the map in the opposite direction; two cells down and one cell to the left. The cells marked with numbers will be filled first in their

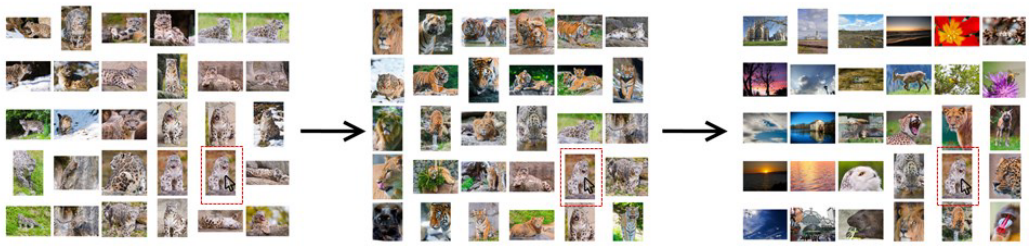


Figure 2.4: *Zooming out. The user hovers over an image and uses the mouse scroll to zoom out. The image stays as a reference point and the images around it are retrieved from a higher zoom level (the red box is only for illustration and is not part of the interface).*

respective order, followed by the rest of the cells on the grid. Existing images which are closer to the panning direction effectively have more weight in the map generation, since their neighbors are selected first.

The map-filling algorithm is simple and easy to adjust to custom graphs. It can be applied to graphs of any shape, and does not require regularity or planarity. Supporting weighted graphs requires a minute change in the compatibility score.

2.5 Interface Enhancements

2.5.1 Zoom levels

Our dynamic maps support zooming out to see a larger variety of images, and zooming in on a region to see more similar images. We support zooming operations by selecting a hierarchy of *high-level delegates* that represent every image in the dataset. All images in the dataset are contained in the first zoom level; every delegate in the second level represents a group of images in the first level, every delegate in the third level represents a group of delegates in the second level, and so on. For each zoom level we connect the delegates in a nearest neighbors graph as described below.

When the user is browsing the map in zoom level l, only images of level $l' \geq l$ are displayed, and the k-NN graph of level l is used. Note that higher level delegates are not excluded from the map when browsing lower levels, and can appear among low level images according to the low level k-NN graph. For zoom in and zoom out operations, we keep the central image as seed image (or the

image the user is focused on while performing the operation) and rebuild the map around that image using only images that belong to the updated zoom level, as displayed in Figure 2.4.

Delegate selection can be implemented using various algorithms, as long as every image has at least one delegate in its nearest neighbors list. In our implementation we use a straightforward algorithm, which can be done once for the whole dataset or incrementally when new images are added. For each image in the dataset, we check whether one of its nearest neighbors is already a high-level delegate. If none of the nearest neighbors of the image is a delegate, the image itself becomes a delegate for all of its neighbors. The same process can be done when adding a new image to an existing dataset.

Next, a list of high-level nearest neighbors is created for each high-level delegate M. A high-level neighbor is a delegate M' that has at least one common nearest neighbor with M. The score of the high-level neighbors is the maximum accumulated score of the path in the k-NN graph that connects the two delegates:

$$S(M, M') = \max_{M'' \in Near(M) \cap Near(M')} (S(M, M'') + S_j(M'', M')).$$

If a delegate has more than k high-level neighbors, only the k neighbors with the smallest scores are kept. This process is repeated recursively on the high-level k-NN graph to create multiple zoom levels. The list of high-level delegates and their k-NN graph is computed as part of the pre-processing, so there is no additional computational cost for browsing when there are multiple zoom levels.

2.5.2 Focusing on an image

We provide the user with the option to focus on a single image by double clicking on it. This regenerates the map around the clicked image in the lowest zoom level (maximum similarity). The images in the rebuilt map are then more likely to be similar to the specific image in focus rather than to the neighborhood of images around it. This option also provides the user with another way to quickly zoom in from higher levels.

2.6 Datasets and Implementation

We implemented dynamic maps for two large-scale datasets: one that contains 4,573 shapes, and another with one million images. The implementation details of the each of these systems are described below. We conducted separate user studies to evaluate these two systems. The results of these evaluations are in Section 2.7.

In both implementations, the user interface displays a grid of shapes, pre-rendered to image files, and enables the user to navigate in the shape space by dragging the mouse cursor over the shapes. The grid pans according to the drag command similar to the way it is done in online maps. As soon as the user releases the mouse button when dragging, the map is populated with new shapes. For zooming, we provide an interface similar to online mapping services, e.g. Google Maps, in which the user sees the current zoom level and can click to zoom-in or zoom-out of the current map. The map can be initialized from a random location or from a manually set location, at the user's discretion. In addition, we provide a double-click feature which allows the user to quickly focus on a single shape.

The implementation is divided into two separate systems. The user interface and map generation algorithm were implemented as a single system using C#, which can be linked with different types of datasets. The input of this system are the thumbnails that represent the shapes or images, and a list of k nearest neighbors for each element. Computing the shape and image descriptors and finding the k nearest neighbors of each element was done as a pre-process in Matlab. After the computation of the nearest neighbors, there is no difference between shapes and images which are both represented by a thumbnail in the interface system. Thus, in the following paragraph we refer only to images even though it applies to shapes as well.

During navigation, new images are loaded almost instantly after every navigation action. Internal profiling of the system shows that the map generation algorithm takes between 0.001 and 0.02 seconds for each page, depending on the number of new images that are fetched. The bottleneck of our system is loading the representative image files from disk which takes a portion of a second. This proves that the algorithm is suitable for handling large datasets with ease. Since the number of candidate images for each cell in the grid is bounded by a constant, regardless of the number of images in the dataset, the time complexity

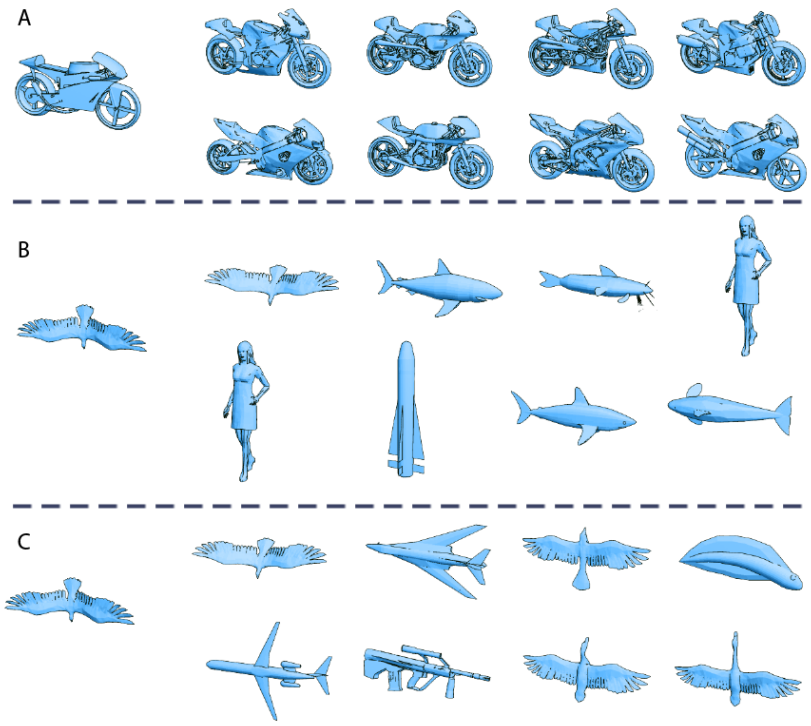


Figure 2.5: *Nearest neighbors of two shapes. (A) A case where LFD works well. (B) Using LFD yields one relevant model out of the first eight. (C) Using a combination of all descriptors yields four relevant models out of the first eight.*

of displaying the map should be the same for very large datasets that contains millions of images. The space complexity is linear since only k nearest neighbors are kept for each shape, so running the system with a very large dataset does not require extraordinary computational resources.

2.6.1 Shapes

Our shapes dataset consists of $4,573$ shapes, collected from two SHREC datasets [Vanamali *et al.*, 2010; Li *et al.*, 2012b] and the *Shape COSEG Dataset* [Wang *et al.*, 2012; Sidi *et al.*, 2011]. To define the nearest neighbors of each shape, it is necessary to effectively measure similarity between shapes. This is a fundamental quesion in shape analysis that is one of the focuses of this dissertation. In this project, we used existing state-of-the-art methods to compute shape similarity, as described below.

Many shape descriptors have been suggested for the task of shape retrieval. A very popular one is the *lightfield* descriptor (LFD) introduced by [Chen *et al.*, 2003]. LFD consists of rendering orthographic silhouettes of the model from ten different angles on a dodecahedron. To compare two models, the rendered silhouettes of the models are compared. All possible rotations of the dodecahedron (60 in total) are considered to compensate rigid rotations of the compared models. LFD is one of the most effective descriptors to discriminate between different shape classes [Tangelder and Veltkamp, 2004; Shilane *et al.*, 2004]. Yet, a nearest neighbors query using LFD may still contain irrelevant shapes. An example is shown in Figure 2.5.

To identify similar objects where LFD fails to do so, we consider two more shape descriptors. D2 descriptor [Osada *et al.*, 2002] is a histogram of Euclidean distance between pairs of points on the shape. The pairs are sampled in a way that ensures invariance to triangulation. Last, we compute a histogram of the discrete Gaussian curvature [Meyer *et al.*, 2002; Atmosukarto *et al.*, 2005], sampled over each vertex in the shape. Each of the descriptors has different strengths and weaknesses, and we aim at combining the results from all descriptors to identify different aspects of similarity between objects.

The list of nearest neighbors for each descriptor is kept separately along with the distance or score of the nearest neighbor. The lists are then merged to a single list of k nearest neighbors by computing a normalized score for each candidate that appears in one or more lists. The normalized score is based on the relative distance between the neighbors compared to all other shapes in the dataset. This way the descriptors which are more meaningful have a higher weight. Figure 2.5 shows an example of nearest neighbors search for two models. For each model on the left, the nearest neighbors are displayed on the right, ordered in two rows from left to right by their relevance score. For the bike model, LFD descriptor works well, and indeed, our feature selection method retrieves the same models as LFD alone. For the bird model, only the first model retrieved by LFD is relevant. Note that there are several other bird models in the dataset which are a better match for the query object, as can be seen in Figure 2.5c. Using our feature selection mechanism, we retrieve four relevant models out of the first eight, where the first nearest neighbor is retrieved using LFD and the rest are retrieved using D2 descriptor.

2.6.2 Images

For this dataset, we downloaded one million images in the public domain (creative commons) from Flickr image hosting service. The image collection spans photos with an upload date within a range of 400 days, where for each day in the range a few thousands of random images were selected. This has resulted in a diverse dataset which contains images of many different types, such as landscapes, urban areas, people, wildlife, birds, vehicles and more. Computing the k nearest neighbors for each image was done as a pre-process using Matlab with k=20 and took a few hours for the entire dataset.

We find the k nearest neighbors of every image using three image metrics, or image descriptors. The distance between two images in each descriptor space is the Euclidean distance between the image descriptors. Average color and color histogram are popular descriptors used in image retrieval [Deselaers *et al.*, 2008]. We used them as described below combined with the spatial envelop descriptor [Oliva and Torralba, 2001].

Average Color. The image is divided into 16 segments, a four by four grid, and the average color in each segment is computed. Similar images in this metric tend to have a similar composition. Of course, the image partitioning does not necessarily need to be four by four, but we find this partitioning appealing in the sense that it seems fine enough to distinguish between images with significantly different compositions, yet sufficiently coarse to ignore small changes in composition of similar images.

Color Histogram. A joint color histogram for RGB values is computed. Each color channel is divided into four bins, to create a total of 64 bins for every color combination. The number of pixels that fall in each bin is counted and divided by the total number of pixels in the image. Similar images in this metric have similar color distributions, which suggest similar atmosphere or surrounding. This descriptor is less sensitive to translation, rotation or reflection of the images compared with the average color descriptor.

Spatial Envelope. The spatial envelope was described in [Oliva and Torralba, 2001] and named *gist* descriptor since it captures the gist or context of a scene. The gist descriptor describes the spatial structure of a scene using a set of spectral signatures which are specifically tailored for the task of scene recognition. It was shown that in the gist descriptor space, scenes that belong to the same context

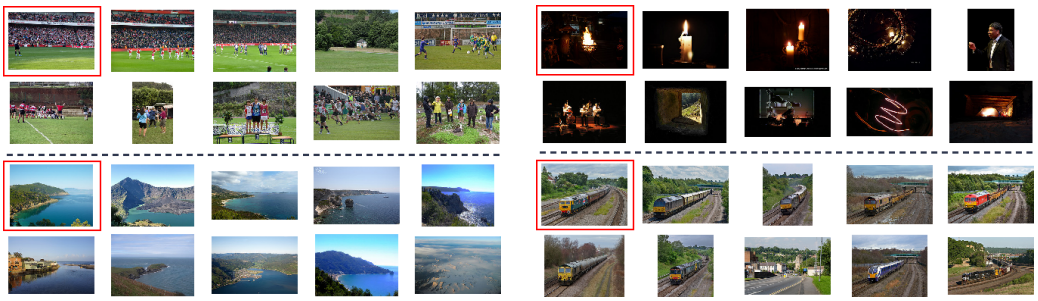


Figure 2.6: *Nine nearest neighbors are displayed for a few source images marked with a red frame.*

are projected close to each other. We use the code provided by the authors to compute the gist descriptor of every image in the dataset.

The three descriptors are calculated for each image, and k nearest neighbors are found for each descriptor space separately. The distance from the image to each nearest neighbor in each descriptor space is kept as well. The three lists are then merged to a single list of k nearest neighbors by computing a normalized score for each candidate that appears in one or more lists. Figure 2.6 shows an example of nearest neighbors search for a few images in our dataset. For each image on the left, the nearest neighbors are displayed on the right, ordered in two rows from left to right by their relevance score.

2.7 Evaluation

In order to evaluate DynamicMaps, we compared it to a relevance feedback method. Relevance feedback (RF) was chosen as the most prominent method for similarity-based browsing, and the only one we are aware of, that can support a corpus of millions of images. For simplicity, we implemented a standard RF method rather than a more complex one (i.e., that might include negative feedback). We employed a within-subject design to compare performance and attitudes of participants. The main variable interface describes the search interface used: DynamicMaps (DM), or relevance feedback (RF).

Interfaces. Both DM and RF interfaces show a grid of 6x5 images at any given time [1]. For the DM interface, we used the system as described above, initialized with the starting image at the center, around which the algorithm builds the initial

[1] When evaluating shapes, 20 thumbnails were displayed on a grid of 5x4.

screen grid of 30 images. For the RF interface, the system initializes showing the starting image on the upper left corner followed by the 29 closest neighbors on a grid. The user can then select up to 3 images and click a button (labeled "more images") to fetch the next set of images closest in similarity to the selected images (ordered by similarity). At any time, the user can press the back button and return to the previous screen. Both interfaces included a "restart" button that returned the view to the initial screen formed by the starting image. As a starting point, users could enter an image number in a provided textbox around which the system initializes as mentioned above.

The evaluations presented here for shapes and images are somewhat different for several reasons. First, this project was first developed and evaluated using the shapes dataset. The evaluation of the images dataset occured several months later, after developing the necessary adjustments to support a massive image collection (mostly in the back-end of computing nearest neighbors for the collection). Second, the experience of browsing a collection of a few thousand shapes differs substantially from that of browsing a collection of one million images. Finally, the massive collection of images allows more elaborate tests as can be seen below.

2.7.1 Shapes

We employed a 2(method) x 3(task) within-subject design to compare performance and subjective opinions of participants. The main variable, $method$, describes the search system used and included either the Dynamic Map method (DM) or the Relevance Feedback method (RF).

The second variable, $task$, describes the tasks that participants were asked to perform. Three different task types were given:

1. Choose a model out of the collection according to subjective preferences (e.g., "find a dining room chair that you would like to have in your home")

2. Find multiple models in a category (e.g., find ten different types of four legged animals such as horse, cow, dog, etc).

3. Given a specific reference image of a model, find that specific model in the collection (e.g., a model of an electric guitar with very distinct body shape was provided. The starting point was a collection of guitars).

For each task type, we devised two similar tasks to be performed. For example, for the third task, either an image of a guitar or an image of a person was given. In addition, for each task, a starting shape was determined. The starting shape was located in the vicinity of the target/s (e.g., for finding a dining room chair, the starting point was a swiveling office chair). For the DM method, the starting shape was used as a basis to create the initial grid. For the RF method, the nearest neighbors of the starting shape were presented as the initial grid.

Sixteen (16) participants took part in the experiment. Participants were mainly students from a local university. Eleven participants were male and five were female with an average age of 29.5 ($SD = 5.2$). All participants had previous experience with searching images on the Web, and no participants had previous experience with searching 3D models.

Participants were seated in front of a 22" screen with 1600x900 pixel resolution. This allowed for a grid of 5x4 models to be displayed. Participants were then presented with one of the two methods. The user interface features were first explained to the participants, who were then allowed to freely browse around the model space using the interface until they felt comfortable using it. Participants were then given the three tasks one after another and were asked to perform each task as best as possible. When they completed all three tasks, participants were asked to fill in a questionnaire on their subjective opinion of the interface. Participants were then presented with the second interface on which they completed the same procedure (using three different tasks of the same task type). At the end of the experiment, a comparative questionnaire was given. The order of interfaces (which interface was first used) as well as which set of tasks to perform on which interface was counterbalanced.

Results. Figure 2.7 presents the average amount of search time per task for both methods. A two-way ANOVA was conducted to assess the time differences between the two methods. Results indicate that it took participants significantly less time to search with the DM method than with the RF method, $F(1,15) = 44.1$, $p<0.001$. A post-hoc analysis using the bonferroni adjustment, examining each task separately, showed that there were also significant differences between the two methods in tasks 1 and 3 with task 2 being very close to significance ($p=0.052$). It should be noted that seven participants in the RF condition were unable to complete task 3 compared to only one participant who was unable to complete the task in the DM condition.

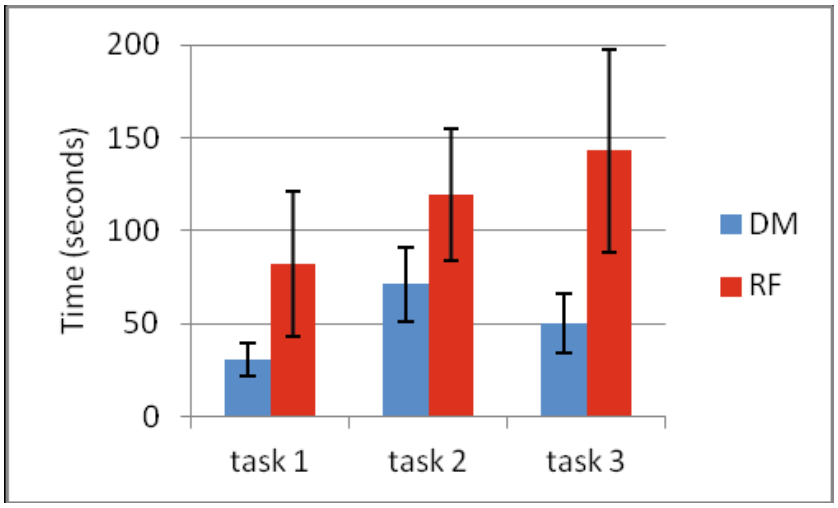


Figure 2.7: *Average search time for the three tasks in both the DM and the RF methods (N=16). Error bars display 95% confidence interval.*

Statement	DM	RF	p-value
The search was enjoyable	5.43(1.26)	3.65(1.5)	0.003
The system was effective for the search purposes	4.56(1.67)	3.50(1.71)	0.065
The system limited my possibilities	4.31(1.4)	5.25(1.52)	0.095
During the search, I stumbled across items I didn't think about	5.25(1.52)	5.06(1.94)	0.687
I easily understood how to use the interface	5.56(1.71)	6.12(1.58)	0.331
I easily understood the efficient way to conduct the search	5.12(1.45)	3.68(1.30)	0.005
It was easy to conduct the searches	5.12(1.31)	3.37(1.58)	0.007
During the search I felt frustrated	3.37(1.66)	4.93(1.73)	0.021

Table 2.1: *Average ratings and standard deviation of the two interfaces. Ratings are given on a 7-point Likert scale ranging from strongly disagree (1) to strongly agree (7). P-values of the Wilcoxon signed test, comparing the two provided interfaces.*

Next, we analyzed participants' opinion of the interfaces. Table 2.1 presents the set of statements presented to participants after interacting with each method (DM and RF) as well as their average responses. Ratings were given on a 7-point Likert scale to indicate how much participants agreed with each statement, ranging from strongly disagree (1) to strongly agree (7). With ranked ordinal data and a relatively small sample size, it is recommended to use a nonparametric statistical test [Huck *et al.*, 1974]. We therefore used the Wilcoxon signed nonparametric test to examine differences in ranking between the groups. As can be seen in Table 2.1, results indicate a preference to the DM method on almost all questions.

Statement	DM	RF	No opinion
Which system was more effective for task 1?	11	3	2
Which system was more effective for task 2?	11	5	0
Which system was more effective for task 3?	9	4	3
Which system was more effective overall?	11	3	2
When you have a vague idea of the search, which system is better?	12	4	0
When the target of the search is clear which system is better?	8	7	1
Overall, which system do you prefer?	12	2	2

Table 2.2: *Direct preferences between the two interfaces (N=16).*

Finally, we analyzed the direct comparison questions given at the end of the experimental session. These results are presented in Table 2.2.

Evaluation Summary. Overall, participants clearly preferred the DM method over the RF one. This is demonstrated both in the direct comparison results (Table 2.2) and in the independent ratings of each interface (Table 2.1). Participants felt that the DM interface was more enjoyable, effective, efficient and easy to use. Together, these measures are used as an indicator of a system's usability [Brooke, 1996], thus our results suggest that the DM method is more usable than the RF method for the given search tasks. Better efficiency is also indicated by the fact that participants completed their tasks faster using the DM method.

2.7.2 Images

Tasks. Tasks were designed to be open-ended and reflect real-world search needs (similar to [Rodden *et al.*, 2001; Yee *et al.*, 2003]). Two general tasks were defined for the within-subject design. In each task participants were asked to find images that would best fit text slides of a given presentation. For example, the first presentation was on a non-profit organization titled "the society of preservation of nature". The initial slide was a title slide, the second slide talked about the organization's mission, the third slide talked about the history of the organization and the final slide talked about the major active projects the organization employs today. All slides included only text with no color or graphic design. Participants were asked to find up to three images that would best fit each slide. The second presentation was similar in nature and had to do with architecture. For each task, participants were given four starting points in the interface. This emulated four possible keyword search queries. The starting

points were chosen as single images relevant to the task (for example, images of animals or nature for the previously mentioned task).

Participants. A total of 24 participants took part in the study, 11 were male and 13 were female with an average age of 27.1 ($SD = 5.1$). Participants were mostly students of a large university from a wide range of departments and faculties. All had normal or corrected-to-normal eye vision. 15 participants reported searching on the Web for images every week, while 5 participants reported searching every two weeks or so and 4 reported a lower rate. Image search task reported including finding images for presentations, looking for images for study purposes, looking for products and more. Most participants indicated using Google images as their main image search tool.

Procedure. Participants were seated in front of a 22" screen with 1440x900 screen resolution. Participants were then presented with one of the two interfaces. The user interface features were first explained to them, after which participants performed one practice task on which they were instructed to use the interface until they felt comfortable with it. Participants were then given one of the two tasks and were asked to perform the task as best as possible. No time limit was given for the task. After they completed the first task using the first interface participants were asked to fill in a questionnaire asking their subjective opinion of the interface they just used. Participants were then given the second interface on which they completed the same procedure using the second task. All interactions with the interfaces were logged and later analyzed. At the end of the experiment a comparative questionnaire was given and participants were asked to comment on each interface. The order of interfaces (which interface was first used), as well as which task set was used with which interface was fully counterbalanced, creating four different configurations (six participants in each configuration).

2.7.2.1 Results

Order effects. To rule out order effect (whether participants started with the DM or the RF interface), we performed a between-subject ANOVA with interface order as the independent variable on both task completion time and on number of unique images seen. No effect was found for both variables. Next, to ensure there were no differences between tasks we performed a within-subject ANOVA with task as an independent variable. Again, no effect was found for both task completion time and number of images seen.

	DM	RF	F	p-value
Images seen per minute	230.0 (51.9)	104.5 (51.0)	98.2	<0.001
Unique images seen per minute	98.4 (21.1)	49.3 (18.6)	107.6	<0.001
Task completion time in seconds	805.8 (334.4)	761.5 (348.9)	0.55	0.47

Table 2.3: *Average (and standard deviation) number of unique and non-unique images seen per-minute, and task completion time.*

Completion time. On average, it took participants 805.8 seconds (13.5 minutes) to complete the task in the DM interface (SD = 334) and 761.5 seconds (12.7 minutes) in the RF interface (SD = 348). A one-way repeated-measures ANOVA on task completion time did not find these differences to be significant, $F(1,23) = .55$, $p = .47$.

Amount of Interaction. We compared the amount of user interaction with each of the interfaces. In the DM interface, an interaction is performed either by dragging the mouse to pan the view in order to bring up more images (number of pans), by zooming in or out (number of zooms), or by doubleclicking on a single image to bring it to the center. In the RF interface, an interaction translates into a "more images" or "back" press which brings up the next or previous set of images (number of presses). Thus, we compared the number of pans + zooms + double clicks in the DM interface with the number of combined "more" and "back" presses in the RF interface. Results indicate that there were many more interactions per task in the DM interface (M = 158.4, SD = 93.3) than in the RF interface (M = 35.9, SD = 31.4). A one-way within subject ANOVA on number of interactions showed these differences were significant, $F(1,23) = 90.9$, $p < 0.001$.

Amount of images seen. Analyzing the log files, we summed up the amount of images seen in each interface. We examined both the total amount of images seen in a specific task, and the total amount of unique images seen, since some images may appear several times during the same task. With the RF interface, the total amount of images seen is equal to the number of interactions (as listed above) plus 1 (for the initial screen) times 30 (each screen showed a grid of 6x5 images). In the DM interface, each pan adds a different amount of images to the screen depending on the pan position. We counted the 30 initial images, and then added the newly filled images in each pan. A zoom, restart, or doubleClick event brought 30 more images. For the unique images seen, in both interfaces, we counted the unique images presented from the beginning till the end of the task. Because there were large individual differences in task completion time, we

Figure 2.8: *Average number of pans made and number of images seen in each of the five zoom levels (level 1 being images that are most similar to each other).*

normalized these results over time and measured the total number of unique and non-unique images seen per minute. Results, presented in Table 2.3, indicate a large, significant difference in both total and unique number of images seen per minute. Users using the DM interface have seen significantly more total images per minute than when using the RF interface, $F(1,31) = 98.2$, $p<0.001$. Users using the DM interface have also seen more unique images per minute than users using the RF interface, $F(1, 23) = 107.6$, $p < 0.001$.

Zooming. All participants used the zooming feature often, with an average of 39.6 times per session (or 2.94 zoom events per minute). To better understand the usage of Dynamic Maps, we analyzed the use of the zooming levels. Figure 2.8 shows the number of pans made and number of (non-unique) images seen in each zoom level. As can be expected, most interaction was done in the first zoom level, with interaction dropping heavily after the first level.

Subjective opinions. After using each interface, participants were presented with a set of statements and were asked how much they agreed with each one on a 7-point Likert scale ranging from strongly disagree (1) to strongly agree (7). Table 2.4 presents these statements and the visitors' responses with both interfaces. A Wilcoxon Signed Ranked non-parametric test did not find significant differences in ranking of any of the statements between the two interfaces.

At the end of the experiment, we presented participants with a final ques-

Statement	DM	RF
The system was efficient for the search tasks	4.20 (1.14)	4.15 (1.22)
The system limited my options	4.75 (1.69)	4.91 (1.28)
The search was fun	4.63 (1.24)	4.33 (1.16)
I quickly understood how to use the interface	5.87 (1.19)	5.87 (1.11)
During the search I felt frustrated	3.25 (1.64)	3.54 (1.84)
I am satisfied with the images I picked for the presentation	5.10 (1.25)	4.83 (1.00)

Table 2.4: *Participants average ratings (and standard deviation) per interface on a 7-point Likert scale ($N = 24$).*

Statement	DM	RF	No pref.
Which system was more efficient?	12	8	4
When you have a vague idea of the search target, which system is better?	15	7	2
When the search target is clear, which system is better?	7	15	2
Which system is best to see a wide variety of images?	16	5	3
Which system is easier to learn?	6	7	11
Which system do you prefer overall?	15	8	1

Table 2.5: *Number of participants preferring each interface on a list of criteria ($N = 24$).*

tionnaire asking them for their preference of interface on a list of criteria (Table 2.5). Results indicate a general preference toward the DM interface, although preference was not absolute. Most participants thought the DM interface was more efficient and preferred it overall. It is interesting to note that most participants thought the DM interface is better when there is a vague idea of the search target and for seeing a wide variety of images, while there was a general preference for RF when the target is clear.

2.7.2.2 Discussion

Our results indicate that Dynamic Maps provide a more interactive experience for the users and allows them to view a wider variety of images than previous methods. Participants viewed many more images (both unique and non-unique) per time with the DM interface compared to the RF interface. While the way of interacting in the two conditions is quite different, still, many more interaction events were measured in the DM compared to the RF interface. Thus, it seems that participants viewed more images by actively interacting more with the interface. It should be noted that we cannot be sure that participants actually saw all the images that were displayed on screen. RF actually forces the user to more closely examine each image, while DM better supports scanning through

images. This may help to explain the large difference in the amount of presented images.

Dynamic Maps provides immediate and continuous interactive feedback that does not require the user to make conscientious sequential selections, but rather asks the user to visually choose a direction to follow based on general perceptive cues. Thus, it affords easier and faster movement in the image space, with less sense of commitment, enabling the user to see a wider variety of images (a fact also realized by participants in the subjective preference questionnaires). This can also be look at from a cognitive load perspective. Cognitive load in the information retrieval context can be seen as a measure of information processing effort a user expends to comprehend the visual stimuli and interact with the system [Hu *et al.*, 1999]. Using the RF interface, the user needs to go over every image and explicitly provide a relevance judgment on the image, a process that requires a high state of cognitive load [Back and Oppenheim, 2001]. Dynamic Maps are less cognitively demanding since the user does not need to make a decision regarding each and every image, but can rather follow general visual cues. As one participant wrote, *"I prefer DM. Less mouse clicking. Dragging is easier then thinking of which images will bring me closer. In DM you can see a larger range of images at once without the need to choose and click over and over"*.

Having easier interaction capabilities and viewing more images per time unit is more useful when the search is vague and it might be difficult to select specific images that lead directly to the target. It is then easier to experiment, and follow one or more visual search directions than to select specific images. Another advantage of faster and more interactive browsing is that it can better support serendipity in the search process, since users interact more and may stumble upon different areas. It is easy for users to explore regions they may not have envisioned. This was reflected in a statement of a participant: *"It is possible to reach different directions, thoughts and ideas that I have initially not thought about"*.

Zooming was often used and was referred to by participants as being very useful. The Zooming option enabled the users to step back and get a wider view of the current corpus. It also supports getting a more diverse view, with the diversity level controlled by the user. Furthermore, using zoom out and pan, the user can view the different topics and content available in the current corpus using simple interactions. This can be useful to get an overview of the image corpus.

No overall significant difference between the interfaces was found for task completion time. Completion time is often looked at as a measure of efficiency. However, in the current study, the task was open-ended and participants were asked to take as much time as needed to find the best possible images. Thus, we do not think that in this case completion time is an indicator of efficiency or quality. On the contrary, it might be that more time spent on the task indicates that the interface was more engaging and caused users to search more thoroughly. Similarly, other studies have found no correlation between task completion time and quality of results or user satisfaction [Rodden *et al.*, 2001].

Finally, we note that many participants mentioned that Dynamic Maps were enjoyable and the interaction with it was much more smooth and fun to use than the RF interface (e.g., *"The [DM] system is enjoyable, it is easy to operate and it naturally flows"*). We believe that this will be highlighted even more when using the system with touchbased interfaces. With its pan-based interaction, Dynamic Maps should be ideal for searching images on a Tablet computer, for which the playfulness of Dynamic Maps would be even more prominent.

2.8 Conclusion

We present Dynamic Maps, a technique for browsing a very large set of shapes, images or any other high-dimensional object which can be represented by a thumbnail. In our method, objects are laid out on a two-dimensional dynamic map that is locally updated according to user navigation. Dynamic Maps enable a smooth, fast and more interactive experience that is best suitable for exploratory search, when the search target is vague. It is also useful for serendipitous browsing in exploring regions not envisioned by the user and for getting a wider view of the corpus. Further work may explore using semantic information in the similarity measures as well as combine Dynamic Maps with keyword search.

One of the most prominent features of our approach is the locality of the solution. The local approach enables the construction of an unconstrained, easy to use and highly scalable system; it can support massive datasets containing millions of models with ease. It can also easily handle frequent changes in the dataset. The local nature of the algorithm allows for a seamless addition of shapes or images, and other on-the-fly changes. This cannot easily be accomplished by other global feature preserving techniques. At the same time, some limitations stem from this locality.

Since we do not keep models outside the current boundaries of the map, models may be repeated during a browsing session, appearing at multiple locations on the map. In practice, it is possible to prevent some repetition of models by excluding models that were recently seen from the search space and remembering previously generated patches on the map. However, this requires a delicate balance, since keeping previously seen regions of the map creates global constraints that often cannot be fully satisfied. Informal feedback from participants in our user study suggests that users do not feel the repetition of models is hindering the user experience, since it is usually easy to avoid by navigating away from seen models, or using the zoom ability to view a greater variety.

Another limitation of our system is that it may be difficult to find non-dominant concepts or particular images. The appearance of a certain image on the map does not guarantee the appearance of all similar images. Rather, only similar images which match the current browsing direction are displayed. Thus, a specific image may be hard to locate. If a concept rarely appears, the user will be unlikely to find it as it will be hidden within another area. This is due to the voting mechanism which ensures only shapes that are relevant to the surrounding appear on the map, thus pruning outliers.

The presented method is most suitable for free-form search, where the user does not have a specific target in mind, and the goal is to browse a variety of shapes rather than retrieving the single most relevant shape. A primary goal of the dynamic map is to aid the refinement of 3D object search. As such, it is our vision that the technique is used in tandem with keyword shape search. In such a setup, the dynamic map can be seeded around an shape which is the best match for the textual keyword search, to provide the user with a variety of objects that resemble the best match. The map generation method is decoupled from the construction of the k-NN graph, which makes the method applicable for other domains as well, such as searching text documents or any kind of high dimensional data.

3 Semantic Similarity from Crowdsourced Clustering

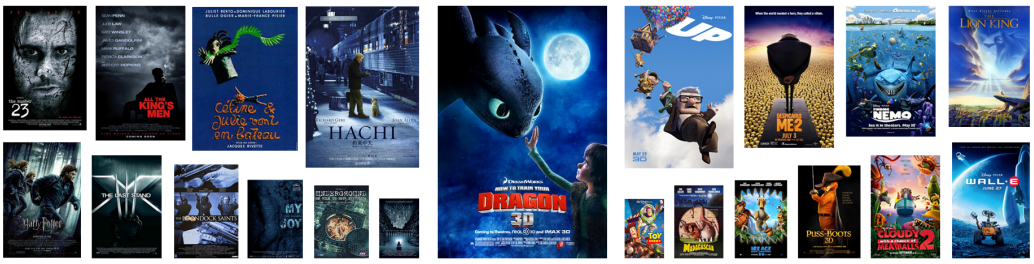


Figure 3.1: *Nearest neighbors of the center image in a collection of movie posters, computed using image descriptors (left), and crowdsourced queries (right). Smaller images mark farther neighbors.*

It is extremely hard to define a distance metric that would capture well the intuitive or semantic similarity between images (see Section 1.1). State-of-the-art analytical methods for computing such a metric fall short when similarities are derived from a broad semantic context. Consider, for instance, the similarity between the movie posters in Figure 3.1. Identifying such similarities is usually easily done by a human observer, but pose a hard computational problem nonetheless.

The natural solution is thus gathering information about semantic similarities between images from people, for example using a crowdsourcing technique.[1] This approach was taken in recent work to collect style similarity measures [Lun *et al.*, 2015; Saleh *et al.*, 2015]. The typical comparison task that the crowd performs

[1] *Crowdsourcing* is a general name for processes that involve posing many small-scale tasks to the crowd of web users, and piecing together the crowd's answers to achieve a larger-scale goal, such as constructing a large knowledge base.

is of the following form: given three images A, B, and C, choose whether A is more similar to B or to C (a *triplet query*). Assuming consistent query responses, querying every image triplet yields the full relative similarity metric over the set of images. However, the number of triplets is prohibitively large. Thus, typically only a sample of the triplets are queried and the rest are estimated based on extracted image features [Lun *et al.*, 2015; Saleh *et al.*, 2015]. In addition, such queries lack context which is often necessary in order to perform comparison tasks (see Figures 1.3 and 1.4).

In this work, we propose an alternative approach for learning image similarities based on *clustering queries* posed to the crowd. Instead of queries of three images, crowd members are given a small set of images and are asked to cluster them into bins of similar images using a drag-and-drop graphical UI (see Figure 3.2). While a single clustering task requires more effort than comparing three images, our approach has two important advantages. First, the results of a single clustering task provide a great deal of information that is equivalent to many triplet comparison tasks. Images placed in the same bin are considered closer to one another than to images in other bins, so triplets can be formed from each pair of images in the same bin along with any third image from another bin. Second, each query provides crowd members with additional context that assists them in performing a more faithful and meaningful comparison.

A key observation of this work is that a similarity metric can be constructed more efficiently by performing comparisons on similar images rather than non-similar ones. This is true in particular in the context of semantic similarities, where local similarities are often more meaningful. Following this observation,

(a) (b)

Figure 3.2: *An example of the clustering interface. (a) The user is presented with* 20 *images to cluster into the four bins on the right. (b) The bins may contain as many images as necessary. When all images are clustered, the user can submit the query and receive another one.*

we develop a novel, adaptive algorithm that aims to generate queries that are as local as possible. The challenge here is that similarities are unknown in advance. Thus, our algorithm works iteratively. At each phase we generate and pose clustering queries to the crowd. As information is collected, we progressively refine the queries to focus on similar images in a narrower local neighborhood. Local similarity comparisons are embedded in Euclidian space to obtain a refined estimation for the global similarity metric. This refined metric is then leveraged for computing more locally-focused queries in the next phase. This progressive method efficiently converges to a meaningful similarity estimate.

Evaluation and experimental study. To test the efficiency of our approach, we implement our technique in a prototype system, and use it to conduct a thorough experimental study, with both synthetic and real crowd data. First, we test our technique over two image datasets where the ground truth is known, examine the results and compare them to a baseline approach that uses the same number of queries but chooses them randomly. Second, we compute the k-NN images for real-world image datasets, where the ground truth is unknown, and evaluate the results manually. Last, we study the effect of parameters such as the number of phases and queries in a series of synthetic experiments. Our experimental results prove the efficiency of our approach for computing semantic image similarity based solely on the answers of the crowd, while using a relatively small number of clustering queries.

Throughout this chapter we again focus on similarity between images. However, since we rely solely on crowd queries, our method is suitable for any objects that can be represented by images or thumbnails. In our experiments, we estimate similarities between 3D shapes (represented by a single rendered image) and fonts. Other possibilities include words, videos (represented by a few significant frames), celebrities, and more. Crowd members can also be instructed to relate to specific properties of the presented object. For example, they can be instructed to cluster the faces of politicians according to their views, rather than according to their physical similarity.

3.1 Related Work

The classification of images is a well-studied problem. A common paradigm is based on image descriptors, such as the color histogram of images, SIFT based descriptors [Lowe, 1999], or GIST descriptors [Oliva and Torralba, 2001]. The

distance between two images is defined as the Euclidean distance between the image descriptors, on top of which machine learning techniques can be employed to find similarities or clusters of the images (e.g., [Wang *et al.*, 2009; Zha *et al.*, 2008]). Other methods employ a bag of features (BoF) approach, using visual segments [Sivic and Zisserman, 2003] and/or textual annotations, either attached to the images manually or from the textual context of a web page (e.g., [Wang *et al.*, 2009; Zha *et al.*, 2008]). However, such methods fall short when classification relies on semantically-rich features, which may be hard to learn from the images, and may only be partially reflected in the labels.

Semi-supervised learning methods can alleviate the problem of lacking semantic features. These methods rely on manual labeling of a small set of image pairs or triplets, rather than per-image labels for the entire set. A large body of work has attempted to classify images by using pair-wise labeling consisting of equivalence (or inequivalence) constraints, i.e., whether or not the pair belongs to the same class [Bar-Hillel *et al.*, 2005; Biswas and Jacobs, 2014; Weinberger *et al.*, 2005; Xing *et al.*, 2003]. Triple-wise constraints are more relevant to *relative* comparisons of images, as they compare the distances of two image pairs [Frome *et al.*, 2007; Lun *et al.*, 2015; O'Donovan *et al.*, 2014; Saleh *et al.*, 2015; Tamuz *et al.*, 2011]. The constraints can then be used to learn a distance metric between images. In particular, the work of [Tamuz *et al.*, 2011] focuses on adaptively selecting optimal triplets based on crowd input. In the recent work of [Lun *et al.*, 2015; O'Donovan *et al.*, 2014; Saleh *et al.*, 2015], triple-wise comparisons have been collected from crowd members in order to learn about style similarities. While these studies highlight the need in collecting similarity comparisons from the crowd, the use of triplet comparisons has shortcomings that our work addresses: this approach requires many crowd tasks, and users are not given context for comparison. These shortcomings were also noted by [Wilber *et al.*, 2014], a study that focuses on redesigning the user interface to derive more image comparisons from each crowd task. This is done by asking users to select the X most similar images to a given image, out of a set of Y images. The new interfaces of [Wilber *et al.*, 2014] is a step forward from triplets, but in contrast with our work, their study does not consider how to effectively choose images to compare.

Another work highly related to ours is *Crowdclustering* [Gomes *et al.*, 2011], which considers clustering images with the crowd. Each crowd member obtains a sample of a few images (a *query*) and classifies them into groups. This input

is used to train a Bayesian model which estimates the ways different crowd members may classify each image. This work resembles ours in letting the user cluster a small set of images, and also in the idea of refining the clustering results by re-applying the technique on the obtained clusters. However, their technique is not designed to compute image similarities. In contrast, we employ the progressive refinement to determine image similarities with faster convergence. We compare the performance of our techniques with [Gomes *et al.*, 2011] in Section 3.3.

The work of [Yi *et al.*, 2012] suggests to only obtain query answers for a small fraction of the data, and use dedicated matrix completion techniques to complete the missing classifications, rather than requiring that every image appears in at least one query as in [Gomes *et al.*, 2011]. This work is orthogonal to ours, and can be employed in our case if the number of queries that can be asked is small relative to the number of images.

Crowdsourcing has been employed for tasks related to ours such as record matching based on images [Marcus *et al.*, 2011], grouping and top-k [Davidson *et al.*, 2013], and entity matching [Wang *et al.*, 2012]. However, no previous work has considered the problem of learning an image similarity metric, nor can be applied in a straightforward manner for this task. For example, k-NN may be viewed as finding the top-k most similar images for each image; however, applying the method of [Davidson *et al.*, 2013] for each image separately is inefficient.

3.2 Algorithm

We next describe our method of generating queries to the crowd based on an estimated similarity metric, and of refining the similarity metric based on answers from the crowd. We aim to use queries that involve images from the same local neighborhood, which are more effective for determining the global similarity metric.

Our algorithm generates clustering queries by selecting sets of n_q images. The answer obtained from crowd members is a division of this image set into n_c clusters. The crowd is a relatively expensive resource in terms of latency, human effort, and often monetary cost as well. Therefore, in many practical cases, the total number of queries that can be asked is restricted by a predefined budget. Given such a budget, the goal of the algorithm we develop is to utilize the queries in the best way possible, by considering only local neighborhoods. This yields

an iterative process, where local neighborhoods change according to queries results.

Our method estimates local distances by maintaining an embedding of the entire set in Euclidian space, in which the distances are calculated. The embedding is initialized randomly, and local neighborhoods are progressively improved. The embedding ensures that even distances that were not queried are consistent with the partial information derived from queried distances. To improve the embedding of local neighborhoods, we pose queries to the users in small batches, and update the embedding after each batch. Interestingly, querying local neighborhoods of the embedding proved beneficial even in early stages when the images are not necessarily semantically close, since such queries provide many constraints on the same neighborhood. In addition, in each iteration we wish to preserve the close neighbors which are already semantically similar. Even in a random embedding, local neighborhood based queries help to detect and preserve cases where some neighbors are also semantically similar.

The main steps of the algorithm are illustrated in Algorithm 1: As input, the algorithm takes the total number of allowed queries (`budget`) and the number of queries to generate at each iteration (`batch_size`). The results of the queries are integrated into the embedding (**E**) and the induced global distance metric (**D**). The output of the algorithm is the distance metric computed based on the last, most refined embedding.

Clustering query. For a set of images $\mathcal{I}$, we define a query Q as a subset of $\mathcal{I}$ containing n_q images. The answer to each query is a division of Q into disjoint clusters $C_1, \ldots, C_{n_c} \subseteq Q$. From these answers we extract similarity comparisons:

Algorithm 1: CrowdSter(budget, batch_size)

```
 1: E = EmbedData()        // random embedding
 2: num_of_queries = 0
 3: while num_of_queries < budget do
 4:    Q = SelectQueries(E, batch_size)
 5:    R = RunQueries(Q)   // using the crowd
 6:    D = DistanceFromEmbedding(E)
 7:    D = UpdateDistances(D, R)
 8:    E = EmbedData(D)
 9:    num_of_queries += batch_size
10: end while
11: D = DistanceFromEmbedding(E)
12: Output D
```

given two images x, y in cluster C_i, and a third image z in a different cluster C_j, we infer that $\Delta(x, y) < \Delta(x, z)$, where Δ represents the similarity metric. As n_q increases, we obtain more comparisons, but the number of images in a query should be small enough to allow a crowd member to view them [Marcus *et al.*, 2011]. In our experiments, we found that $n_q = 20$ is a good balance of this tradeoff between effectiveness and simplicity. Following this, we found that setting the number of clusters n_c to 4 is optimal, as it balances between inferring more comparisons (smaller n_c values) and quickly pruning less similar images (larger n_c values).

Generating queries. Queries are generated in our algorithm based on the embedding from previous phases. In each phase, we generate queries that (a) are local, and (b) cover the set of images as evenly as possible. To do so, we sample random images uniformly while making sure they are not nearest neighbors of each other. When no such samples remain we start over. For each sampled image, we find its k nearest neighbors in the embedding. Then, out of these neighbors we sample a random subset of size n_q and use it as the next query.

Embedding. We maintain an embedding of all images in the dataset in a d-dimensional space. The embedding infers a consistent distance between every pair of images, to be used in the next phase, and is gradually improved with each batch of queries. In our experiments, we used $d = 6$. We also experimented with higher values of d, but there was no significant effect on the efficiency of our method. Before the first queries are sent to the users, the images are embedded into the Euclidian space using a uniform random distribution. To gradually improve the embedding, we calculate the distance between each pair of images in the embedding, update the distances according to the query results, and embed the images again using the updated distances. This consolidates the updated distance and resolves any inconsistencies among them. To compute the embedding we use multidimensional scaling (MDS), whose input is the distance between each pair of images.

More specifically, we want to find an embedding by taking into account only distances that we have information of (via query results), ignoring all other distances. For this we use Sammon Projection [Sammon, 1969], which is a multidimensional scaling technique that computes an embedding using a stress function and gradient descent. The weighted stress function can take into account the relevant distances and ignore other distances by giving them a very small weight. All weights are initialized to a very small value ϵ. In each phase, we

set the weight for each updated distance to 1. Distances that were updated in previous phases maintain a weight of value 1, so once a pair of images is queried its distance is always taken into account when computing the embedding in subsequent phases.

Updating the distance. To update the distance, all the query results in the batch are aggregated and analyzed. For each pair of images in each query, we refer to a query result as *positive* if the images were assigned to the same cluster, and *negative* if the images were assigned to different clusters. The distance between a pair of images is shortened if the pair has more positive than negative query results, and made longer if the pair has more negative query results. The distances between pairs of images for which there was a tie and pairs of images that did not appear in the same query are not affected.

Distances are shortened by dividing by β and are made longer by multiplying by β. In our experiments β is set to 4. Note that we do not take into account the number of times a pair of images appeared in the same batch of queries. For example, a pair of images that has two out of two positive query results is updated in the same manner as a pair of images that has three out of four positive query results. Since the phases tend to be short, the probability that the same pair of images will appear in many queries is small, and inferring from the exact ratio between positive and negative results is too sensitive to randomness.

3.3 Experiments

To evaluate the efficiency of our approach, we conduct three sets of experiments, described below. First, to verify the correctness of our approach, we conduct a set of small-scale experiments for a data set where the ground truth is known. This ground truth allows evaluation of the result quality. Second, we test the practicality of the approach for semantically-rich image similarities, using larger sets of images, where the ground truth is unknown. Finally, to further investigate each component of our solution, we conduct synthetic experiments where the ground truth similarity is known, and crowd answers to queries are simulated accordingly instead of using real crowd. We vary different parameters of our system, and observe the effect on the output quality. In all sets of experiments, we further compare the results we obtained to alternative, baseline algorithms.

- **Random:** Randomly select queries, equivalent to executing our algorithm in a single phase.

- **Crowdcluster:** Using the method of [Gomes *et al.*, 2011]. The results of this method are targeted to identify clusters, but also include a mean spatial location for every image, which we use as an alternative to our embedding.

- **Feature-based:** Estimate the similarity of images based on automatically extracted image features, which serves as a baseline where ground truth is not available.

Implementation and crowd UI. Our crowdsourcing system includes a dedicated, user-friendly crowd interface. The UI of the system is implemented on the Google App Engine platform. The back-end analysis of the crowd answers and the computation of the next queries to be posed to the crowd is performed in MATLAB R2014b. A screenshot of the UI is shown in Figure 3.2. Initially, we display 20 images on the left-hand side of the screen (the query), and the crowd member is asked to drag and drop the images in one of the 4 right-hand side bins (and also move images between bins). Crowd members can also decide to leave images outside of any bin if they are unrelated to any of the other images, indicating that the leftover images are dissimilar to the images within the bins. This UI was used in the experiments described below.

3.3.1 Crowd Experiments with Ground Truth

As a sanity check, we executed two small scale experiments, with a small crowd (about 10-15 crowd members) and small sets of images, where the ground truth is known. We experimented with two different computation tasks: top-k and clustering. For each task, the crowd members answered queries of both the baseline algorithm and our algorithm.

Top-k similar colors. The simplest set of images that we have used is a set of 300 solid colors, whose ground truth similarity can be measured, e.g., by embedding the colors into 3-dimensional space according to their RGB or HSL values (we have used RGB). The goal was to compute, for each color, the k-NN most similar colors for varying values of k. We have compared the results of our algorithm to the results of the baseline random and crowdcluster algorithms, using the same number of queries overall in the three algorithms.

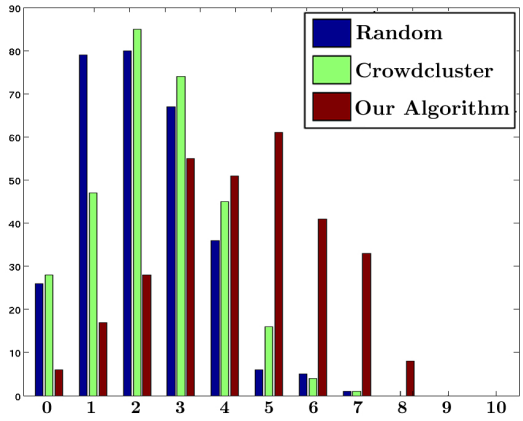


Figure 3.3: *A comparison of the accuracy of 10-NN images based on real crowd input, using our algorithm and two baseline alternatives.*

The results indicate that our algorithm identifies a larger percentage of the nearest neighbors for a larger percent of the images. Figure 3.3 illustrates the 10-NN results for the three algorithms using 235 queries overall. Five phases were used in our algorithm. For each algorithm, we show a histogram of intersection between the true 10-NN (according to the ground truth) and the computed 10-NN. Note that crowdcluster slightly outperforms the random baseline, but our algorithm generally identifies a larger fraction of the true 10-NN images, "pushing" the histogram rightwards (red bars). Overall, our algorithm identifies 43.4%-50% more of the true nearest neighbors than the baseline alternatives, which demonstrates the effectiveness of our progressive refinement approach.

Clustering fonts. In this experiment we have tested the ability of our algorithm to cluster letter images into fonts, where the ground truth is the font to which the letters belong. We have used 180 letters of 12 different fonts, and asked crowd members to evaluate the similarity of letters with respect to their appearance. The results have been used to compute 12 letter clusters, which should ideally match exactly the 12 original fonts. Our algorithm has used 123 queries in total over 5 refinement phases. For comparison, we have executed the same task with 123 random queries.

Figure 3.4 illustrates the experimental results and in particular the progressive refinement, via heatmaps that represent the cluster quality after each of the 5 phases. The results of the algorithm are almost perfect, with only 1.1% errors (two letters). In comparison, the random query selection resulted in around 60% errors, and was outperformed by our algorithm already after the second

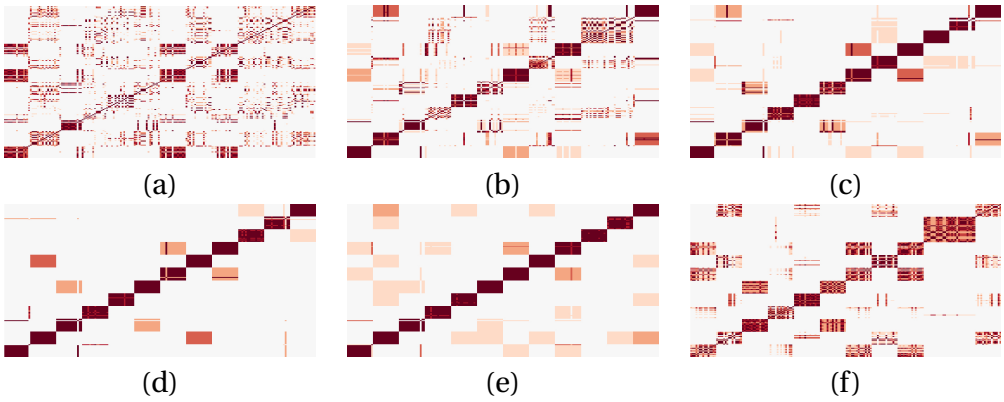


Figure 3.4: *Heatmaps displaying the accuracy of clustering for the font dataset. Figures (a)-(e) illustrate the cluster quality after phases 1-5 of our algorithm, respectively, and 123 queries in total. For comparison, Figure (f) displays the cluster quality after 123 random queries.*

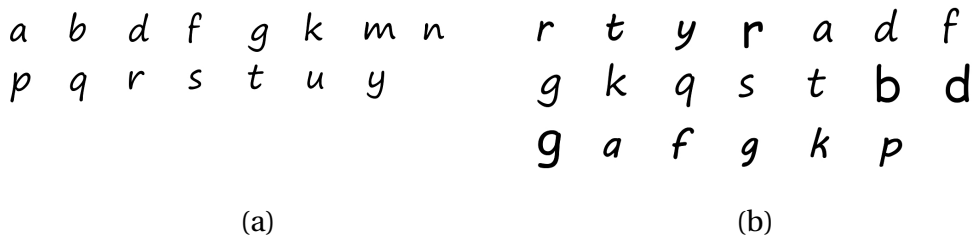


Figure 3.5: *Two examples for clusters produced for the same letter "a" (on the top left), based on the similarity metric of (a) our algorithm, and (b) random baseline.*

phase. Figure 3.5 displays an example cluster produced by our algorithm, and the corresponding cluster produced by the random baseline. The latter cluster makes sense in the broader context of the fonts, since it contains only handwriting fonts; but the progressive refinement in our method allows distinguishing also between the different handwriting fonts.

3.3.2 Crowd Experiments with Real-world datasets

Next, we have executed experiments with two real-world datasets where the image similarity is highly semantic and therefore image features may not be sufficient for estimating this similarity. The first dataset consists of 910 images of movie posters downloaded from the movie pages in Wikipedia, where similarity

Dataset	Number of images	Success %	Δ
Movie posters	910	87.2%	2.5
Chairs	1024	76.2%	3

Table 3.1: *Real-world dataset results*

is usually based on genre, style of the poster, characters, and so on. For this set we have collected 547 query answers from about 60 crowd members.

The second dataset consists of 1024 chairs, of different types and angles from the ShapeNet dataset [Chang *et al.*, 2015]. Similarity in this dataset is based, among others, on semantic features such as the usage of the chairs, the material they are likely to be made of, and their assessed level of comfort. For this set we have collected 559 query answers from about 60 crowd members.

As in many real-life scenarios, for these sets there is no ground truth or gold-standard. Hence, we have manually examined the results of our algorithm by sampling images with with their k-NN images, and comparing these results with the results obtained by automatic means based on image features. For the movie dataset, we used a color histogram with 64 bins (four bins for each of the RGB channels), and an image thumbnail of four by four pixels, or a total of 16 RGB values. The two descriptors were concatenated and treated as a single vector for the distance calculation. For the chair dataset we have used features derived from HoG descriptor [Dalal and Triggs, 2005].

For the manual examination, we used 50 random "seed" images sampled from each of the datasets. For each seed image, we took its 10 NN images from the dataset according to both our algorithm and the feature-based baseline. Each of the images was labeled "very similar", "similar", or "unrelated" with respect to its seed image. We counted the percent of seed images for which our algorithm finds a greater number of similar images than the baseline, breaking ties by the number of "very similar" images. The results are displayed in the **Success %** column of Table 3.1. To quantify by how much we outperform the baseline, we also computed the average difference between the number of similar images our algorithm has discovered and the baseline. This difference is marked by the Δ column in the table.

We illustrate a specific example of the observed difference in Figure 3.1. The figure displays the 10-NN images (a) computed by our algorithm based

(a) (b)

Figure 3.6: *Image retrieval results: nearest neighbors of the center image in a collection of chairs, computed using (a) HoG descriptor, and (b) crowdsourced queries. Smaller images mark farther neighbors. Less similar chairs are highlighted.*

on clustering queries and (b) according to color descriptors. The seed image is displayed in the middle. In this case, the results of our semantic similarity estimation retrieve movies of the same genre (animated adventure films). Within that genre, most of the closest neighbors (four out of the top five) have the same visual appearance (blue background) as the seed image. On the other hand, the movies retrieved by using image descriptors have a similar visual appearance in terms of color scheme and mood but are very different semantically. Note that while we use rather simple image descriptors, even extremely sophisticated descriptors would fail to associate posters of movies in the genre which has different visual appearance with the seed image.

Figure 3.6 displays similar results for the chair dataset, but where the baseline k-NN results (a) are computed according to HoG descriptor. The seed chair is a school chair with curvy tubes supporting the back. The 10-NN chairs given by our algorithm are all school chairs and many of them contain similar style elements such as curvy tubes. In contrast, the chairs computed using the HoG descriptor seem superficially similar (and also have the same orientation) yet include office and dinning room chairs, and vary more in their style (the less similar chairs are highlighted in the figure).

Figure 3.7 displays a few more selections of k-NN results for movie posters and chairs. In each set the top left image is the seed and its 7 nearest neighbors are presented from left to right. In many cases, the similarity between images can be both semantic *and* visual. We have deliberately selected cases which present a purely semantic relation which may be very hard or impossible to capture

Figure 3.7: *Image retrieval results: K-NNs of images from the movie posters and chairs datasets.*

using image descriptors. The semantic connection between movie posters vary greatly, and spans movies from the same genre (a), posters that have dominant typographic elements (b), posters of old movies (c), or the same expression of the faces in the poster (d). The semantic connection between chairs may be similar style elements (e), similar overall shape (f), similar function (g) or even chairs with wheels (h). The k-NN results for all movie posters and chairs in the dataset can be found on the project's website.

3.3.3 Synthetic Experiments

We next provide further analysis of our algorithm via synthetic experimental results. The experiments were conducted on datasets with available ground truth, and with answers from a simulated crowd. The simulated answer for a given query was computed using a k-means algorithm, which has split the 20

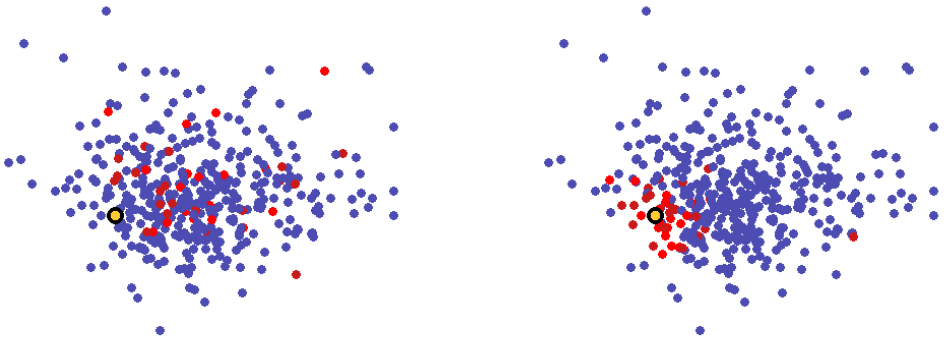
images in the query into 4 clusters. Using synthetic answers allows us to test the performance of our algorithm in a variety of scenarios.

Effect of locality. In the Introduction, we have stressed the importance of using queries about local neighborhoods of images. To test this claim in isolation, we have conducted a dedicated synthetic experiment, as follows. We have used a set of 1000 colors sampled uniformly. Since the true similarities are known for this image set, we could vary the locality of queries: for each query we started from a seed image, then sampled the rest of the images from within a certain distance from the seed image. We have then used the results of the queries to compute the embedding as usual. We have observed an almost linear decrease in the average precision of the computed 10-NN images as the distance between images in each sample increases.

Co-occurrence of similar images. One of the indications for the effectiveness of the progressive refinement in our algorithm is the frequent co-occurrence of similar images in the same query. Ideally, as the similarity metric that we compute converges to the true one, similar images are more likely to appear in a query together. Moreover, the distance between pairs that appeared together in many queries is expected to be more accurate, since more data is available. Since the budget of queries is limited, each pair that is queried comes at a cost of another pair for which there will be less available information. We show that our algorithm effectively favors pairs which are close to each other and therefore need more accurate information.

Figure 3.8 illustrates this. We simulate a two dimensional embedding of images, where each point represents an image in the dataset. The distance between each pair of points (or images) is taken from the embedding, which simulates ground truth similarity. The dataset contains 400 images, and we ran 400 simulated queries, once using our algorithm and once with random queries. We then select an arbitrary image (marked in gold) and count how many times each image in the dataset has co-occurred with it. We rank the images according to their mutual queries count. The top 20 images (5% of the dataset) that were queried together the most with the golden image are colored bright red. The next 20 images (5%) are colored dark red. The rest of the images (360 or 90%) are colored light blue.

Figure 3.8(a) shows that using random query selection, the images that co-occurred the most with the golden image are randomly scattered, as expected. In contrast, using our algorithm to select the queries (Figure 3.8(b)), the frequently



(a) (b)

Figure 3.8: *Visualization of the images that appeared in the same query as the image marked in gold. The images are ranked by the number of mutual queries and the top* 10% *images are colored red. (a) Mutual queries after* 400 *random queries. (b) Mutual queries after* 400 *queries using our algorithm.*

co-occurring images are centered around the golden image. Evidently, we do not spend queries on pairs which are known to be far away, since their distance from each other matters less and is expected to be less accurate. This allows our algorithm to better estimate the relative local similarities, and use them to estimate the global similarities.

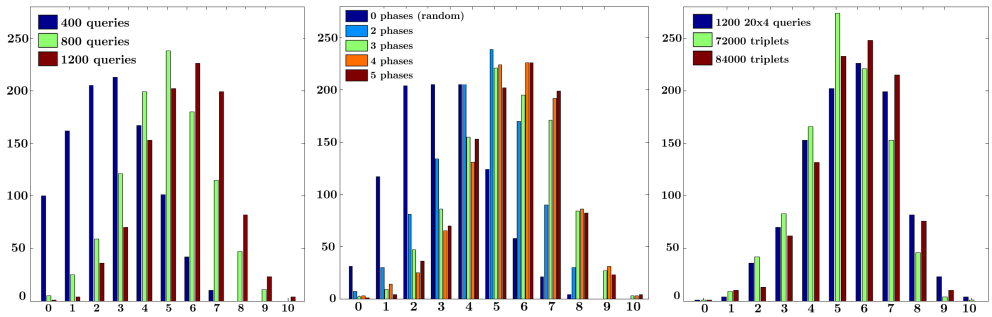


Figure 3.9: *Number of correct 10-NN images as a function of number of queries (left) and number of phases (center), and versus a triplet-based algorithm (right).*

Varying the algorithm parameters. We next execute our algorithm while varying the value of two parameters: the total number of queries and the number of phases, to demonstrate the impact of these parameters on the query results. Figure 3.9(right) illustrates the effect of varying the total number of queries, for a

synthetic 1000 random color dataset, and 5 phases of our algorithm. As expected, there is a positive correlation between the number of queries we use and the quality of the results, measured by the size of the intersection between the true 10-NN images and the 10-NN images that we compute. This means that with a greater budget we can improve the estimation of the similarity metric.

Figure 3.9(center) illustrates the impact of number of phases on the quality of the results (using the same image set as above, the same quality metric, and 1200 queries overall). The number of phases ranges from 0 (which is equivalent to random query selection) to 5. Note that increasing the number of phases increases the result quality, since recomputing the embedding more frequently allows creating better queries. However, the margin by which the quality improves decreases, so the difference between 4 and 5 phases is small.

Queries versus triplets. A common solution for collecting image comparisons from the crowd is based on triplet queries of the form "Is image A more similar to image B or to image C?". We have already noted that one advantage of our approach over the triplet-based one is that clustering queries provide context for comparison. In this synthetic experiment we ignore the context, and focus on the number of questions needed for each type of solution. As shown in Figure 3.9(right), our algorithm's performance using 1200 queries is comparable to the triplet-based algorithm's performance using 84000 queries.

3.4 Conclusion

In this chapter, we presented an efficient approach for estimating the similarity of images based solely on the input of the crowd. Our system progressively refines the images posed to the crowd, in order to obtain similarity comparisons between images in the same neighborhood, allowing faster convergence to an accurate similarity metric. In our experimental study we have used a particularly small number of queries, and have shown that even on this basis we can obtain a fair estimate of the semantic similarity. Our method can also be used to estimate the similarity of any collection of objects that can be represented by images, such as 3D shapes.

Limitations and future work. This work focuses on input from the crowd alone. However, it is often the case that some clues for the semantic similarity of images are available in the form of image features or textual context. Even if

these clues do not account for the full range of semantic connections, it would be interesting to examine how to leverage them in conjunction with our algorithm. This direction may benefit the method's scalability, since in very large image sets, the affordable number of queries might not even be linear in the size of the set. A straightforward approach for integrating semantic clues would use our algorithm to learn similarities for a small fragment of the image set, and then apply machine learning techniques to complete the rest, using features based on semantic clues (in the spirit of [Lun *et al.*, 2015; Saleh *et al.*, 2015; Yi *et al.*, 2012]). A more interesting solution may further combine the clues within the query generation phases. This is non-trivial, since the usage of other estimates can potentially cause semantically similar images to be overlooked.

Another challenging direction for future work includes a more elaborate treatment of the uncertainty stemming from the crowd. Crowd members often disagree on the similarity of images, or provide inconsistent answers. So far, we have assumed that the embedding we perform mitigates the impact of such inconsistencies. However, we may want to explicitly account for inconsistencies, by a probabilistic modeling of the crowd's behavior, e.g., as done in [Gomes *et al.*, 2011] for the purpose of clustering. It would thus be interesting to develop probabilistic models dedicated for the learning of a similarity metric. In particular, this method should support efficient computations, due to the interactive nature of our algorithm.

4 SHED: Shape Edit Distance

One of the primary goals of shape analysis is understanding what type of object is represented by the shape. In Section 1.1 We referred to this as the *basic semantic similarity* between shapes. Naturally, the development and evaluation of similarity measures in the 3D shapes domain is typically geared towards classification of shapes into broad sets of categories [Tangelder and Veltkamp, 2008]. Detection of *inter-class* differences has been emphasized over quantification of *intra-class* differences, and little attention has been given to estimating the similarity between shapes that belong to the same class. For example, in the context of shape retrieval, the success of a method is often evaluated based solely on the number of shapes that are retrieved from the same class. The question whether the retrieved shapes are the most similar within the class remains unanswered. However, with the large repositories available today, organization and exploration of shapes from the same class have become as important as categorizing shapes into different classes. These tasks require an estimation of fine-grained shape similarities, including similarities in function, style, and the part composition of a shape.

We aim to improve on existing methods by identifying both inter-class and intra-class similarities. Our premise is that the semantic similarity humans perceive between shapes is very much related to the part composition of each shape and the similarity between each part and its counterpart. Thus, we introduce *shape edit distance* to measure similarities between shapes. Intuitively, the shape edit distance (SHED) measures the amount of effort needed to transform one shape into the other, in terms of rearranging the parts of one shape so that they closely match the parts of the other shape, or by adding and deleting parts (Figure 4.1). SHED takes into account both the similarity of shape structure and the similarity of individual shape parts. We follow a recent trend of representing shapes as graphs of parts [Kalogerakis *et al.*, 2012; Laga *et al.*, 2013;

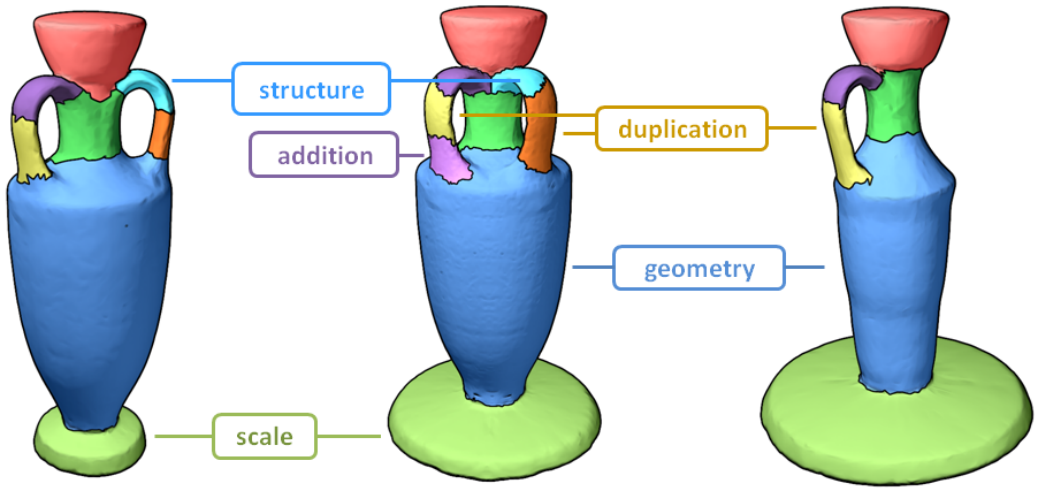


Figure 4.1: *Shape edit distance: the distance between shapes is measured by edit operations that transform the parts of one shape into corresponding parts in the other shape.*

Mitra *et al.*, 2013; Zheng *et al.*, 2014]; however, we use the matching between graphs to extract a *global measure* of shape similarity.

The strength of the shape edit distance is its tolerance to part re-arrangements, additions and deletions. Thus, SHED is flexible in quantifying the similarity between shapes that have partial similarities, articulated parts or repositioned parts. This leads to a similarity measure that accurately captures finer shape differences, enabling a finer-grade organization of shapes. In contrast, other traditional similarity measures are oblivious to the shape structure: for example, the light field descriptor, popular in shape retrieval [Chen *et al.*, 2003], is highly sensitive to any type of shape difference or deformation, while in bag-of-feature approaches, the similarity is invariant to the arrangement of shape components [Bronstein *et al.*, 2011; Litman *et al.*, 2014].

We do not explicitly find a sequence of editing operations that transforms one shape into the other. Instead, we indirectly estimate the edit distance by using a part correspondence to extract a measure of similarity. First, shapes are segmented into parts, and an approximate correspondence is computed between the parts of each shape. We do not enforce a strict one-to-one correspondence, since a part in one shape may be duplicated or missing from the second shape. Instead, we apply constraints to the matching by associating additional costs when parts change their context. Then, each match between two parts is associated with a transformation cost: a weighted sum of terms that relate to

the differences in part geometry, scaling and position of the parts in the shape. Finally, the edit distance of the shape is the aggregated cost of transforming all parts in the correspondence. We use supervised learning for automatic computation of the weights from examples, as opposed to manual tuning, which could be unintuitive.

We demonstrate the advantage of using SHED with a series of experiments. First, we evaluate SHED in a quantitative manner by constructing categorization trees that can be used for shape exploration. We compare these trees to the trees generated using other state-of-the-art similarity measures, as well as ground truth trees created by expert users. In addition, we cluster shapes into a predefined number of clusters and compare the results to clusters generated from the ground truth trees. These evaluations demonstrate that the similarity estimated by SHED is preferable to other distance measures and leads to a more intuitive shape organization in the intra-class context. In the inter-class context, we perform shape retrieval according to SHED and show that it yields comparable results to state-of-the-art similarity measures. Finally, in settings where ground truth data is not well defined, we show qualitative results of nearest neighbors queries and embeddings of sets of shapes.

4.1 Related Work

This work comprises ideas such as shape comparison, graph edit distances and part-based matching, which we discuss as follows.

Shape comparison, retrieval and exploration. There has been much work on the development of shape similarity measures that can be used for retrieval, exploration, or any type of shape comparison [Tangelder and Veltkamp, 2008]. In terms of shape retrieval and categorization, state-of-the-art approaches that currently give the best performance are a combination of the light field descriptor with bag-of-features and metric learning approaches [Li *et al.*, 2012a]. For intra-class organization, Xu et al. [Xu *et al.*, 2010] cluster a set of shapes into different groups by factoring out the effect of non-homogeneous part scaling and then establishing a correspondence between shape parts. Huang et al. [Huang *et al.*, 2013a] present an approach for fine-grained labeling of shape collections. Similarly to our work, their goal is to learn a distance metric within a class of shapes to capture finer shape differences. However,

their method follows a different paradigm than our work: the shapes are globally aligned with an affine transformation followed by local deformations, and the metric is learned on the aligned shapes. Individual parts obtained from segmentation and their transformation are not considered as in our approach. In the more restricted context of isometric matching, there has been much activity in deriving signatures for shape comparison, such as GPS embedding [Rustamov, 2007] or heat kernel signature [Ovsjanikov *et al.*, 2010]. Kurtek et al. [Kurtek *et al.*, 2013] define a shape space and metric that capture more comprehensive deformations than nearly isometric, but require surfaces of the same topology. Bag-of-feature approaches [Bronstein *et al.*, 2011; Litman *et al.*, 2014] are considered state of the art for retrieval of non-rigid isometric shapes. The goal of these methods is to retrieve shapes with similar topology from a collection of shapes in the same class, such as human models in different poses. Hence, these methods are not suitable for comparison of shapes with different part composition, structure or topology, which is the focus of our work.

Shape exploration necessitates not only the estimation of the similarity of shapes to a query shape, but also a way of organizing the shapes. Thus, different strategies have been proposed for exploration, such as the use of a deformable template [Ovsjanikov *et al.*, 2011], region selection [Kim *et al.*, 2012], dynamically adapted views of close neighborhoods [Kleiman *et al.*, 2013], or parameterization of the template space [Averkiou *et al.*, 2014]. In the work of Huang et al. [Huang *et al.*, 2013b], the goal is to obtain a qualitative organization of a collection of shapes, since an organization based on a single similarity measure is not always meaningful when comparing both similar and dissimilar shapes. Likewise, our goal is to properly capture both inter- and intra-class differences. However, instead of aggregating the scores of several similarity measures, we develop an edit distance to estimate the shape similarity.

Graphs of parts for shape analysis. The idea of describing 2D shapes and images as graphs of parts has appeared prominently in the field of computer vision. A few representative works include matching shapes according to *shock graphs* [Sebastian *et al.*, 2004] and skeletons [Sundar *et al.*, 2003], and matching images according to graphs that represent their segmentations [Harchaoui and Bach, 2007]. In the graphics literature, comparing shapes by matching graphs was utilized for consistent joint segmentation [Huang *et al.*, 2011] and

co-segmentation [Sidi *et al.*, 2011] of a set of shapes. A group of works has estimated the similarity between shapes by matching Reeb graphs, which are constructed from functions defined on manifold shapes [Hilaga *et al.*, 2001; Barra and Biasotti, 2013]. Other works have explicitly segmented shapes and created graphs of segments, with applications in shape synthesis [Kalogerakis *et al.*, 2012] and semantic correspondence [Laga *et al.*, 2013]. These works are directly related to the idea of modeling shapes by combining parts from different models [Funkhouser *et al.*, 2004]. Templates or part arrangements have also been learned from collections, although these do not explicitly represent the connectivity between parts [Kim *et al.*, 2013; Zheng *et al.*, 2014]. The fundamental difference of our approach to these representative works is that we use the matching between two graphs of parts as input to estimate the overall similarity between two shapes; the correspondence between the graphs is the base for a distance measure that enables us to quantify finer shape differences.

Graph matching and integer programming. The graph matching problem is commonly posed as an integer quadratic programming problem, which is NP-hard. There is a large body of work regarding the relaxation of such problems to a tractable convex quadratic programming optimization. Two prominent works in this area are the spectral correspondence presented by Leordeanu and Hebert [Leordeanu and Hebert, 2005] and a relaxation of the quadratic optimization by using bounding linear integer optimizations, proposed by Berg et al. [Berg *et al.*, 2005]. These relaxations often yield good results in practice in the one-to-one matching scenario. However, performing gradient descent from a continuous relaxation of the integer problem has been shown to yield non-optimal permutations in most cases [Lyzinski *et al.*, 2015]. Indeed, the above methods perform poorly in our one-to-many scenario where a part can correspond to several parts in the other shape. Recently, Kezurer et al. [Kezurer *et al.*, 2015] suggested lifting the problem to a higher dimension, followed by a linear semi-definite relaxation. However, their method is computationally expensive and does not extend easily to one-to-many scenarios. Bommes et al. [Bommes *et al.*, 2012] perform iterative relaxation of the problem where in each iteration a single integer constraint is added to the optimization. We follow a similar approach, but instead of adding hard constraints in each step, we adjust the objective function to give precedence to solutions which are compatible with previously selected matches.

Graph edit distance. The graph edit distance has been used to find a correspondence between graphs in several areas of visual computing, such as computer vision and medical imaging [Gao *et al.*, 2010]. The idea of an edit distance is attractive because it poses the problem of matching two graphs as finding a sequence of operations that transforms one graph into the other. The edit distance can consider not only the matching of similar nodes and edges, but also their addition, duplication and deletion. However, finding the minimal edit distance is NP-hard, so different heuristics have been proposed to compute it. A common technique is to use a graph kernel that estimates the similarity between two nodes according to their attributes and their neighborhoods in the graphs [Neuhaus and Bunke, 2007]. Our shape edit distance does not require an explicit sequence of operations, but an aggregation of all the changes necessary to transform one shape into the other.

In the context of computer graphics, Fisher et al. [Fisher *et al.*, 2011] used graph kernels to estimate the similarity between graphs representing scenes composed of multiple objects. In addition, Denning and Pellacini [Denning and Pellacini, 2013] proposed a technique based on the edit distance to quantify *localized* differences between two models. Their method is better suited for comparing models generated by editing the same source shape. On the other hand, our work is aimed at computing the similarity between any pair of shapes. We derive the edit distance directly from a correspondence between graph nodes, as opposed to the methods above based on graph kernels. In addition, we do not require a one-to-one correspondence between the shape parts, but find a one-to-many correspondence and quantify the edit distance without explicitly searching for a sequence of editing operations. We explain the details in the next section.

4.2 Shape Edit Distance

Input, output, and shape representation. The edit distance measure takes as input two shapes and returns a real number representing the distance (dissimilarity) between the shapes. The distance is lower for shapes that have similar part geometry and structure, taking into account part rearrangements and partial correspondence, and higher for shapes that differ in these aspects.

We represent each shape as a collection of parts and connections among these parts, i.e., a graph of parts. Our method is generic and can take as input different

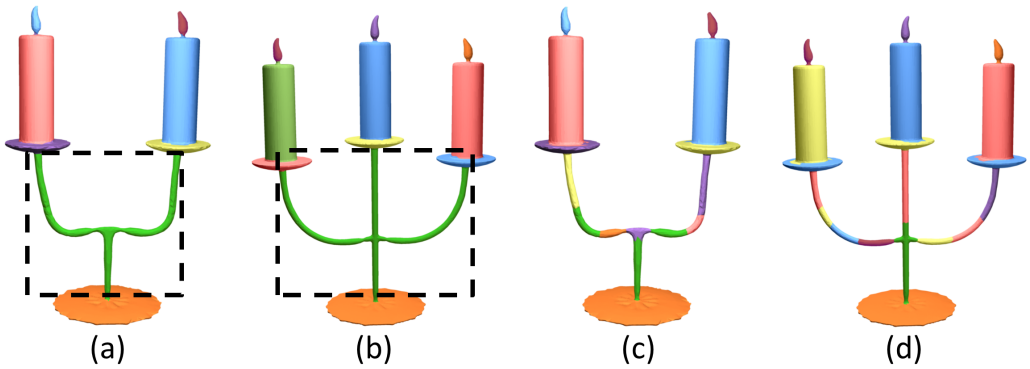


Figure 4.2: *Difference between the semantic segmentation of two shapes in (a) and (b), and their nearly convex decomposition in (c) and (d). Note how, in (a) and (b), the bounding boxes of the parts corresponding to the candle supports have considerably different sizes. In (c) and (d), both supports are composed of small nearly convex segments with similar sizes.*

shape representations, although in this work we represent the shapes as triangle meshes. The first step in our method is the partitioning of input meshes into parts. One possibility is to use semantic segmentation techniques [Shapira *et al.*, 2008; Shamir, 2008]. However, semantic parts do not have a clear definition and can greatly vary among different shapes. Moreover, a semantic part can have a complex geometry, making its comparison to other semantic parts non-trivial (Figure 4.2). In a sense, the problem of comparing two complex segments can be as involved as that of comparing two shapes. Instead, we segment the shapes into simpler primitives that can be more easily analyzed. For this task, we use the recent weakly-convex decomposition technique of van Kaick et al. [van Kaick *et al.*, 2014], which partitions the input shapes into nearly convex parts. Nearly convex parts are easier to analyze, since they have a simpler geometry and can be approximated well by their bounding boxes (Figure 4.2). In addition, the convex decomposition of a shape is robust to small changes in the shape.

Our method also supports using a manual segmentation of the shapes into parts, if such data is available. However, the results in this paper were produced using the automatic weakly-convex decomposition to provide a complete solution. The part graph is defined by creating a node for each nearly convex part of the shape, and an edge between adjacent parts in the shape segmentation.

Part similarities and matching. Given two shapes represented as graphs of parts, our goal is to find a set of editing operations that transform the parts of one shape into the parts of the other. Possible editing operations include deforming, displacing, duplicating, adding, or removing parts. Then, a cost is associated with each editing operation based on the extent of the transformation. The editing costs are aggregated to produce the final shape edit distance between the two input shapes.

In SHED, we derive the set of editing operations from a mapping between the parts, since we can associate each pairwise match with a single operation. This mapping depends on the similarity of parts to each other as well as their context and the structure of the shape. For example, two parts with different geometry can be matched if their neighborhood is similar. On the other hand, two parts in different locations in the shape can be matched if their geometry is similar. Thus, the mapping of each part depends not only on the part properties, but on the mapping of all other parts of the shape. This makes the problem of finding the correct matching intractable, so an approximate solution is necessary. To this end, we formulate our objective in a quadratic form by constructing unary terms for each match between two parts, and binary terms for pairs of matches, representing only pairwise dependencies between matches. Then, we develop a novel adaptive spectral matching technique to find an approximate solution for this formulation. Our technique uses similar principles as the method of Leordeanu and Hebert [Leordeanu and Hebert, 2005], but instead of solving the optimization once and applying constraints in a greedy manner, we iteratively improve the optimization by incorporating the constraints that arise in previous steps. We explain the computation of the matching in detail in Section 4.3.

Given the mapping between two shapes, a cost can be computed for each edit operation. The costs reflect the following aspects of shape similarity:

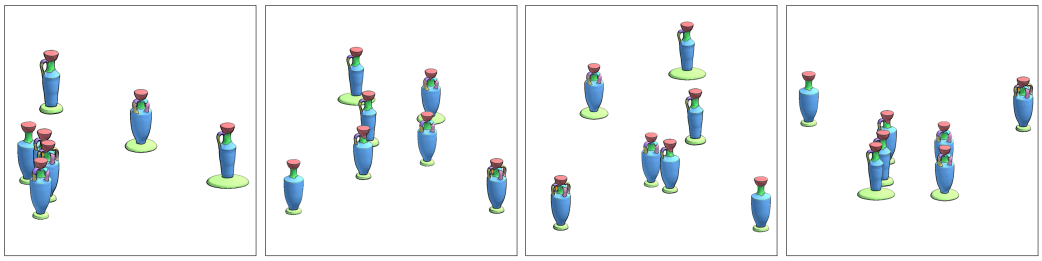
- **Similarity of the geometry of the parts.** For example, morphing a cylindrical part into another cylindrical part is less costly than morphing a cube into a cylinder, as the former pair is geometrically more similar than the latter.

- **Similarity of the structure of the part graphs.** We allow nodes to move in the part graph, with a cost proportional to the magnitude of the structural change. Duplicated parts and additional parts also incur additional costs as the structure of the shape changes.

- **Scaling of the parts.** The scale of each part plays a critical role in the global similarity of a shape; different shapes can have similar graphs of parts where each part is scaled differently relative to its neighborhood. Thus, we introduce scale-specific terms in our formulation.

To produce a scalar similarity measure between two shapes, the terms described above need to be weighted and aggregated. A question arises of how to determine the weights for each term. Shape similarity is a subjective measure, so different users might have different views on which shapes are more similar, which implies that different weights are necessary. Moreover, while a set of manually selected weights can provide a reasonable similarity measure for all shapes, it is clearly beneficial to fine-tune the weights to better reflect the variation in a specific set. Therefore, we employ a weight learning scheme that finds the optimal weights to match a set of given distances. We elaborate on the details of the distance formulation and the weight learning scheme in Section 4.4.

In Figure 4.3, we show the effect of considering these different factors in the edit distance. We compare a 2D embedding created with multi-dimensional scaling, according to the similarities given by SHED and the light field descriptor (LFD). For SHED, we show the results of using three configurations of weights: equal weights for each term (b), weights learned from user input giving high priority to the scaling of parts (c), and weights learned from user input giving high priority to the structural difference between parts (d). The example set contains vases ranging from zero to four handles, some with a slightly thinner body and some with a bigger base. The consistency of distances provided by SHED yields an intuitive embedding that is true to the observed properties of



(a) (b) (c) (d)

Figure 4.3: *Embeddings obtained with multi-dimensional scaling on a small set of vases, based on the following distance measures: (a) LFD, (b) SHED with default weights, (c) SHED with high weight for scaling changes, and (d) SHED with high weight for structural changes.*

the shapes, namely the number of handles and size of the parts. On the other hand, the embedding generated by LFD groups shapes according to their overall appearance, and does not take into account the finer details of the shapes. Thus, LFD is not able to distinguish well between the vases that differ by the number of handles, as their projected views are very similar.

4.3 Part Matching

The correspondence between two shapes can be represented as a list of *matches* or pairings between two parts, one from each shape. The mapping does not have to be one-to-one; a part in one shape can be duplicated and have several matches in the other shape. However, we constrain the mappings so that if a part is duplicated, then its matching parts in the other shape are not duplicated, to ensure consistency in the editing operations. In other words, for each edge in the matching graph, the degree of at least one of its vertices is one. The dependencies between different possible matches are complex and can involve more than two matches. We approximate such dependencies by using pairwise constraints only, so the problem becomes tractable. We formulate the correspondence problem using unary terms that depend on a single match, and binary terms involving a pair of matches. Unary terms represent the likelihood of a match, or the affinity between a part in one shape and a part in the other shape. Binary terms represent the compatibility of two matches, i.e. the likelihood that both matches will be a part of the same mapping.

Unary term. The unary term represents the amount of effort necessary to morph the geometry of a part into another part. One of the advantages of segmenting the shape into nearly convex parts is the simplicity of each part, which allows us to use efficient descriptors to effectively distinguish between part geometries. We use the shape distribution signatures to represent the geometry of the parts [Osada *et al.*, 2002]. Specifically, we use the D1 descriptor (also called *shell histogram* [Ankerst *et al.*, 1999]), which computes a histogram of the distance between uniformly sampled points on the surface and the center of mass of the part, and the D2 descriptor, which computes a histogram of the distance between pairs of uniformly sampled points on the surface. These descriptors are relatively simple and fast to compute, yet they are able to distinguish well between parts with simple geometry such as nearly convex parts. The D1 and D2

histograms are computed for each part, and compared using χ^2 distance, which is defined as

$$d_{\chi^2}(H_i, H_j) = \sum_{k=1}^{K} \frac{(H_i(k) - H_j(k))^2}{H_i(k) + H_j(k)}, \tag{4.1}$$

where H_i, H_j are the input histograms, and K is the number of bins in each histogram. The geometry cost is thus

$$C(i,j) = \alpha \cdot d_{\chi^2}(D1_i, D1_j) + (1-\alpha) \cdot d_{\chi^2}(D2_i, D2_j) \tag{4.2}$$

where $D1_i$ and $D2_i$ are respectively the D1 and D2 histograms for part i, and α controls the balance between the D1 and D2 descriptors. In our implementation $\alpha = 0.5$ (equal weights). The cost is transformed into an affinity using the natural exponent:

$$U(i,j) = exp(-C(i,j)/\sigma), \tag{4.3}$$

where σ is chosen such that the affinity values have a wide spread between 0 and 1. In our implementation $\sigma = 0.5$.

Binary term. The binary term represents the compatibility of one match (i,j) to another match (k,l). When two shapes are similar, adjacent parts in one shape are expected to be mapped to adjacent parts in the other shape. In addition, the scaling factor of all matches is expected to be similar, since a match that has significantly different scale than other matches in the mapping is less likely to be correct. Therefore, we define a graph distance cost and a scaling factor cost for each possible match.

The graph distance is defined for each pair of parts on the same shape as the length of the shortest path between these parts in the shape graph. We use the ratio between the graph distances of each match to measure the compatibility between matches:

$$G(i,j,k,l) = \frac{\max(g(i,k), g(j,l))}{\min(g(i,k), g(j,l))} - 1, \tag{4.4}$$

where $g(i,k)$ is the graph distance between parts i and k on the same shape. This term is zero when both pairs of parts have the same graph distance in their respective shape, and is highest when one pair of parts is adjacent and the other is not. Note that the cost is low when the graph distances between both pairs are high, so adjacent parts have more weight in the total cost.

We define the scaling factor of each match as the ratio between the volumes of the source and target part: $s(i,j) = \frac{VOL(i)}{VOL(j)}$. Similarly to the graph distance cost, we use the ratio between the scaling factors of two matches as the scaling factor cost:

$$S(i,j,k,l) = \frac{\max(s(i,j), s(k,l))}{\min(s(i,j), s(k,l))} - 1. \tag{4.5}$$

The binary term is defined as the affinity between two matches, which is computed from the above costs as follows:

$$B(i,j,k,l) = \beta \cdot \exp(-(G(i,j,k,l) + S(i,j,k,l))/2). \tag{4.6}$$

The parameter β controls the weight of the binary term compared to the unary term. If β is large, the structure of the shape takes precedence over the geometry of parts, and if β is small, the geometry of the parts is more important than the shape structure. If $\beta = 0$, the only consideration is the part geometry and the correspondence resembles a bag-of-features approach. In our implementation, $\beta = 0.3$.

Matching technique. There are several matching techniques in the literature that find an approximate solution to pairwise-constrained correspondence problems, such as the spectral matching technique of Leordeanu and Hebert [Leordeanu and Hebert, 2005], or the integer quadratic programming relaxation proposed by Berg et al. [Berg *et al.*, 2005]. The main idea of these methods is that the pairwise constraints can be presented in a quadratic form by constructing a matrix M of $n \cdot m$ rows and $n \cdot m$ columns, where n and m are the numbers of parts in the first and second shape, respectively. The diagonal of M contains the values of the unary term $U(i,j)$, and the values outside of the diagonal of M are the binary terms $B(i,j,k,l)$. The best correspondence is then represented by the binary vector x that maximizes the product $x^T M x$ and does not break additional constraints, such as the requirement for one-to-one mapping, etc. This poses an integer quadratic programming problem, which is NP-hard, therefore different approximation methods are suggested in the above methods.

Leordeanu and Hebert [Leordeanu and Hebert, 2005] propose to first solve an un-constrained assignment problem in the continuous setting, where x is allowed to have values in the range $[0,1]$. This can be solved easily by setting x to the normalized principal eigenvector of M. Then, the result vector x is

binarized in a greedy manner, taking into consideration additional constraints in the process. In each step, the match with the highest value in x is marked, and the values of the match and all conflicting matches in x are reduced to zero. This process continues until all values in x are zeros, and the final mapping is returned as the collection of marked matches. Since the constraints are not incorporated into the cost matrix, the greedy binarization process is less successful when several conflicting mappings are possible. While strictly conflicting matches are filtered out, matches which are compatible with those conflicting matches might still be selected since their score is computed before the conflicting matches are discarded. This effect is most prominent in less constrained scenarios such as ours. For example, we allow duplications of parts, but a matching in which almost all parts are matched to the same part is valid but not desirable in most cases.

To address these issues, we introduce *adaptive spectral matching*, which incorporates the desired constraints directly into the objective function, leading to a more consistent global solution to the correspondence problem. We iteratively adjust the affinity matrix M according to the constraints and re-run the eigenvector decomposition. In this way, not only conflicting matches are excluded from the solution, but matches that are compatible with conflicting matches are also less likely to be selected in subsequent steps. The iterative method starts by setting x to the principal eigenvector of M, and then performs the following steps:

- Mark the match with the highest value in x.
- Set the affinity of the match in M to 1.
- Incorporate constraints into M, by setting the affinities of each conflicting match or pair of matches to zero. In our case, once a match (i, j) is selected, the compatibility of matches that contain part i to matches that contain part j becomes zero (i.e. the binary scores $B(i, j', i', j) = 0$ for each i' and j'), since having both of these matches would mean that there is a many-to-many relation between parts i and j.
- Set x to the principal eigenvector of the adjusted M, and ignore all matches that are conflicting or were already selected.
- Repeat until there are no more valid matches.

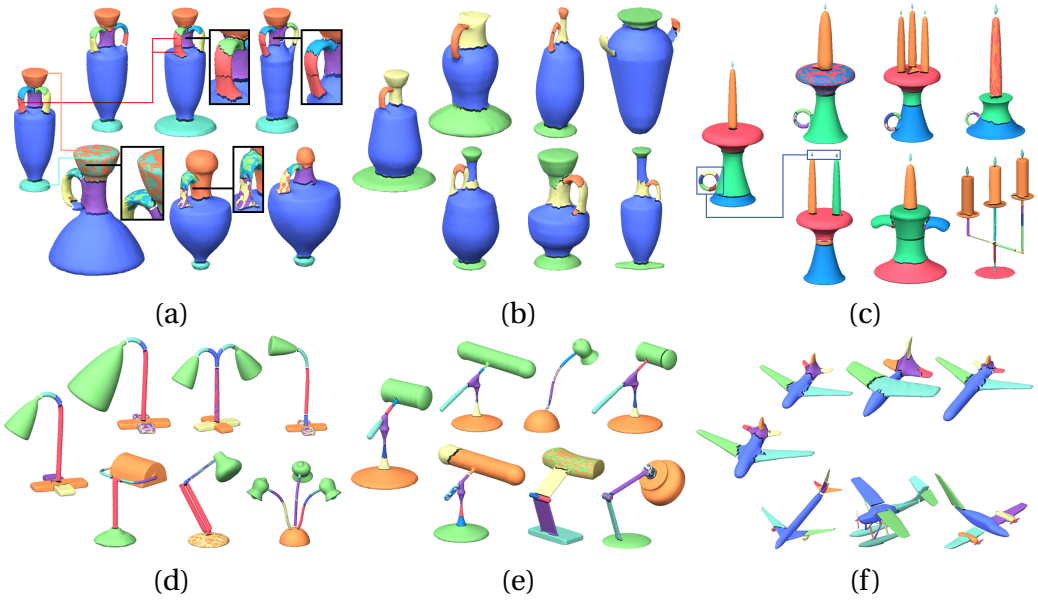


Figure 4.4: *Matching between shapes. In each set, the source shape (left) is matched with three nearest neighbors according to SHED (top), and three additional shapes which are not neighbors (bottom). Multiple target parts that match the same part in the source shape are marked with the same color (see red line, top insets in (a)). A single target part that is matched with multiple parts in the source shape is marked with mixed colors (see orange and cyan lines, bottom insets in (a)). Note that minor differences in the segmentation do not affect the matching or nearest neighbors computation (a, d, f). On the other hand, significant differences in the segmentation may lead to incorrect matching (b, e).*

A few examples of matchings between segmented shapes using the above algorithm are shown in Figure 4.4. In each sub-figure, the parts are color coded according to their matching to the shape on the left. Parts that are matched to the same part in the source shape have the same color. Parts that are matched to more than one part in the source shape have the colors of all matching parts mixed in a random pattern. For example, in the bottom left of (a), indicated by cyan and orange lines, both the top and the base of the source vase were matched to the top of the target vase, since it has no base. Similarly, for vases with one handle, both handles of the source vase are matched to the parts of a single handle.

Minor differences in the segmentation of similar shapes do not typically cause significant changes in the matching. For example, as can be seen by the red line in (a), two of the nearest neighbors of the shape have an extra part in the handle.

The extra part is matched to a similar part, and the rest of the matching remains correct. Since the duplicated part is small, the similarity between these shapes according to SHED remains high. Similarly, most of the shapes on the top rows of (c), (d) and (f) have minor differences in their segmentation, yet they are considered similar by SHED. On the other hand, significant differences in the segmentation may lead to incorrect matchings, as can be seen in (b) and (e). The vase in (b) is only segmented into four parts while similar vases are segmented into seven parts. Thus the matching between these vases is weak, and matched parts are not similar in their geometry, scale and structure. This causes SHED to assign low similarity score to similar shapes.

4.4 Distance Formulation

The matching algorithm output is a list of matches $(i, j) \in \mathcal{M}$. The transformation of each part in the shape is directly defined by the matches it belongs to. Each transformation is associated with a cost which is determined by the magnitude of change and the relative volume of parts in the shape. Below we describe the four types of transformations and how their associated costs are computed.

Change of geometry. For each match (i, j) in the mapping, the cost of deforming the geometry of one part into the other is computed using the same formula for $C(i, j)$ in Equation 4.2. Each term is weighted according to the volume of the parts associated with it. For this, we define a match volume $m(i, j) = VOL(i) + VOL(j)$, and normalize it using the sum of volumes of all matches $\hat{m}(i, j) = m(i, j)/\sum_{(i,j)\in\mathcal{M}} m(i, j)$. The geometry cost $C(i, j)$ is then weighted by the normalized match volumes $\hat{m}(i, j)$.

Change of scale. Since the global scale of two shapes can be different, the change of scale between parts must be measured compared to the change of scale in other matches in the mapping. Thus, the scaling costs are computed for each pair of matches (i, j) and (k, l). The scale term is similar to the formula in Equation 4.5 and measures the difference between the change of scale in the two matches:

$$C_s(i, j, k, l) = \frac{\max(s(i, j), s(k, l))}{\min(s(i, j), s(k, l))} - 1. \tag{4.7}$$

Note that $C_s = 0$ when the scale change of the two matches is exactly the same, and $C_s = 1$ when the magnitude of change in one match is exactly twice than the other match. The scale costs are weighted by $\hat{m}(i,j) \cdot \hat{m}(k,l)$, such that the total weights of all the pairs which contain match (i,j) is $\hat{m}(i,j)$.

Change of position. To detect a part that changed position, it must be compared with its environment, so the position costs are also computed for each pair of matches (i,j) and (k,l). We compare the graph distance of parts i and k in the first shape $g(i,k)$ and the graph distance of parts j and l in the second shape $g(j,l)$:

$$C_p(i,j,k,l) = abs(g(i,k) - g(j,l)). \tag{4.8}$$

Note that if a part is duplicated, we compare the adjacency with the most similar instance, such that if several parts are duplicated together as a group they will only be compared to parts in the same group. The position costs are also weighted by $\hat{m}(i,j) \cdot \hat{m}(k,l)$.

Duplication costs. When a part is duplicated, there are two or more matches with the same part. Each of the matches incurs the above costs if applicable. In addition, we aggregate the volume of the shape that is being duplicated, by summing the volume of all parts in all matches and subtracting the total volume of the shapes. The remainder is the volume of all parts (in both shapes) that appear twice or more in the matches. The duplication cost is normalized by the total volume of the matches, so it represents the percent of matches that have duplicated parts. It is formulated as:

$$C_d = \frac{\sum\limits_{(i,j)\in\mathcal{M}} m(i,j) - \sum\limits_{i\in\mathcal{S}} VOL(i) - \sum\limits_{j\in\mathcal{T}} VOL(j)}{\sum\limits_{(i,j)\in\mathcal{M}} m(i,j)}, \tag{4.9}$$

where $\mathcal{S}$ and $\mathcal{T}$ are the shapes being compared. Note that we do not define a cost for parts that were added, since adding a new part can be thought of as duplicating the most similar part and morphing it to the desired shape.

Aggregation and weight learning. The shape edit distance is formulated as a weighted sum of the above costs:

$$\begin{aligned}\mathrm{SHED}(\mathcal{S},\mathcal{T}) = w_g \cdot & \sum_{(i,j)\in\mathcal{M}} \hat{m}(i,j)\cdot C(i,j) \\ +w_s \cdot & \sum_{(i,j)\in\mathcal{M},(k,l)\in\mathcal{M}} \hat{m}(i,j)\cdot\hat{m}(k,l)\cdot C_s(i,j,k,l) \\ +w_p \cdot & \sum_{(i,j)\in\mathcal{M},(k,l)\in\mathcal{M}} \hat{m}(i,j)\cdot\hat{m}(k,l)\cdot C_p(i,j,k,l) \\ +w_d & \cdot C_d,\end{aligned} \tag{4.10}$$

where w_g, w_s, w_p and w_d are the respective weights of the geometry term, scale term, position term and duplication term. Since semantic similarity between shapes is a subjective matter, it makes sense to learn the values of these weights from user input. However, similarity or semantic distance between two shapes cannot be quantified numerically by the user. Instead, we ask users to indirectly provide the semantic similarity of a set of shapes by generating categorization trees, which group together similar shapes in several levels of hierarchy. For more details see Section 4.5. To learn the weights from the categorization trees, we extract trios of shapes, where in each trio two shapes are similar (i.e. they belong to the same subtree of depth two), and the third shape is semantically far (i.e. it belongs to a different subtree). Each trio of shapes defines a relative relation of the form *"shape A is closer to shape B than to shape C"*. Each categorization tree provides many thousands of trios, from which we randomly select 1000 trios as a training set. To learn the weights from such relations, we employ a weight learning scheme suggested by [Schultz and Joachims, 2004]. Each relation between shapes A, B, and C, is transformed into a constraint of the form: $D(A,C) - D(A,B) \geq 1$ where $D(A,B)$ is the weighted distance between shapes A and B. Then, a convex quadratic optimization is formulated and solved similarly to a support vector machine. For more details see [Schultz and Joachims, 2004].

Using this method, we can fine tune the weights for a specific set of shapes such as lamps or vases. For example, the scaling differences between parts affects the semantic distance between lamps more than it affects the semantic distance between vases. Alternatively, we can use trios from categorization trees of several sets of shapes to learn a global set of weights. Using this method, we propose a set of default weights (see Table 4.1) that would approximate well the semantic similarity of any set of shapes. Note that these weights also reflect the relations between the different units in which the different costs are measured.

Set Name	Default	Lamps	Candles	Vases	Airplanes
Geometry	0.4795	0.4376	0.2779	0.4788	0.4285
Scale	0.1258	0.1921	0.1794	0.0256	0.0206
Position	0.0034	0.1216	0.1203	0.1697	0.0047
Duplication	0.3914	0.2486	0.4224	0.5396	0.5462

Table 4.1: *Learned weights for different sets of shapes. Each column is normalized such that its sum is one.*

4.5 Evaluation

The distance between two shapes cannot be directly measured or estimated numerically by a human observer, hence evaluating the accuracy of a similarity measure is somewhat challenging. Still, we are able to compare SHED with state-of-the-art distance measures, namely the light field descriptor (LFD) [Chen *et al.*, 2003] and the spherical harmonic descriptor (SPH) [Kazhdan *et al.*, 2003], and demonstrate its success in various applications. We evaluate the results quantitatively using ground truth data for shape exploration and clustering, and qualitatively for nearest neighbors queries and embedding, where ground truth data is not well defined.

Datasets. We evaluate SHED using three sets of shapes from the COSEG dataset [Wang *et al.*, 2012] and three sets from PSB [Chen *et al.*, 2009]. In addition, we collected a set of airplanes from Google Warehouse and other online resources. The airplanes and COSEG datasets were enriched by introducing finer intra-class variation. The enriched sets include 100 lamps, 80 vases, 70 airplanes, and 40 candelabra, and contain shapes that vary in their part composition, geometry, and articulation. The PSB sets include 20 humans, 20 hands and 20 Teddy bears, which vary mostly in articulation.

Categorization trees. We present an application where categorization trees of shapes are automatically generated for each enriched set. The resulting trees hierarchically organize the shapes in a set and can be used for exploration. The trees are created using Self-Tuning Spectral Clustering [Zelnik-Manor and Perona, 2004], which is a non-parametric clustering method, i.e., the number of clusters in each set is selected automatically. We used this method recursively to build a categorization tree for each distance measure (SHED, LFD, and SPH). An example of the generated trees on a subset of shapes is presented in Figure 4.5, where the

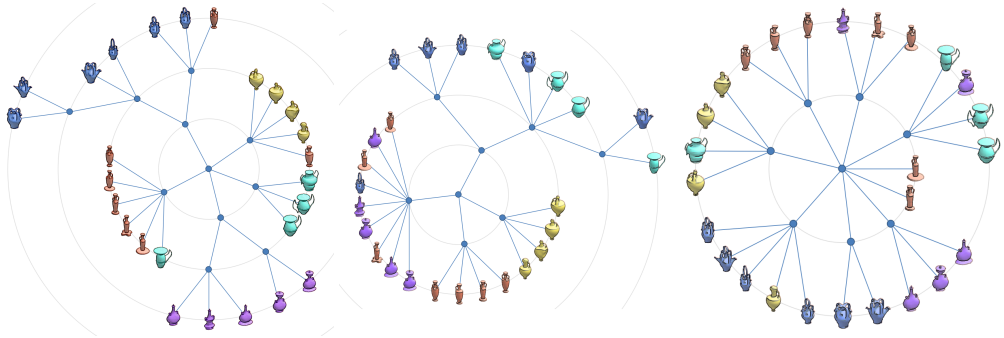


Figure 4.5: *Categorization trees automatically generated for a set of vases according to SHED, LFD and SPH. The vases are colored according to their shape style. Note that the organization of shapes is more consistent when using SHED (3 categorization errors) than when using LFD or SPH (6 categorization errors each), as seen by the number of shapes with a different color than their lowest level neighbors in the tree.*

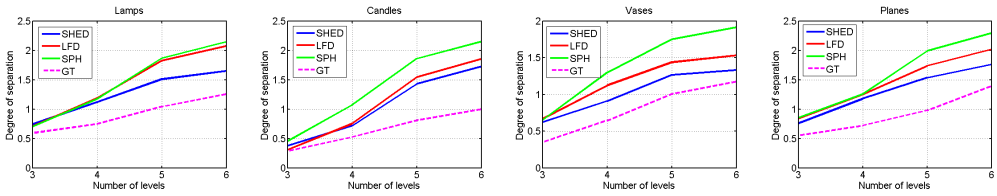
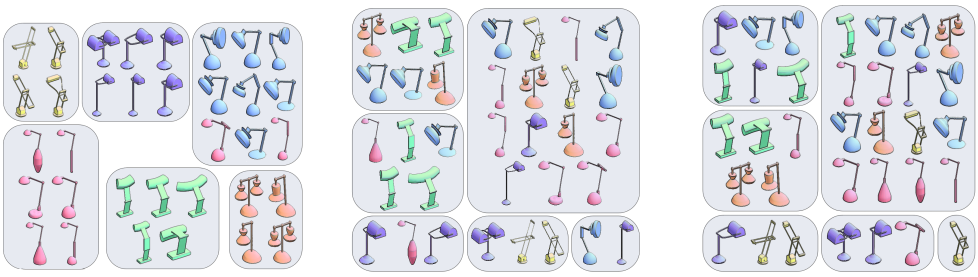


Figure 4.6: *Comparison of automatically generated trees to ground truth trees, according to the average difference in the degree of separation.*

shapes are colored according to their shape style. Note that the tree generated using SHED has fewer categorization errors. The generated trees for the full sets can be found in the supplementary material.

To evaluate the quality of the generated trees in a quantitative manner, we use multiple ground truth categorization trees. Since creating a single categorization tree of a set may be subjective, we asked three expert users to independently create a tree for each enriched set. All of the ground truth trees can be found in the supplementary material. The ground truth trees are compared to the generated trees by averaging the difference in the *degree of separation* (DoS) between each pair of shapes in the trees. The DoS is defined as the length of the path between two shapes in the tree [Huang *et al.*, 2013b]. The average difference of DoS measures whether shapes are organized in a similar manner in two trees, without being influenced by the specific structure of each tree. To compare the trees at different levels of granularity, we truncate the trees up to



SHED LFD SPH

Figure 4.7: *Comparison of clustering results according to SHED, LFD, and SPH on a set of lamps. The shapes are clustered into six groups and colored according to their ground-truth clusters.*

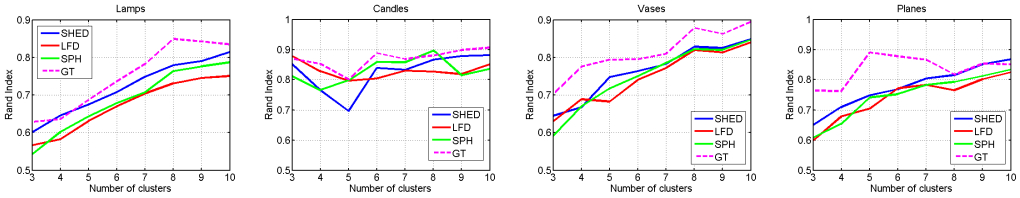


Figure 4.8: *Comparison of automatically generated clusterings to ground truth, according to the Rand Index (see text for details).*

a given number of levels by connecting all the shapes in lower levels directly to their ancestor at the lowest allowed level. The results for a level are given by averaging the difference in DoS over all pairs of shapes and all ground truth trees. The results are shown in Figure 4.6 (lower values imply trees closer to the ground truth). The curve labeled GT denotes the average difference in DoS between the ground truths. It indicates how much variation exists among the different ground truths and establishes a bound for the accuracy. Note that trimming a tree after two levels effectively provides a quantitative comparison of the first level of clustering. Similarly, trimming the tree after three levels provides a comparison of the clustering generated in the second level, and so on for other levels.

Clustering. In addition to the hierarchical clustering, we also experiment with clustering when the number of clusters is known in advance. We cluster each set of shapes using the self-tuning spectral clustering method mentioned above [Zelnik-Manor and Perona, 2004], this time providing the number of clusters as a parameter. We compute ground truth clusterings from each ground truth tree by measuring the degree of separation between every two shapes, and then using the computed DoS as a measure of shape similarity to cluster the

shapes with the same clustering method. We generate clusterings according to SHED, LFD, and SPH and measure the difference between the generated clusters and the ground truth using the Rand Index [Chen *et al.*, 2009]. Figure 4.8 shows the average Rand Index over all ground truths for each set and measure (higher values imply clusters closer to the ground truth). The curve labeled GT denotes the average Rand Index between the ground truth clusterings. It indicates the level of agreement between clusterings generated from different ground truth trees. Figure 4.7 shows visual results for a subset of lamps.

Shape retrieval. As a shape retrieval experiment, a nearest neighbors search was performed for each shape according to SHED and LFD. Figure 4.9 shows a selection of shapes from four different sets along with the retrieved nearest neighbors. The full results containing each of the shapes as a query are available in the supplementary material. The distances measured by SHED reflect changes in part composition such as parts that change position on the graph or parts that exist in one shape and not the other, as well as changes in geometry. Therefore, shapes retrieved using SHED tend to have similar part composition. For example, vases tend to have the same number of handles as the query shape (g, i), and candelabra tend to have a similar number of candles (e). In contrast, some of the shapes retrieved by LFD have a different shape structure (c, i). Additionally, SHED retrieves shapes whose parts have a similar geometry to the parts of the query shape (b, f, g, k), whereas shapes retrieved by LFD are more varied. Moreover, SHED deals particularly well with articulations (a), added parts (b), and partial shape matching (h), which pose a challenge to existing methods.

Ground truth data is not well defined for such tasks in intra-class scenarios, where all the shapes belong to the same class. For such scenarios we show qualitative results only. However, for inter-class scenarios, we can quantify how many of the retrieved shapes belong to the same class as the query shape. Figure 4.10 shows the precision recall curves obtained for all shapes from the PSB sets using SHED, LFD and SPH. The curve labeled "SHED Equal Weights" shows the results when all weights are set to 1. The curve labeled "SHED" shows the results when using the default weights suggested in Table 4.1. Note that these weights were learned using a different sets of shapes, and the results could be improved further by fine-tuning the weights specifically for the PSB sets.

Embedding to a lower dimension. Another important application that benefits from defining a more accurate distance measure between shapes is mapping a set of shapes onto a low dimensional manifold. We use standard

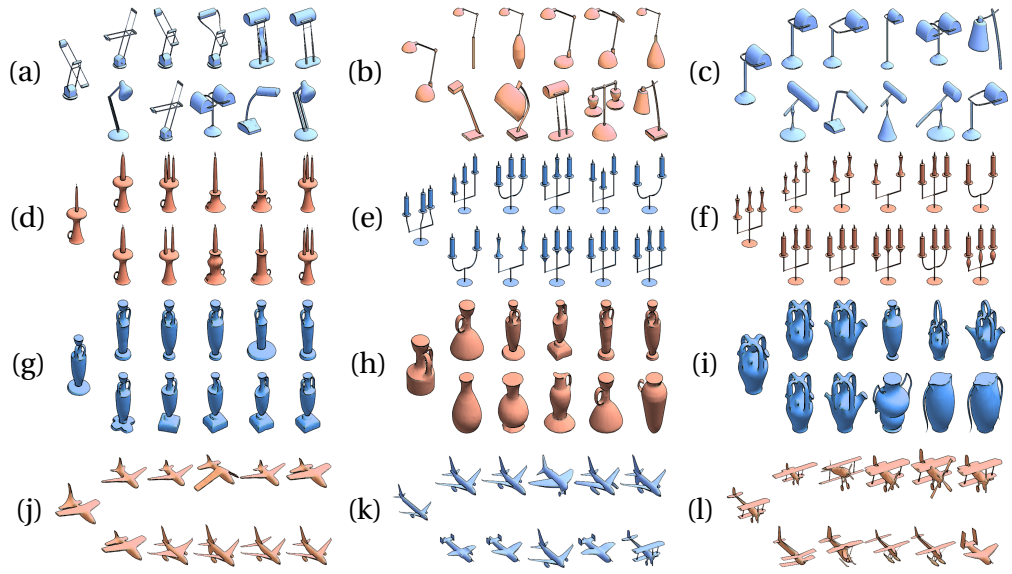


Figure 4.9: *Shape retrieval for four sets, ordered by similarity to a query. In each example, the shape on the center left is the query, the first row are the 5 nearest neighbors ordered according to SHED, and the second row are the neighbors ordered according to LFD.*

multi-dimensional scaling (MDS) to generate an embedding of a set of shapes in two dimensions. In Figure 4.3 we show a toy example comparing the embedding generated by SHED and LFD for a small set of vases. For inter-class similarity estimation, we show in Figure 4.11 the MDS embedding of shapes from the PSB sets using SHED, LFD, and SPH. The figure clearly shows that SHED produces an intuitive map with a significant distinction between different sets, while LFD and SPH tend to produce less organized maps where shapes of different sets are mixed together. This experiment and the quantitative evaluation in Figure 4.10 allow us to conclude that SHED is effective when used to separate shapes into different classes (inter-class context), while the previous experiments show that SHED is able to appropriately quantify finer shape differences, which is of importance in an intra-class context.

Weights. The weights for the sets of lamps, candles, vases, and airplanes were learned from training sets of 1000 trios each, obtained from the ground truth of each set separately. In addition, default weights were learned using a training set of 1000 trios, obtained from the ground truth of all four sets collectively. The default weights were used to produce the results for the PSB sets in Figures 4.11 and 4.10. The weights for each set are given in Table 4.1.

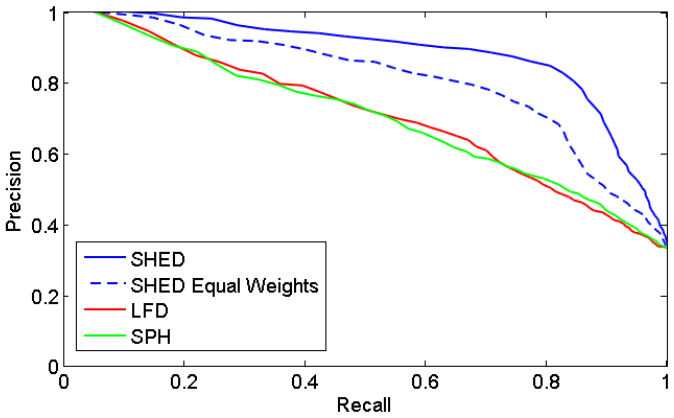
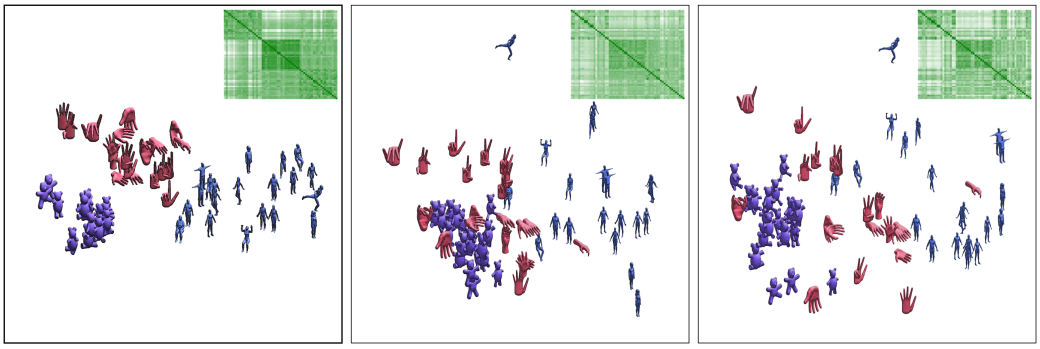


Figure 4.10: *Precision-recall on sets of articulated shapes.*

Timing. Our method can be decomposed into two parts: finding the matching between two shapes and computing the SHED according to a given matching and weights. The computation time of the matching algorithm described in Section 4.3 depends on the number of parts in each shape, and takes up to 5 seconds for shapes with up to 20 parts. Given the matching and weights, computing the SHED takes a fraction of a second, and the computation of the entire set of 100 lamps, or 4950 pairs of shapes, takes a total of 9 seconds. Segmenting the shapes using [van Kaick *et al.*, 2014] takes up to 5 minutes per shape. Note that the segmentation method can be easily replaced. In some cases the segmentation of shapes can be given as input, in which case the method is very fast to compute.

4.6 Conclusion

We introduce SHED, an edit distance that quantifies shape similarity based on a structural comparison of shapes. The shape edit distance captures re-arrangements, additions, and removals of parts. We show a variety of applications which benefit from an accurate distance measure between shapes. Finally, we demonstrate that SHED leads to a more intuitive estimation of the similarity between two shapes than state-of-the-art methods, when comparing shapes within the same class as well as shapes from different classes.



SHED LFD SPH

Figure 4.11: *Embedding obtained with multi-dimensional scaling on a set of articulated shapes with three classes. The insets show the distance matrix for each method, where dark green is low distance and white or light green is high distance. Note how SHED groups the shapes into their respective classes, while the distance matrices and embeddings given by LFD and SPH are less organized.*

Future work and limitations. The current formulation of SHED takes into account the similarity of the shape parts and the shape structure in terms of connectivity of the parts. Additional relationships between parts can be considered, for example, the difference in rotation of pose after an alignment of matched parts. Incorporating pose considerations may constitute an advantage on sets where the pose of the shape parts is one of the main dissimilarity factors, while it may be less suitable for more general sets where pose-invariance is sought.

An adequate segmentation of the shapes is required for the computation of SHED. In general, segmentation is an ill-posed problem. As a practical solution, we opted to use a segmentation into approximately convex parts, although other segmentation methods can be used. For example, methods that aim at obtaining a close-to-semantic segmentation of the shape are possible, although their usage would require the introduction of more sophisticated measures to compare the geometry of parts.

Finally, distances between shapes are subject to interpretation and are dependent on the semantics of the shapes. Thus, we would like to conduct an investigation to gain insight on how humans perceive finer shape differences, to enhance our edit distance. Quantification of intra-class distances is still an open avenue for further research.

5 Symmetry Aware Correspondence

Finding correspondence between shapes is a fundamental problem in computer graphics. Many existing methods aim to find point-to-point correspondences [Kim *et al.*, 2011; Ovsjanikov *et al.*, 2012], or a mapping between feature points [Berg *et al.*, 2005; Leordeanu and Hebert, 2005; Kezurer *et al.*, 2015]. Shapes with intrinsic symmetry pose a particularly difficult problem for such methods, as there can be many solutions which are equally likely to be correct; an inverted map where the left side of one shape is mapped onto the right side of the other and vice versa may not incur additional cost. Moreover, in some cases combinations of several solutions are also likely, e.g. a subset of correspondences from one solution and another subset from another solution.

To solve this problem, some methods factor the symmetry out of the map computation, thus finding a map which may be correct, inverted, or a non-continuous blend of several symmetric maps [Ovsjanikov *et al.*, 2012; Sahillioğlu and Yemez, 2011]. Another prominent method, Blended Intrinsic Maps [Kim *et al.*, 2011], outputs a continuous map, which is usually not inverted, as long as there is some isometric distortion between the source and target. This is a result of the natural distortion in elbows and knees due to pose changes, which is non-symmetric. However, the blended intrinsic maps are computationally expensive and may be distorted, and they do not allow matching partial shapes or shapes with different topology.

We propose to use a symmetry aware correspondence between segments as an in-between step. The matching between segments can later be used to improve the point-to-point map between the shapes. Our symmetry aware correspondence factors out symmetries by allowing all symmetric segments to match to each other as a clique. Thus, the correct and inverted maps are merged onto the same map. Clearly, our method does not produce the same level of details as existing methods. However, there are numerous advantages to this

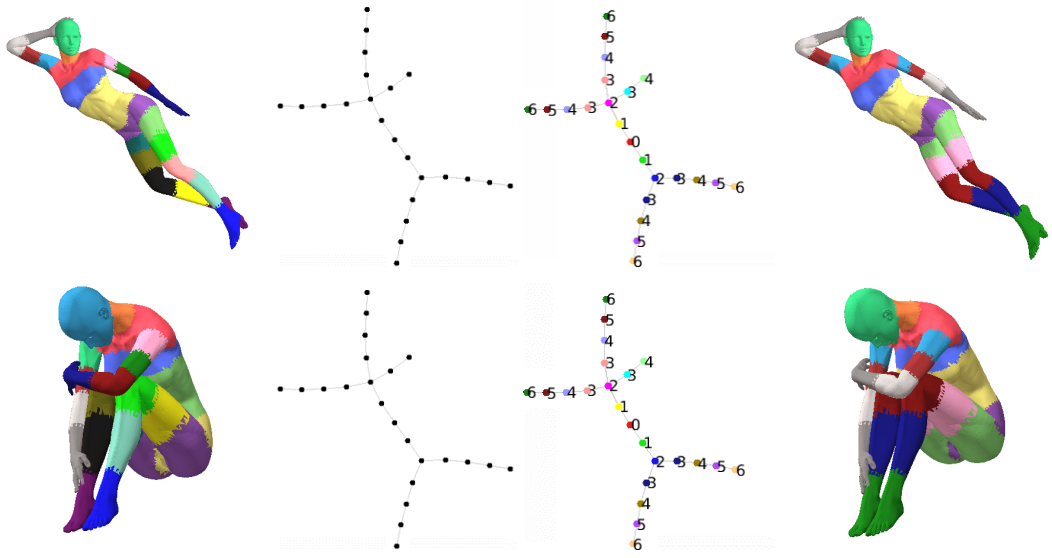


Figure 5.1: *Overview of our symmetry aware correspondence method. The two shapes (presented here one above the other) are first mutually segmented into consistent segments (a) and a shape graph is created for each segmented shape (b). Note that the shape graphs are in fact isometric even though the shapes vary greatly in pose. Then, the shape graphs are matched without breaking node symmetry. In (c), the value of each node marks the graph distance from the root node, and matching nodes are shown with the same color. The matching between graphs induces a symmetry aware matching of the segmented shapes (d).*

combined approach of using segments and factoring symmetry out. In most cases, there is only one solution to the optimization, which allows a straight forward optimization which is very quick. The solution is more stable and there is less distortion between segments compared to state of the art methods. We show that this correspondence is still useful for significantly improving point-to-point maps. In addition, we can easily identify parts that do not have a match in the other shape, and discard those parts from the point-to-point map. This allows partial matching of shapes with different topology. Another application of our method is detection of intrinsic symmetries within a shape by matching the shape to itself. The symmetry detection is also a lot quicker and more stable than existing methods.

Figure 5.1 provides an overview of our method. First, we segment the pair of shapes using HKS descriptor [Sun *et al.*, 2009]. This co-segmentation is semantically consistent between the two shapes, i.e. the segment edges are

usually in a similar semantic position. Then, we build shape graphs from the segments. Geometric data is not incorporated in the shape graphs, which indicate only the structure of each shape. The shape graphs are matched in a symmetry aware manner using a sparse graph matching technique. This provides the matching between segments, or segment symmetry detection when a shape is matched to itself.

To compute point-to-point maps between the shapes, we use the functional maps framework of [Ovsjanikov *et al.*, 2012]. In that paper, a matching between segments is used to improve point-to-point maps. The segments used in [Ovsjanikov *et al.*, 2012] are sparse (they do not cover the entire shape) and non symmetric, whereas we provide a dense map in which every segment can have multiple connected components which denote symmetric parts in the shape. Using the symmetry aware correspondence between matches significantly improves the point-to-point map.

We evaluate our method in a number of experiments. First, we compare the correspondence between segments to blended intrinsic maps, or BIM [Kim *et al.*, 2011]. We compare to these point-to-point maps by transferring the segments from the source shape to the target shape using the map and counting the number of vertices that are mapped to the correct segment. This is possible for datasets where the mapping between vertices is known, such as TOSCA [Bronstein *et al.*, 2008], SCAPE [Anguelov *et al.*, 2005] and FAUST [Bogo *et al.*, 2014]. For datasets where there is no ground-truth mapping between shape, such as SHREC [Li *et al.*, 2012a], and for inter-class correspondence such as matching a human shape to a gorilla, we provide visual results which can be assessed in a qualitative manner. Typically, the segments that BIM create on the target shape are more distorted than our method, and tend to be cut in different semantic locations from the source segments.

In another experiment, we generate point-to-point maps according to the matching between segments, using the functional maps method of [Ovsjanikov *et al.*, 2012]. We show that using the symmetry aware segments significantly improve the accuracy of the map, even without producing a one-to-one mapping between segments.

5.1 Related Work

Methods for computing correspondences between shapes can be divided into two categories according to their output: dense correspondence maps and sparse correspondence between feature points on the shape. A survey of shape correspondence methods is provided in [Van Kaick *et al.*, 2011]. In recent years, there were a few prominent works in the area of dense correspondence maps. In [Ovsjanikov *et al.*, 2012], a functional maps approach was presented, in which a mapping between function spaces on the shape is sought instead of a mapping between points on the shape. This way, many global constraints can be formulated as linear constraints on the functional maps, such as landmark correspondence and segment correspondence. We use this framework to derive point-to-point maps from our symmetric segment correspondences in Section 5.4. Another state-of-the-art work is Blended Intrinsic Maps or BIM [Kim *et al.*, 2011], in which several maps with locally low distortion are blended to form a single map which has low distortion everywhere. Sparse correspondence between feature points is an equally important problem, since sparse correspondences can usually be transferred into dense correspondence maps using stable algorithms. Recent works in this area include [Sahillioğlu and Yemez, 2011; Kezurer *et al.*, 2015].

Shapes with intrinsic symmetry have been identified as a challenge in many of these works. The problem is inherent to the task of intrinsic shape matching: for shapes with perfect intrinsic symmetry, there are several symmetric maps, and none of them can be defined as more correct than the other. As most optimization methods become unstable when there is more than one correct solution, shapes with symmetry commonly produce less accurate maps or non-continuous maps in which each part is taken from a different potential map. As a result, the evaluation of some methods is done with respect to symmetry, i.e. every potential symmetric solution is considered correct [Ovsjanikov *et al.*, 2012; Sahillioğlu and Yemez, 2011]. Alternatively, the ambiguity can be resolved using a small set of landmark correspondences, which are often manually selected [Ovsjanikov *et al.*, 2010]. In BIM [Kim *et al.*, 2011], maps are always continuous, and thus symmetric maps can not blend and only one of the potential maps is selected as the final output. In addition, most shapes with bilateral symmetries (e.g. humans) have intrinsic differences between front and back (such as feet, knees, and elbows), which can be used to find the correct map. Still, no solution is given for the generic symmetric case (e.g. octopuses, ants, etc.).

Some methods have specifically targeted matching between symmetric shapes. In [Ovsjanikov *et al.*, 2013], a symmetry map is used to compute a set of symmetric correspondences between shapes. However, for this method a reference shape with a known symmetry map is necessary. This work is an extension of the symmetry invariant function space which is used in [Lipman *et al.*, 2010] for symmetry detection using self-correspondence graphs. Recently, a stable region correspondence has been proposed [Ganapathi-Subramanian *et al.*, 2016]. In this work, points on the shape are ordered according to their feature function values, and the rank of each point is used to find correspondence between stable regions. Since the values of the feature functions are discarded and only the rank is used, this method can successfully find stable regions in shapes with very different geometry. This approach is quite similar to our work: we also use the relative value of the feature function rather then the absolute value, by quantizing the feature function and using it to construct a shape graph. However, we produce finer correspondences and smaller symmetry orbits. For example, in the stable region approach, extremities of the shape such as the legs and tail often belong to the same stable region, while we are able to distinguish between front legs, hind legs, tail, etc.

5.1.1 Segmentation and shape graphs

Correspondence between segmented shapes is related to co-segmentations of shapes, which many works have explored [Sidi *et al.*, 2011; Kalogerakis *et al.*, 2012]. We elaborate on a few of these works in Section 4.1. Note that the goal of these works is usually a semantic high-level segmentation of the shapes, while we aim to find correspondences between smaller segments which are more useful for subsequent matching of feature points. Shape Edit Distance (see Chapter 4) uses a similar notion of segmenting shapes into parts and matching the graph of parts, and indeed was an inspiration to this work. However, in this work we aim to identify correspondences between shapes with different geometry, while SHED uses the shape graphs to identify the most similar shapes and accentuate the differences between them. Thus, in SHED the nodes of the shape graphs are rich with geometric data and are more sensitive to geometry changes.

Shape graphs were also used in [Singh *et al.*, 2007], where the `Mapper` graphs were introduced and used for shape similarity. We are inspired by these graphs and use a similar construct to compute the symmetry aware correspondence between segments.

5.2 Consistent Segmentation

The input to our method is two 3D meshes. The first step is to co-segment these shapes in a consistent manner. By that we mean that the edges between segments have a similar position in both shapes. For example, if there is a cut above the knee in one shape, there should be a cut above the knee in the other shape, even if their geometry and triangulation are different. The consistency of the segmentation is important for two reasons. First, a consistent segmentation implies that two similar shapes produce a similar shape graph which is easy to match. Second, a matching between consistent segments induces a more accurate matching between vertices. Note that we do not seek a semantic segmentation of the shape; a semantic part may be cut into several segments.

To consistently segment the shapes, we use a quantization of the *heat kernel signature* or HKS [Sun *et al.*, 2009]. We compute the HKS for each shape, and quantize the HKS value into k bins, so each vertex on the shape has a label between 0 and $k-1$ based on its HKS value. Typically, the lowest HKS value is located at the center of mass of the shape and the highest HKS values are at the shape's extremities. We then segment the shapes by forming a segment for each connected component of vertices with the same label.

In the next step, we build a shape graph in which each segment is connected to its neighboring segments. The approach of representing shapes using a graph of segments is similar to the `Mapper` graphs of [Singh *et al.*, 2007]. They use overlapping clusters to create a shape graph, in which two clusters are connected if there is an overlap between them. It has been shown that the `Mapper` graphs are useful for computing shape similarity and produce similar graphs for different poses of the same shape. Our shape graphs have no overlap between segments, and we deliberately use simpler shape graphs with fewer nodes to capture common elements of shapes with significantly different geometry. Our goal is to use these shape graphs to compute a matching between the shapes. An example of consistent segmentation is given in Figure 5.1(a). The shape graphs that correspond to these segmented shapes are shown in Figure 5.1(b).

Often, similar shapes have similar HKS values, which in turn generate isomorphic shape graphs. However, small changes in the shape may cause the HKS value to differ such that extra segments are generated. These differences may occur for some selections of k (the number of bins in the quantization) and not for others. The question arises, how to select k. We find that there is a range

of values that produce acceptable graphs. For small values of k, the resulting segments are quite large and not very indicative of the shape structure. For very large values of k, the segments are usually too small and the shape graph does not convey well the structure of the shape. In our experiments, we found that values between 6 and 12 produce acceptable results. To set the exact value of k we follow the assumption that shapes which are very different from each other are not likely to produce similar shape graphs. Thus, we assume that isomorphic graphs will produce a better correspondence between the shapes than non-isomorphic graphs, and select a value of k which produces isomorphic graphs of the two shapes, if such k exists. For most isometric pairs of shapes, there is a value of k which produces isomorphic graphs: in the sets of TOSCA and SCAPE, isomorphic shape graphs can be found for 95% of pairs. If there is no selection of k which produces isomorphic graphs, we select the k which produces the most similar graphs of the two shapes in terms of the number of nodes and their connectivity.

The combination of using a stable descriptor (HKS) and searching for segmentations with similar structure generates consistent segmentations for shapes from the same set (for example two human shapes) as well as shapes from different sets, such as matching a cat to a dog or a man to a gorilla.

5.3 Symmetry Aware Matching

After we find a consistent segmentation, we compute a symmetry aware matching between the shape graphs. We do not keep geometric data of each segment in the shape graph. Instead, we rely only on the graph structure of the shape to compute the matching. This allows us to match very different shapes as long as they have the same intrinsic structure. The only geometric information we keep for the graph is the root of the shape. We define the root of the shape graph as the segment in the shape which has the lowest HKS value. This segment is typically located at the center of mass of the shape. Then, we measure the graph distance between each node and the root node and assign this value as the label of the node. Therefore, a node in the shape graph can only be identified by its distance from the root node and its connectivity to other nodes, and symmetric branches cannot be distinguished from each other. Branches which are not symmetric can be distinguished by their position in the shape graph. For example, the nodes of each leg have the same properties, while the branches of an arm and a leg can

be distinguished by the position of the head, even if each branch has the same number of nodes.

Our symmetry aware matching technique takes advantage of these properties to quickly identify symmetric nodes in the graph structure. Finding a matching between isomorphic graphs is usually simple and can be done in several different ways. To provide a solution for non-isomorphic graphs as well, we use a spectral method that produces a sparse vector in which non-zero elements mark correspondences between nodes. This method matches groups of symmetric nodes in one shape to their matching symmetric nodes in the other shape.

5.3.1 Formulation

We follow the quadratic assignment model definitions and formulations presented in SHED (see Section 4.3). Similarly, we define a *unary term* and a *binary term* and use them to construct an affinity matrix M. The unary terms describe the relation between a segment i from one shape and a potentially matching segment j in the other shape. The binary terms describe the compatibility between a match between two segments (i, j) and another match between segments (k, l). For example, if segments i and k are adjacent, and so are j and l, there is a high compatibility between the matches. But if one pair of segments is adjacent and the other is not, there is a low compatibility between the matches.

The unary cost is the sum of the following three components:
$g_0(i)$: The graph distance between the segment and the root node.
$H(i)$: The histogram of graph distance. For each segment, we count how many segments are adjacent to it (or have a graph distance of 1), how many segments have a graph distance of 2, and so on. This forms a vector which signifies the connectivity of the segment.
$g_l(i)$: The distance of the segment from the closest leaf of the shape graph, or a node with a degree of 1. This term is useful to distinguish between branches of different length in the shape graph (for example an arm with 4 segments vs. a leg with 5 segments).
The first order cost is then given by the following formulation:

$$C(i,j) = \|g_0(i) - g_0(j)\| + \|H(i) - H(j)\| + \|g_l(i) - g_l(j)\| . \tag{5.1}$$

The affinity between the segments is computed similarly to Section 4.3 by:

$$U(i,j) = exp(-C(i,j)/\sigma), \tag{5.2}$$

where in our experiments $\sigma = 0.5$.

For the binary term, we compute the difference of graph distances:

$$d_g(i,j,k,l) = \|g(i,j) - g(k,l)\|, \tag{5.3}$$

where $g(i,j)$ is the graph distance between nodes i and j, and the difference of the unary cost between the matches:

$$d_u(i,j,k,l) = \|C(i,j) - C(k,l)\|, \tag{5.4}$$

where $C(i,j)$ is as defined above. Local structural deformations in a shape, such as missing or additional parts, affect the unary costs of nearby parts by the same amount (for example, adding a constant amount to $C(i,j)$ for every part j in the same branch). The second term d_u is not affected by these changes, so it is helpful when matching partial shapes or shapes with partial matches. Again, the affinity between the two matches is computed by:

$$U(i,j) = exp(-(d_g(i,j,k,l) + d_u(i,j,k,l))/\sigma), \tag{5.5}$$

with $\sigma = 0.5$.

5.3.2 Sparse spectral matching

In common spectral correspondence techniques such as [Leordeanu and Hebert, 2005], the discretization (or binarization) of the output vector is driven by the constraints. For example, when searching for a one-to-one matching, the first selected match for each node determines that there are no other matches in that row. The discretization ends when there are no more possible matches, or for one-to-one matching, when all elements have exactly one match. A similar process is used in the matching needed for SHED (see Section 4.3). The iterative adaptation continues as long as there are possible matches. In our symmetry aware matching, groups of segments of any size can match groups of segments in the other shape. Thus, a matching in which *all* the segments of one shape match *all* the segments of the other shape is valid (though not very useful). A different

method is necessary to guide the discretization. By examining the continuous output vector, we can see that it is dense and unpredictable. The scale of values changes greatly between different nodes in the shape graph. In addition, for some nodes, there are a few significant values in the vector, while for others, all the values are of similar scale. Thus, a threshold which might work for some of the segments may not be effective for others.

In our case, we can use the properties of the shape graphs and the structure of the symmetry aware correspondence to find a sparse solution to the matching problem. The sparse solution we find is easy to discretize as there is a clear distinction between correct matches and incorrect matches. The sparse vector can also be interpreted as a confidence value, so when a node has no matches of high value, we consider it a low confidence node and discard it from the matching. This is useful for matching partial shapes or shapes with some added parts.

Our solution is based on the fact that the shape graphs contain only discrete information (the graph distance from the root node), and therefore many nodes share similar properties. In fact, the first order term of many nodes is equal. We can compute a matching between shape graphs using first order data only, for example by matching each node i with the nodes j which have the smallest first order costs $C(i,j)$ (see Equation 5.1). We observe that in most cases, the matching found by the first order data is locally correct, but includes incorrect symmetries. For example, the front legs are correctly matched to the front legs but also matched to the hind legs, or the head is correctly matched to the head but also matched to the tail. Thus, our goal is to break the incorrect symmetries and keep only the globally consistent ones: for example, front legs can be distinguished from hind legs by using the position of the head in the shape graph. However, we would like to keep the local relations between nodes, such as the order of nodes in a branch.

Suppose the matrix M contains only the first order data on its diagonal, and zeros everywhere else. If the values on the diagonal are unique, the primary eigenvector of M is 1 for the highest first order affinity, and 0 everywhere else. However, the first order values are not unique. First, compatible matches tend to have similar values, for example in a perfect matching the values of all matches that belong to it would have an affinity of 1. Second, in our discrete shape graphs multiple matches can have the exact same value, and therefore the same affinity. Thus, the primary eigenvector of M consists of a random vector in the *subspace* that is spanned by multiple correct matchings, and zeros everywhere else. Note

that this vector is sparse. In other words, the eigenvector contains values that correspond to a mix of matches from all the possible matchings which are equally likely. These contain symmetries that originate from intrinsic symmetry in the shape, such as left-to-right symmetry in the shapes we experiment with, as well as symmetries that originate in the first order data for which we have additional information, such as matching of arms to legs. We would like to collapse the space of solutions such that only the correct matching forms a possible solution to the optimization. Unfortunately, for shapes with intrinsic symmetry, we do not have enough information to distinguish the correct matching and symmetric matching. However, we have additional information to distinguish arms from legs or the head from the tail.

Our goal is to restrict the eigenvector to a specific subspace which spans only the correct solution (up to intrinsic symmetry), while keeping it sparse. To this end, we use the second order data we have as a *tie-breaker*; it should direct the optimization towards a specific solution within the subspace of possible solutions spanned by the first order data, while staying within that subspace, without overriding the first order data. We thus scale the second order data to a small value such that it can not affect the first order data, for example by dividing it by the number of non-zero elements in the matrix M. This has the effect of producing a sparse vector within the subspace, which respects the second order data as well as the first order data. The sparse vector can be easily discretized by searching for a gap in the values that correspond to the matches of each node. If such a gap does not exists, it means the confidence of the node is low and we do not match it. Note that this scheme relies on the discrete nature of the shape graphs, and would not work in a setting where the nodes contain geometric data which may deem some matches more likely than others.

To summarize, our algorithm consists of the following steps:

- Computing the first order and second order data terms.
- Scaling the second order terms to a small fraction, for example by dividing them by the number of non-zero elements in the matrix.
- Computing the first eigenvector of the affinity matrix M.
- Discretizing the sparse output eigenvector.

An example of the matching between graphs is shown in Figure 5.1(c). Nodes with the same color are matched to each other. The induced matching between shape segments is shown in Figure 5.1(d).

5.4 Evaluation

5.4.1 Qualitative evaluation

To start the discussion of our results, we show a gallery of a few examples for various shapes in Figure 5.2. Similar colors denote matching segments, and the dark gray areas in (e) are segments for which no matching was found. In (a) and (b), we show typical results for shapes from the same category. An isometric shape graph can be found for almost all of the shapes of the same category. Note that for many categories, relatively small segments can be matched, as can be seen in (b). Using small segments increases the accuracy of the correspondence and is beneficial for the various application discussed below, such as point-to-point maps, symmetry detection and mapping feature points. In (c) and (d) we show cross-category correspondence, such as matching a man to a woman (c) and a man to a gorilla (d). These shapes have large variations in both pose and intrinsic geometry, yet the shape graphs are still isometric and in most cases the segments are cut in roughly the same positions (e.g. above the neck, below the knee, etc). An exception to this the knee area in (d) where there is a misalignment of the segments due to the short legs of the gorilla. Still, this discrepancy covers a local area and most of the shape is matched correctly.

In (e), we show an example where the shape graph is not isometric within the same category. In our experiments this is a rare case which happens for less than 5% of the shape pairs. In this particular case, the legs of the cat show a lot of flexibility, and essentially emerge from slightly different locations in the torso, thus breaking the symmetry of the shape. As a result, the left leg is not matched. Note that our method discards correspondences with low confidence rather than forcing a match for every segment. Thus, the incomplete matching is still useful for the applications described below without introducing errors.

In (f), we show a matching between two centaur shapes. While most of the symmetries are identified correctly (front legs, arms, etc), the tail is considered symmetric to the hind legs. This is a limitation of our approach, since in this case the branches in the shape graph that correspond to the tail and back legs have exactly the same length and connectivity, and cannot be distinguished according to the shape graph without additional geometric data.

In Figure 5.3, we show an example of matching shapes with substantially different shape graphs. The shape graphs are also displayed. The numbers on the

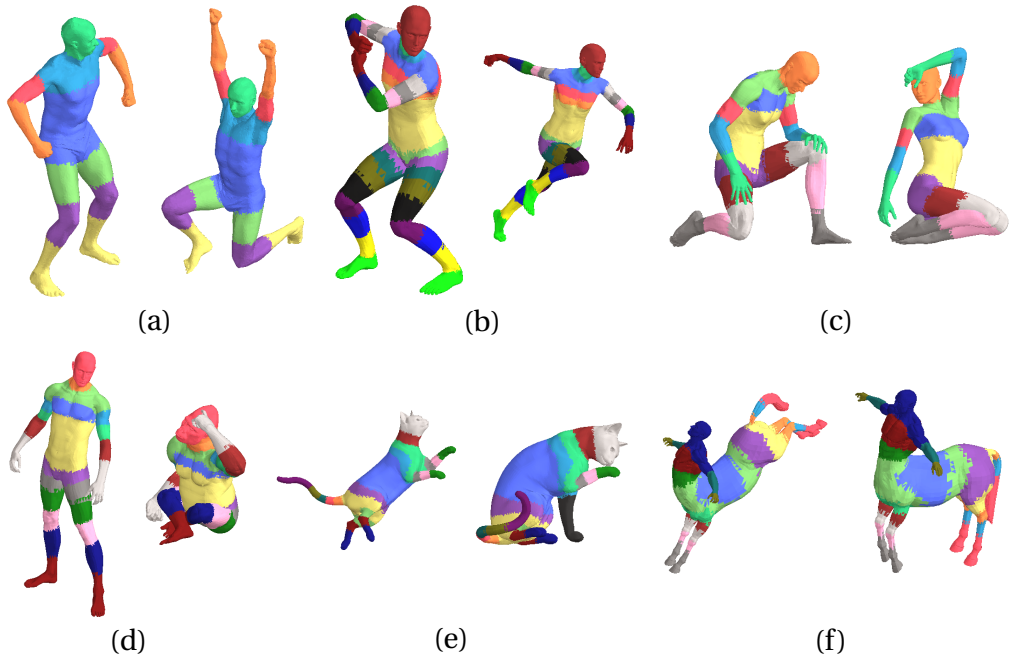


Figure 5.2: *Symmetry aware correspondence between shapes.*

graph denote the node value, or graph distance from the root node, and the colors of the nodes denote the correspondence between nodes (nodes in the second shape which were not matched are black). Note that our correspondence method matches the similar sections of the shapes while leaving dissimilar sections unmatched. Thus, this matching is also useful for the applications described below. Interestingly, even though the shape graph is quite different many of the segments are still cut in similar semantic positions of the shape (for example in the legs and neck).

5.4.2 Comparison to BIM

To evaluate the consistency of the segments we find and the accuracy of the matching between these segments, we compare the segment-to-segment correspondence with BIM [Kim *et al.*, 2011]. The comparison is performed by mapping the segments using the point-to-point map and then counting the percent of vertices which are in the correct segment, weighted by the area covered by each vertex. Ideally, for sets where we have a ground truth mapping of vertices, the segment that contains the source vertex should be mapped to the segment that contains the target vertex. For this quantitative experiment,

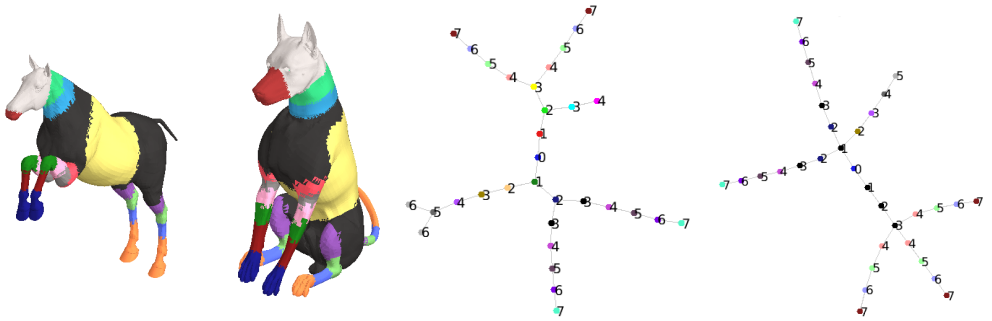


Figure 5.3: *Matching shapes with non-isometric shape graph. (a) Symmetry aware correspondence between a horse shape and a dog shape. (b) The shape graph of the horse shape. (c) The shape graph of the dog shape.*

we map shapes for which we have ground truth in the TOSCA [Bronstein *et al.*, 2008], SCAPE [Anguelov *et al.*, 2005] and FAUST [Bogo *et al.*, 2014] datasets. The results are shown in Table 5.1.

We also provide qualitative results where the slipping of segments is visible, as shown in Figure 5.4. In each subfigure we show the source shape in the center, the matching segments in our method on the left, and the matching segments using BIM on the right. Note that in (a), the matching of the segments in the head, chest and legs is more accurate with our method than with BIM. Similarly, in (b), the difference in matching is particularly visible in the head and arms, where the segmentation of our method more closely resembles the source segmentation. In (c), we provide an example from SHREC dataset [Li *et al.*, 2012a] where there is no ground truth matching between vertices. The head is matched more accurately with our method than with BIM. Another advantage of our method that is illustrated in this example is the matching of partial shapes. In BIM, the tail of the bull is matched to the significantly smaller tail of the pig, causing a lot of distortion in that area. Similarly, the BIM between two shapes always include the entire shape, even for partial shapes. In our method, we can identify segments which do not have a match in the other shape, and consequentially provide a better mapping for the segments which do have a match.

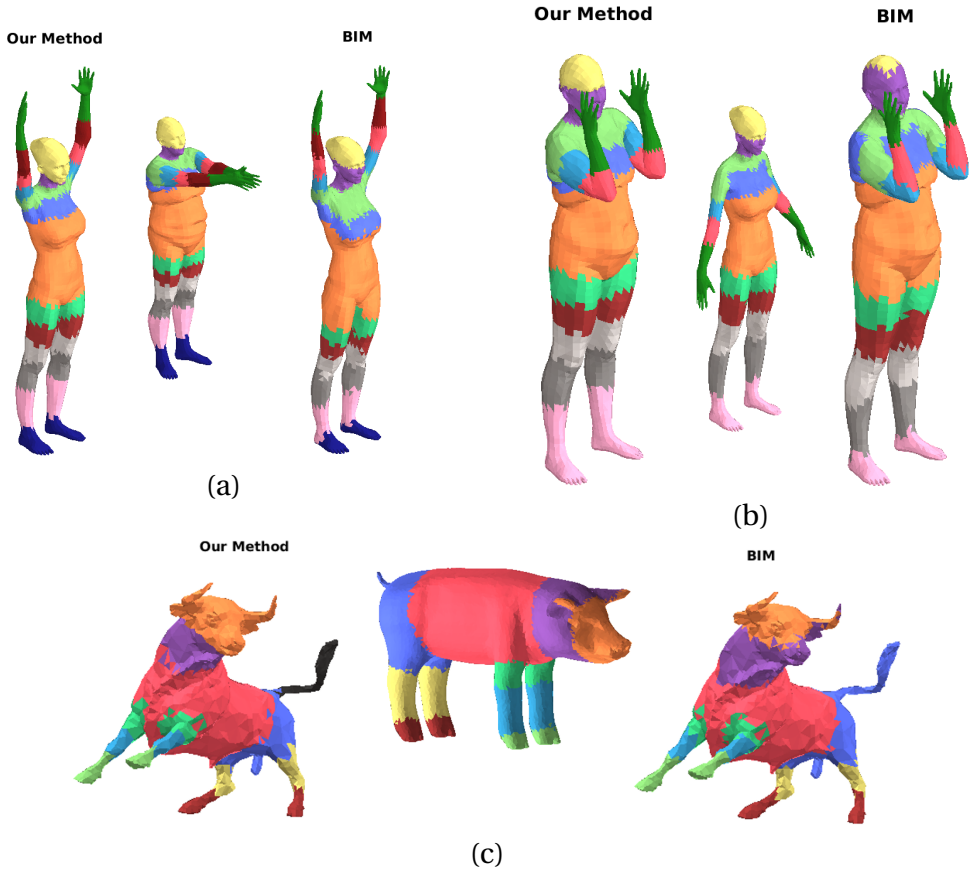


Figure 5.4: *Comparison of our method to BIM. In each subfigure, the segments of the central shape were transferred to the shape on the left using our method and to the shape on the right using BIM. Matched segments are shown in the same color.*

5.4.3 Point-to-point maps

The correspondence between segments can be used to enhance point-to-point maps between the shapes. For this, we use the functional map framework of Ovsjanikov et al. In [Ovsjanikov *et al.*, 2012], the point-to-point maps are improved using large and sparse segments (i.e., the segments do not cover the whole shape). Note that the segments in that work are not symmetric. Naturally, breaking the symmetry of our segments prior to computing the maps can be helpful. However, in this work we aim to show that the matching between symmetric segments, which can be found quickly and in a stable manner, can still greatly improve the point-to-point maps without resolving the ambiguity. The advantage of using our approach is the use of dense and stable segments:

Figure 5.5: *Examples of segment-level symmetry detection.*

our segments cover the whole shape and are smaller and more accurate than the large segments used in [Ovsjanikov *et al.*, 2012].

To evaluate the point-to-point maps, we measure the average geodesic error between points in the target mesh and their ground truth correspondence. In [Ovsjanikov *et al.*, 2012], the functional maps framework was evaluated using WKS descriptors and functional constraints based on matching segments. We compute point-to-point maps using WKS descriptors only (without using segments), using only constraints based on our symmetric segments, and using both. The results are presented in Table 5.2. Our results show a significant improvement of the geodesic error when using our symmetric segments, even without resolving the symmetric ambiguity. Interestingly, when applying the segment-based constraints, the results do not further improve with the addition of WKS descriptors. This suggests that our symmetric segments are fine and accurate enough to capture a lot of the information contained in the point-specific WKS descriptors.

5.4.4 Symmetry detection

Another application of our method is segment-level symmetry detection. Again, we sacrifice the precision of point-to-point symmetry maps to provide a solution which is extremely quick and stable; the entire process of symmetry detection consists of computing the HKS function for the two shapes and solving a single eigenvector problem of moderate size. Symmetry detection is computed by matching a shape with itself. This generates symmetric segments, or cliques of segments where all pairs match to each other. This matching can easily

Category	Symmetry Aware Correspondence	BIM
TOSCA		
cat	0.804639	**0.870072**
centaur	**0.941216**	0.937418
david	**0.966373**	0.946934
dog	**0.936976**	0.907939
gorilla	**0.952195**	0.938877
horse	0.924313	**0.935324**
michael	**0.962636**	0.951064
victoria	**0.961154**	0.953213
wolf	0.982075	**0.991781**
SCAPE	**0.921698**	0.914941
FAUST	**0.855928**	0.822367

Table 5.1: *Comparison of Symmetry Aware Correspondence to BIM.*

be translated into a segment-level symmetry map by filtering out all the self-matching segments.

Note that in this application, we always match a shape graph to itself. This provides a guarantee that there is no slipping of segments, and that the matching is always performed on isometric graphs. Thus, the only possible errors in the matching are due to the structure of the shape graph. For example, if the branches of a leg and a tail are of the same length in the shape graph, they are considered symmetric even though the geometry of the shape suggests otherwise. These errors can happen when too few segments are used and they fail to capture the details of the shape, or when too many segments are used and the finer details of the shape are interfering with the symmetry detection. We find that for most shapes, the method is stable when the number of bins k is between 6 and 10. A higher number of bins is generally preferable for this application as it generates smaller segments.

A few examples of our symmetry detection are given in Figure 5.5. Note that our method is robust to isometric distortion, as visible particularly in the two shapes on the left. In the centaur shape, it can be seen that our method distinguishes well between similar elements with structural differences such as arms, hind legs and front legs. Finally, note that our method can produce segments which are small enough to indicate symmetry between feature points, as suggested in the next section.

Category	WKS	WKS + Segments	Only Segments
TOSCA			
cat	6.271	5.821	5.821
centaur	5.197	4.089	4.089
david	6.371	4.091	4.091
dog	7.409	4.394	4.398
horse	10.422	6.402	6.403
michael	6.845	5.465	5.465
victoria	5.169	3.850	3.853
wolf	1.414	1.206	1.204
Per-category average:	6.137	4.415	4.416
Per-shape average:	6.552	4.830	4.831
SCAPE	0.104	0.055	0.055

Table 5.2: *Evaluation of point-to-point maps. The numbers represent the average geodesic error (lower is better).*

5.4.5 Matching of feature points

Another possible application of our method is using the matching between symmetric segments to compute symmetry aware matching between feature points. This can be used for matching feature points between two shapes or for detecting symmetric feature points when matching a shape with itself. For example, in the recent work of [Kezurer *et al.*, 2015], between 10 and 20 feature points were matched for most shapes (due to the high complexity of the method, a higher number of feature points is generally not feasible). In our method, the shapes are typically segmented to between 15 and 40 segments. Since feature points are laid out rather uniformly over the shapes, it is likely that the segments produced by our method are such that each segment contains no more than a single feature point, at least for the majority of segments. These feature points can be matched directly according to the segments that contain them. In case a few feature points fall in the same segment, the matching between unique feature points can be used to resolve the ambiguity.

5.5 Conclusion

We present the concept of symmetry aware correspondence, in which we compute correspondence between shapes while factoring out the symmetric

components. We develop a method for computing symmetry aware correspondence between shape parts. The level of details in this mapping is not as high as in conventional correspondence methods which produce matching between feature points or point-to-point map. For the price of this compromise, we get the following advantages. Our method is extremely efficient, requiring a single extraction of an eigenvector. The method is computationally stable, and it is robust to various changes in the shape, such as differences in body types or extreme pose changes. We are also capable of matching shapes of different types with different geometry, as long as the structure of the shape is similar. It is also important to note that within the limited domain of matching shape segments, our method outperforms state-of-the-art shape correspondence methods.

We show that a point-to-point map can be improved by using our symmetric segments, even without breaking the symmetry. We also show an application of detecting segment-level symmetry in a shape. As a possible future application, transferring the segment-level correspondence to a correspondence between feature points is potentially quite simple, both for symmetry detection and matching between shapes.

An interesting direction for future development is breaking the symmetry. To this end, we propose a two-step solution, consisting of first finding the symmetry aware correspondence, and then computing a non-symmetric (i.e. one-to-one) correspondence while making sure parts in the same branch are matched consistently. State of the art correspondence techniques have known limitations when considering shapes with intrinsic symmetries, and this solution has the potential to be significantly less complicated than computing a non-symmetric matching in one shot, without prior information.

6 Conclusion

6.1 Summary of Contributions

In Chapter 2, we presented Dynamic Maps, a method for browsing and exploration of shapes, images, or any other collection of elements that can be represented by a thumbnail image. Using our method, thumbnails are laid out on a seemingly infinite grid which can be navigated in any direction like a standard map. The map is built dynamically in real-time after every user action, and the currently displayed patch is always continuous and smooth, i.e. near by images are similar to each other. This is unlike global mapping solutions which are bound to have discontinuities within the map. The local nature of the map also allows it to be created in a very efficient manner, and in constant time, regardless of the number of elements in the dataset. Thus, it is suitable for massive online collection of images. We implemented and evaluated the system for a dataset of several thousand 3D shapes and another dataset of one million images. The evaluations show that this method is generally preferable to common browsing methods such as relevance feedback (or "similar images" in modern search engines).

In Chapter 3, we proposed a method for learning a similarity measure for a collection of images from crowdsourced data. Once again, the method is applicable to any type of data that can be represented by a thumbnail image. The advantage of a crowd-based similarity measure is that it contains a lot of semantic information: it is directly derived from human perception of similarity. This work has two main contributions. First, the definition of a suitable question to ask the crowd. Since semantic similarity between elements is relative to the context, this is not a trivial task. We showed that clustering queries provide a lot of information and require a relatively small amount of effort from each crowd worker. Second, the development of an iterative technique that uses previous

answers to construct the most effective queries in the next phase. Using the iterative process, one can gather a lot of information from a relatively small number of queries.

In Chapter 4, we constructed an automatic shape similarity measure which is more semantic in nature than previous state-of-the-art methods. We defined the shape edit distance (or SHED) as a summary of all the transformations that a shape has to go through to become the other shape. To this end, our method segments the shape into parts, and compares the parts' geometry and position in order to find a matching between the two shapes. The matching between parts defines the transformations of each part, which are aggregated to form the shape edit distance. We evaluated our method on a variety of fundamental applications, and proved that it is more effective at capturing the semantic similarity between shapes, or a similarity measure which is closer to the human perception of similarity, than existing state-of-the-art shape similarity measures.

A notable additional contribution of this work is the method that computes the matching between shape parts. Existing methods for matching graphs or feature points usually focus on finding a one-to-one matching or one-to-many matching, and perform poorly for other correspondence structures. Since SHED requires a more complex correspondence structure, we developed *adaptive spectral matching* (see Section 4.3), an extension of spectral correspondence [Leordeanu and Hebert, 2005]. Adaptive spectral matching iteratively adapts the optimization to selected matches within the correspondence. The iterative process directs the optimization towards a solution which is more consistent with the structure of correspondence. Adaptive spectral matching can be used for one-to-one or one-to-many correspondences as well, and in most cases it outperforms the original spectral correspondence method of [Leordeanu and Hebert, 2005].

In Chapter 5, we delved deeper into the realm of correspondences between shapes and between shape parts. We proposed symmetry aware correspondence, a type of correspondence in which symmetric elements can be matched to groups of symmetric elements, without resolving the symmetry into a one-to-one matching. We developed a method that computes this symmetry in an extremely efficient manner. The output is a less detailed correspondence than one-to-one matching, but for this cost one gains greater accuracy, greater stability, and a much shorter computation time. This can be viewed as a decoupling of computing the correspondence and resolving the symmetry. After

the symmetric correspondence is found, it can potentially be resolved into a one-to-one matching as a post-process, instead of solving the two problems at the same time. Even without resolving the symmetry, the symmetry aware correspondence is useful. We showed that it can improve non-symmetric point-to-point correspondences using the functional maps framework [Ovsjanikov *et al.*, 2012]. In addition, our method can be used for segment-level symmetry detection or symmetry detection of feature points on the shape.

6.2 Future Directions

In recent years, image similarity measures have become more and more semantic in nature. This is due to advancements in deep learning and convolutional networks, as well as the high availability of textual context for images as training data [Krizhevsky *et al.*, 2012; Wang *et al.*, 2014; Szegedy *et al.*, 2015]. Deep learning methods for image retrieval require massive training sets with millions of element with some sort of ranking among them. Still, the ranking of training data is usually computed automatically by hand-crafted features. It would be interesting to combine this approach with a crowdsourcing technique for gathering image similarity. Our crowdsourcing solution can be used to measure similarities between key images in the dataset, while automatic features propagate the computed similarities to other images.

A key element of our crowdsourcing solution is the efficient query selection. This algorithm can also aid the computation of deep learning networks which require many queries. For example, in [Wang *et al.*, 2014], triplets were sampled based on a relevance score for computational reasons. Our clustering based queries may drastically reduce the necessary computation, thus allowing a larger portion of the training data to be used effectively.

While the image similarity ranking has greatly improved, the image browsing *experience* has not changed much in recent years. In the last decade, commercial image search providers have kept the same paradigm of keyword search with a rarely used relevance feedback feature (or "similar images"). Introducing a more flexible paradigm such as our Dynamic Maps into commercial image search can have a great effect on the quality of image search and in particular image browsing where the end result is not specific or known in advance. The Dynamic Maps framework is an initial step towards this goal. There are many directions for future research such as combining our method with keyword search, constructing

local patches in a non-greedy manner, and reflecting past user actions (rather than just the current state) in the choice of which images to display next. Another interesting direction for future development is using Dynamic Maps on a mobile device, for example for personal photo collections. Such collections typically have a more limited keyword support if any, and are large enough to be hard to manage or navigate using the common paradigm of a list ordered by the date the picture was taken.

In the shapes domain, leading similarity measures are far less advanced than in the image domain. One of the difficulties in shape analysis is the balance between local features and global context. For example, in a human shape, a cylindrical area could be a part of a finger, an arm or the torso, and global information is necessary to determine which of these options is correct. In SHED, we offer a such a balance by computing a shape graph of nearly convex parts [van Kaick *et al.*, 2014] The shape graph provides global context for the local geometric properties computed for each segment. Similarly, our Symmetry Aware Correspondence uses the global context of segments that are created using local features. In both SHED and Symmetry Aware Correspondence, failure cases are mostly a result of inconsistencies in the segmentation. Coarse segmentations tend to be more consistent but provide less information on the finer details of the shapes. Fine segmentations tend to be less consistent since small changes in the geometry may significantly change the shape of a segment or introduce additional segments. Thus, future work should carefully examine the balance between level of details in the shape and the consistency of the segmentation. A possible direction for future research is combining several representations with different level of details.

Shape similarity can also make use of deep learning from large amounts of tagged data, similarly to image similarity measures. Such methods are becoming popular recently (e.g. [Qi *et al.*, 2016]) but are not yet as refined as methods for image similarity. In these methods, the representation of the shape is a rendering or a volumetric rasterization of the shape into voxels. This representation mimics the 2D representation used for images, but it does not fit naturally with common sparse 3D shape representations such as triangular meshes or point clouds, and contains a lot of redundant information. An interesting question is whether the representation of a shape as a collection of segments or a shape graph can be used in a deep learning approach.

References

[Akgül *et al.*, 2010] Ceyhun Burak Akgül, Bülent Sankur, Yücel Yemez, and Francis Schmitt. Similarity learning for 3d object retrieval using relevance feedback and risk minimization. *Int. J. Comput. Vision*, 89:392–407, September 2010.

[André *et al.*, 2009] Paul André, Edward Cutrell, Desney S Tan, and Greg Smith. Designing novel image search interfaces by understanding unique characteristics and usage. In *IFIP Conference on Human-Computer Interaction*, pages 340–353. Springer, 2009.

[Anguelov *et al.*, 2005] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 408–416. ACM, 2005.

[Ankerst *et al.*, 1999] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *Proc. Int. Symp. Advances in Spatial Databases*, pages 207–226, 1999.

[Atmosukarto *et al.*, 2005] Indriyati Atmosukarto, Wee Kheng Leow, and Zhiyong Huang. Feature combination and relevance feedback for 3d model retrieval. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*, pages 334–339. IEEE, 2005.

[Averkiou *et al.*, 2014] Melinos Averkiou, Vladimir G. Kim, Youyi Zheng, and Niloy J. Mitra. ShapeSynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum (Eurographics)*, 33, 2014.

[Back and Oppenheim, 2001] Jonathan Back and Charles Oppenheim. A model of cognitive load for {IR}: implications for user relevance feedback interaction. *Information Research 2001*, 2001.

[Bar-Hillel *et al.*, 2005] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(6):937–965, 2005.

[Barra and Biasotti, 2013] Vincent Barra and Silvia Biasotti. 3D shape retrieval using kernels on extended Reeb graphs. *Pattern Recognition*, 46(11), 2013.

[Bederson, 2001] B.B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 71–80. ACM, 2001.

[Berg *et al.*, 2005] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. IEEE Conf. on CVPR*, pages 26–33, 2005.

[Biswas and Jacobs, 2014] Arijit Biswas and David Jacobs. Active image clustering with pairwise constraints from humans. *International Journal of Computer Vision*, 108(1-2):133–147, 2014.

[Bogo *et al.*, 2014] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE.

[Bommes *et al.*, 2012] David Bommes, Henrik Zimmer, and Leif Kobbelt. Practical mixed-integer optimization for geometry processing. In *Curves and Surfaces*, pages 193–206. Springer, 2012.

[Bronstein *et al.*, 2008] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[Bronstein *et al.*, 2011] Alexander M Bronstein, Michael M Bronstein, Leonidas J Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions

for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1, 2011.

[Brooke, 1996] John Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189:194, 1996.

[Cao *et al.*, 2006] Liangliang Cao, Jianzhuang Liu, and Xiaoou Tang. 3d object retrieval using 2d line drawing and graph based relevance reedback. In *Proceedings of the 14th annual ACM international conference on Multimedia*, MULTIMEDIA '06, pages 105–108, New York, NY, USA, 2006. ACM.

[Chang *et al.*, 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[Chaudhuri and Koltun, 2010] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3d modeling. In *ACM Transactions on Graphics (TOG)*, volume 29, page 183. ACM, 2010.

[Chen *et al.*, 2000] Chaomei Chen, George Gagaudakis, and Paul Rosin. Similarity-based image browsing. In *Int. conference on intelligent information processing*, pages 206–213. Citeseer, 2000.

[Chen *et al.*, 2003] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3D model retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003.

[Chen *et al.*, 2009] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Trans. on Graph (SIGGRAPH)*, 28(3):73:1–12, 2009.

[Chew *et al.*, 2010] Boon Chew, Jennifer A Rode, and Abigail Sellen. Understanding the everyday use of images on the web. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 102–111. ACM, 2010.

[Chung and Yoon, 2011] EunKyung Chung and JungWon Yoon. Image needs in the context of image use: An exploratory study. *Journal of Information Science*, page 0165551511400951, 2011.

[Coifman and Lafon, 2006] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[Combs and Bederson, 1999] Tammara TA Combs and Benjamin B Bederson. Does zooming improve image browsing? In *Proceedings of the fourth ACM conference on Digital libraries*, pages 130–137. ACM, 1999.

[Cour *et al.*, 2006] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. Balanced graph matching. In *Advances in Neural Information Processing Systems*, volume 19, pages 313–320, 2006.

[Croft *et al.*, 2001] W.B. Croft, S. Cronen-Townsend, and V. Lavrenko. Relevance feedback and personalization: A language modeling perspective. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[Dalal and Triggs, 2005] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005.*, volume 1, pages 886–893. IEEE, 2005.

[Davidson *et al.*, 2013] Susan B Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. Using the crowd for top-k and group-by queries. In *International Conference on Database Theory*, pages 225–236. ACM, 2013.

[Denning and Pellacini, 2013] Jonathan D. Denning and Fabio Pellacini. MeshGit: Diffing and merging meshes for polygonal modeling. *ACM Trans. on Graph (SIGGRAPH)*, 32(4):35:1–10, 2013.

[Deselaers *et al.*, 2008] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

[Elad *et al.*, 2002] Michael Elad, Ayellet Tal, and Sigal Ar. Content based retrieval of vrml objects: an iterative and interactive approach. In *Multimedia 2001*, pages 107–118. Springer, 2002.

[Fan *et al.*, 2009] Jianping Fan, Daniel A Keim, Yuli Gao, Hangzai Luo, and Zongmin Li. Justclick: personalized image recommendation via exploratory search from large-scale flickr images. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(2):273–288, 2009.

[Fisher *et al.*, 2011] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. on Graph (SIGGRAPH)*, 30(4):34:1–12, 2011.

[Frome *et al.*, 2007] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[Funkhouser *et al.*, 2004] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. on Graph (SIGGRAPH)*, 23(3):652–663, 2004.

[Ganapathi-Subramanian *et al.*, 2016] Vignesh Ganapathi-Subramanian, Boris Thibert, Maks Ovsjanikov, and Leonidas Guibas. Stable region correspondences between non-isometric shapes. *Computer Graphics Forum (SGP)*, 2016.

[Gao *et al.*, 2010] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, 2010.

[Gomes *et al.*, 2011] Ryan G Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In *Advances in neural information processing systems*, pages 558–566, 2011.

[Harchaoui and Bach, 2007] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *Proc. IEEE Conf. on CVPR*, pages 1–8, 2007.

[Hearst, 2009] Marti Hearst. *Search user interfaces*. Cambridge University Press, 2009.

[Hilaga *et al.*, 2001] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc.SIGGRAPH*, pages 203–212, 2001.

[Hu *et al.*, 1999] Paul Jen-Hwa Hu, Pai-Chun Ma, and Patrick YK Chau. Evaluation of user interface designs for information retrieval systems: a computer-based experiment. *Decision support systems*, 27(1):125–143, 1999.

[Huang *et al.*, 2011] Qixing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. *ACM Trans. on Graph (SIGGRAPH Asia)*, 30(6):125:1–12, 2011.

[Huang *et al.*, 2013a] Qi-Xing Huang, Hao Su, and Leonidas Guibas. Fine-grained semi-supervised labeling of large shape collections. *ACM Transactions on Graphics (TOG)*, 32(6):190, 2013.

[Huang *et al.*, 2013b] Shi-Sheng Huang, Ariel Shamir, Chao-Hui Shen, Hao Zhang, Alla Sheffer, Shi-Min Hu, and Daniel Cohen-Or. Qualitative organization of collections of shapes via quartet analysis. *ACM Trans. on Graph (SIGGRAPH)*, 32(4):71:1–10, 2013.

[Huck *et al.*, 1974] Schuyler W Huck, William Henry Cormier, and William G Bounds. *Reading statistics and research*. Harper & Row New York, 1974.

[Jing *et al.*, 2012] Yushi Jing, Henry Rowley, Jingbin Wang, David Tsai, Chuck Rosenberg, and Michele Covell. Google image swirl: a large-scale content-based image visualization system. In *Proceedings of the 21st International Conference on World Wide Web*, pages 539–540. ACM, 2012.

[Kalogerakis *et al.*, 2012] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model of component-based shape synthesis. *ACM Trans. on Graph (SIGGRAPH)*, 31(4):55:1–11, 2012.

[Kazhdan *et al.*, 2003] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symp. on Geom. Proc.*, pages 156–164, 2003.

[Kazhdan *et al.*, 2004] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Symmetry descriptors and 3d shape matching. In *Proceedings*

of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 115–123. ACM, 2004.

[Kezurer *et al.*, 2015] Itay Kezurer, Shahar Z Kovalsky, Ronen Basri, and Yaron Lipman. Tight relaxation of quadratic matching. *Computer Graphics Forum*, 24(5), 2015.

[Kim *et al.*, 2011] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30, page 79. ACM, 2011.

[Kim *et al.*, 2012] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring collections of 3D models using fuzzy correspondences. *ACM Trans. on Graph (SIGGRAPH)*, 31(4):54:1–11, 2012.

[Kim *et al.*, 2013] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3D shapes. *ACM Trans. on Graph (SIGGRAPH)*, 32(4):70:1–12, 2013.

[Kleiman *et al.*, 2013] Yanir Kleiman, Noa Fish, Joel Lanir, and Daniel Cohen-Or. Dynamic maps for exploring and browsing shapes. *Computer Graphics Forum (SGP)*, 32(5):187–196, 2013.

[Kleiman *et al.*, 2015a] Yanir Kleiman, Joel Lanir, Dov Danon, Yasmin Felberbaum, and Daniel Cohen-Or. Dynamicmaps: Similarity-based browsing through a massive set of images. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 995–1004. ACM, 2015.

[Kleiman *et al.*, 2015b] Yanir Kleiman, Oliver van Kaick, Olga Sorkine-Hornung, and Daniel Cohen-Or. Shed: shape edit distance for fine-grained shape similarity. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 34(6):235, 2015.

[Kleiman *et al.*, 2016] Yanir Kleiman, George Goldberg, Yael Amsterdamer, and Daniel Cohen-Or. Toward semantic image similarity from crowdsourced clustering. *The Visual Computer*, 32(6):1045–1055, 2016.

[Kohonen, 1990] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kruskal and Wish, 1978] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[Kurtek *et al.*, 2013] Sebastian Kurtek, Anuj Srivastava, Eric Klassen, and Hamid Laga. Landmark-guided elastic shape analysis of spherically-parameterized surfaces. *Computer Graphics Forum*, 32(2pt4):429–438, 2013.

[Laga *et al.*, 2013] Hamid Laga, Michela Mortara, and Michela Spagnuolo. Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes. *ACM Trans. on Graph*, 32(5):150:1–16, 2013.

[Lasram *et al.*, 2012] A. Lasram, S. Lefebvre, and C. Damez. Procedural texture preview. In *Computer Graphics Forum*, volume 31, pages 413–420. Wiley Online Library, 2012.

[Layne, 1994] Sara Shatford Layne. Some issues in the indexing of images. *Journal of the American Society for Information Science (1986-1998)*, 45(8):583, 1994.

[Leifman *et al.*, 2005] George Leifman, Ron Meir, and Ayellet Tal. Semantic-oriented 3d shape retrieval using relevance feedback. *The Visual Computer*, 21(8-10):865–875, 2005.

[Leordeanu and Hebert, 2005] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the International Conference on Computer Vision*, pages 1482–1489, 2005.

[Leordeanu *et al.*, 2009] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and map inference. In *Advances in neural information processing systems*, pages 1114–1122, 2009.

[Li *et al.*, 2012a] B Li, A Godil, M Aono, X Bai, T Furuya, L Li, R López-Sastre, H Johan, R Ohbuchi, C Redondo-Cabrera, et al. Shrec'12 track: Generic 3d

shape retrieval. In *Proceedings of the 5th Eurographics conference on 3D Object Retrieval*, pages 119–126. Eurographics Association, 2012.

[Li *et al.*, 2012b] Congcong Li, D. Parikh, and Tsuhan Chen. Automatic discovery of groups of objects for scene understanding. In *Proc. IEEE Conf. on CVPR*, pages 2735–2742, 2012.

[Lipman *et al.*, 2010] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. In *ACM Transactions on Graphics (TOG)*, volume 29, page 103. ACM, 2010.

[Litman *et al.*, 2014] Roee Litman, Alex Bronstein, Michael Bronstein, and Umberto Castellani. Supervised learning of bag-of-features shape descriptors using sparse coding. *Computer Graphics Forum*, 33(5):127–136, 2014.

[Liu *et al.*, 2004] H. Liu, X. Xie, X. Tang, Z.W. Li, and W.Y. Ma. Effective browsing of web image search results. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 84–90. ACM, 2004.

[Lowe, 1999] David G Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer vision, 1999.*, volume 2, pages 1150–1157. Ieee, 1999.

[Lun *et al.*, 2015] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: learning perceptual shape style similarity. *ACM Transactions on Graphics (TOG)*, 34(4):84, 2015.

[Lyzinski *et al.*, 2015] Vince Lyzinski, Donniell Fishkind, Marcelo Fiori, Joshua Vogelstein, Carey Priebe, and Guillermo Sapiro. Graph matching: relax at your own risk. *IEEE TPAMI*, 2015.

[Marcus *et al.*, 2011] Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *Proceedings of the VLDB Endowment*, 5(1):13–24, 2011.

[Markkula and Sormunen, 2000] Marjo Markkula and Eero Sormunen. End-user searching challenges indexing practices in the digital newspaper photo archive. *Information retrieval*, 1(4):259–285, 2000.

[Meyer *et al.*, 2002] Mark Meyer, Mathieu Desbrun, Peter Schröder, Alan H Barr, et al. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics*, 3(2):52–58, 2002.

[Mitra *et al.*, 2013] Niloy J. Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, and Martin Bokeloh. Structure-aware shape processing. In *Proc. Eurographics State-of-the-art Reports*, 2013.

[Neuhaus and Bunke, 2007] Michel Neuhaus and Horst Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific, River Edge, NJ, USA, 2007.

[Novotni and Klein, 2003] Marcin Novotni and Reinhard Klein. 3d zernike descriptors for content based shape retrieval. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 216–225. ACM, 2003.

[O'Donovan *et al.*, 2014] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Transactions on Graphics (TOG)*, 33(4):92, 2014.

[Oliva and Torralba, 2001] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[Osada *et al.*, 2002] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. on Graph*, 21(4):807–832, 2002.

[Ovsjanikov *et al.*, 2010] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas. One point isometric matching with the heat kernel. *Computer Graphics Forum (SGP)*, 29(5):1555–1564, 2010.

[Ovsjanikov *et al.*, 2011] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM Trans. on Graph (SIGGRAPH)*, 30(4):33:1–10, 2011.

[Ovsjanikov *et al.*, 2012] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):30, 2012.

[Ovsjanikov *et al.*, 2013] Maks Ovsjanikov, Quentin Mérigot, Viorica Pătrăucean, and Leonidas Guibas. Shape matching via quotient spaces. In *Computer Graphics Forum*, volume 32, pages 1–11. Wiley Online Library, 2013.

[Pečenović *et al.*, 2000] Zoran Pečenović, Minh N Do, Martin Vetterli, and Pearl Pu. Integrated browsing and searching of large image collections. In *International Conference on Advances in Visual Information Systems*, pages 279–289. Springer, 2000.

[Qi *et al.*, 2016] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016.

[Rodden *et al.*, 1999] Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Evaluating a visualisation of image similarity as a tool for image browsing. In *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pages 36–43. IEEE, 1999.

[Rodden *et al.*, 2001] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 190–197. ACM, 2001.

[Roweis and Saul, 2000] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[Rui *et al.*, 1998] Y. Rui, T.S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):644–655, 1998.

[Rustamov, 2007] Raif M. Rustamov. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Symp. on Geom. Proc.*, pages 225–233, 2007.

[Ruthven and Lalmas, 2003] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(02):95–145, 2003.

[Sahillioğlu and Yemez, 2011] Y Sahillioğlu and Yücel Yemez. Coarse-to-fine combinatorial matching for dense isometric shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1461–1470. Wiley Online Library, 2011.

[Sakamoto *et al.*, 2004] Yasuhiko Sakamoto, Shigeru Kuriyama, and Toyohisa Kaneko. Motion map: image-based retrieval and segmentation of motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 259–266, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[Saleh *et al.*, 2015] Babak Saleh, Mira Dontcheva, Aaron Hertzmann, and Zhicheng Liu. Learning style similarity for searching infographics. In *Graphics Interface Conference*, pages 59–64. Canadian Information Processing Society, 2015.

[Sammon, 1969] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 1969.

[Schultz and Joachims, 2004] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *Advances in neural information processing systems (NIPS)*, page 41, 2004.

[Sebastian *et al.*, 2004] T.B. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Trans. Pat. Ana. & Mach. Int.*, 26(5):550–571, 2004.

[Shamir, 2008] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[Shapira *et al.*, 2008] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008.

[Shapira *et al.*, 2009] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Image appearance exploration by model-based navigation. In *Computer Graphics Forum*, volume 28, pages 629–638. Wiley Online Library, 2009.

[Shi and Malik, 2000] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[Shilane *et al.*, 2004] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape Modeling Applications, 2004. Proceedings*, pages 167–178. IEEE, 2004.

[Sidi *et al.*, 2011] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. on Graph (SIGGRAPH Asia)*, 30(6):126:1–10, 2011.

[Singh *et al.*, 2007] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.

[Sivic and Zisserman, 2003] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision, 2003.*, pages 1470–1477. IEEE, 2003.

[Strong and Gong, 2008] Grant Strong and Minglun Gong. Browsing a large collection of community photos based on similarity on gpu. In *International Symposium on Visual Computing*, pages 390–399. Springer, 2008.

[Strong *et al.*, 2010] Grant Strong, Orland Hoeber, and Minglun Gong. Visual image browsing and exploration (vibe): User evaluations of image search tasks. In *International Conference on Active Media Technology*, pages 424–435. Springer, 2010.

[Suditu and Fleuret, 2011] Nicolae Suditu and Francois Fleuret. Heat: Iterative relevance feedback with one million images. In *International Conference on Computer Vision*, October 2011.

[Sun *et al.*, 2009] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[Sundar *et al.*, 2003] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International*, pages 130–139, 2003.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[Talton *et al.*, 2009] Jerry O Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics-TOG*, 28(5):167, 2009.

[Tamuz *et al.*, 2011] Omer Tamuz, Ce Liu, Ohad Shamir, Adam Kalai, and Serge J. Belongie. Adaptively learning the crowd kernel. In *International Conference on Machine Learning (ICML-11)*, pages 673–680. ACM, 2011.

[Tangelder and Veltkamp, 2004] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings*, pages 145–156. IEEE, 2004.

[Tangelder and Veltkamp, 2008] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008.

[Umetani *et al.*, 2012] Nobuyuki Umetani, Takeo Igarashi, and Niloy J Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics*, 31(4), 2012.

[Van Kaick *et al.*, 2011] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011.

[van Kaick *et al.*, 2014] Oliver van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or. Shape segmentation by approximate convexity analysis. *ACM Trans. Graph.*, 34(1):4, 2014.

[Vanamali *et al.*, 2010] TP Vanamali, A Godil, H Dutagaci, T Furuya, Z Lian, and R Ohbuchi. Shrec'10 track: Generic 3d warehouse. In *Proceedings of the 3rd Eurographics conference on 3D Object Retrieval*, pages 93–100. Eurographics Association, 2010.

[Vieira *et al.*, 2009] Thales Vieira, Alex Bordignon, Adelailson Peixoto, Geovan Tavares, Hélio Lopes, Luiz Velho, and Thomas Lewiner. Learning good views through intelligent galleries. In *Computer Graphics Forum*, volume 28, pages 717–726. Wiley Online Library, 2009.

[Wang *et al.*, 2009] Chong Wang, David Blei, and Fei-Fei Li. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition, 2009.*, pages 1903–1910. IEEE, 2009.

[Wang *et al.*, 2012] Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):165, 2012.

[Wang *et al.*, 2014] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[Weinberger *et al.*, 2005] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2005.

[Wilber *et al.*, 2014] Michael J Wilber, Iljung S Kwak, and Serge J Belongie. Cost-effective hits for relative similarity comparisons. In *Conference on Human Computation and Crowdsourcing*, 2014.

[Xing *et al.*, 2003] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2003.

[Xu *et al.*, 2010] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhiquan Cheng. Style-content separation by anisotropic part scales. *ACM Trans. on Graph (SIGGRAPH Asia)*, 29(6), 2010.

[Yang *et al.*, 2011] Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J Mitra. Shape space exploration of constrained meshes. *ACM Trans. Graph*, 30(124):1–124, 2011.

[Yee *et al.*, 2003] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408. ACM, 2003.

[Yi *et al.*, 2012] Jinfeng Yi, Rong Jin, Shaili Jain, Tianbao Yang, and Anil K Jain. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *Advances in Neural Information Processing Systems*, pages 1772–1780, 2012.

[Zelnik-Manor and Perona, 2004] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, volume 17, pages 1601–1608, 2004.

[Zha *et al.*, 2008] Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. Joint multi-label multi-instance learning for image classification. In *Computer Vision and Pattern Recognition, 2008.*, pages 1–8. IEEE, 2008.

[Zhang *et al.*, 2001] Lei Zhang, Fuzong Lin, and Bo Zhang. Support vector machine learning for image retrieval. In *Proc. IEEE Int. Conf. on Image Processing*, pages 721–724, 2001.

[Zheng *et al.*, 2014] Youyi Zheng, Daniel Cohen-Or, Melinos Averkiou, and Niloy J. Mitra. Recurring part arrangements in shape collections. *Computer Graphics Forum (Eurographics)*, 33, 2014.

[Zhou and Huang, 2003] X.S. Zhou and T.S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 8(6):536–544, 2003.

התאמה סימטרית בין חלקי הצורות השונות. באופן אינטואיטיבי, ההתאמה הסימטרית מאפשרת לכל חלק להתאים לכמות כלשהי של חלקים סימטריים בצורה השניה. בפועל, קבוצות בגדלים שונים של חלקים בצורה אחת מותאמות לקבוצות של חלקים בקבוצה השניה, כך שכל חלק בקבוצה הראשונה מותאם לכל חלק בקבוצה השניה. קבוצות אלו מתארות את אוסף הסימטריות של החלקים בכל צורה.

התאמה של חלקים הכוללים סימטריות פנימיות (intrinsic symmetries) היא בעיית התאמה קשה במיוחד, מאחר ובהכרח יש כמה פתרונות סבירים במידה שווה. פתרון אופטימיזציה מסוג זה הוא יקר ומועד לשגיאות, ולכן במקרים רבים מתקבל פתרון פחות מדויק מן הרצוי. ההתאמה הסימטרית מרפה את האילוץ למציאת התאמה חד-חד ערכית, ובפועל מכווצת את מרחב החיפוש כך שקיים רק פתרון יציב אחד לבעיה, שניתן למצוא באופן מאוד מהיר ויעיל.

יש לשים לב שעבור התאמה של רבים לרבים (כפי שנדרשת כאן), פתרון המתאים כל חלק בצורה אחת לכל חלק בצורה השניה הוא תקין ומותר על פי האילוצים, אם כי לא רצוי. זאת בניגוד להתאמות חד-חד ערכיות או של אחד לרבים, שלא מאפשרות פתרונות מנוונים מסוג זה. אם פתרון האופטימיזציה אינו דליל (sparse), לא ניתן לקבוע בצורה מהימנה כמה מן ההתאמות בין חלקים יש לכלול בפתרון. לכן אנו מציעים אלגוריתם המבוסס על התאמה ספקטרלית שמבטיח פתרון דליל שבו ערכים גבוהים מופיעים רק עבור התאמות ששייכות לפתרון הכולל.

מציאת התאמה חד-חד ערכית כאשר נתונה התאמה סימטרית אינה טריוויאלית וידרשו לשם כך צעדים נוספים. אף על פי כן, אנו מראים כי ישנם שימושים להתאמה הסימטרית גם ללא מציאת התאמה חד-חד ערכית. בפרט, אנחנו מראים שניתן להשתמש בהתאמה הסימטרית על מנת לשפר התאמה צפופה בין הצורות (שהיא חד-חד ערכית מטבעה). כמו כן, ניתן להשתמש בהתאמה סימטרית על מנת לזהות סימטריה פנימית בצורה ברמת המקטעים. ברוב המקרים, ניתן לתרגם באופן מיידי התאמה סימטרית בין מקטעים להתאמה סימטרית בין נקודות עניין בצורה, מאחר ואנו משתמשים במספר גבוה של מקטעים שלרוב לא משויכת אליהן יותר מנקודה אחת. בעוד ההתאמה הסימטרית שאנו מוצאים היא פחות פרטנית מהתאמות חד-חד ערכיות, הפתרון שלנו יעיל יותר ומהיר יותר, וההתאמות ברמת המקטע מדויקות יותר מהתאמות שנמצאו על ידי שיטות אשר מוצאות התאמה חד-חד ערכית.

תוצאות פרק זה טרם פורסמו בעת הגשת המסמך.

לחלקים כמעט-קמורים [Van Kaick et al. 2014], ומוצאים התאמה בין החלקים של הצורות השונות. על פי ההתאמה אנחנו מעריבים את פעולות העריבה הנדרשות על מנת להעתיק כל חלק אל החלק התואם לו בצורה השניה. לכן, שיטה זו מסוגלת לזהות צורות בעלות אותו סגנון או אופן שימוש. מרחק עריבת הצורה מהווה מידת דמיון אינטואיטיבית ויחסית קרובה לתפיסה אנושית של דמיון בין צורות. לכן הוא שימושי כדי לזהות צורות דומות בתוך אותה הקטגוריה, בנוסף ליכולתו להבדיל בין צורות השייכות לקטגוריות שונות.

חלק מרכזי בשיטה הוא חישוב ההתאמה בין חלקי הצורות. התאמה בין גרפים או בין נקודות עניין הוא נושא שנחקר רבות בשנים האחרונות, אך לרוב לצורך מציאת התאמה חד-חד ערכית או התאמה של אחד לרבים [Leordeanu and Hebert 2005, Berg et al. 2005, Kezurer et al. 2015, Cour *et al.*, 2006; Leordeanu *et al.*, 2009]. שיטות אלו פחות אפקטיביות עבור מבנה התאמה מורכב יותר, או שהן בכלל אינן תומכות בסוגי התאמה אחרים. עבור מרחק עריכת הצורה, נדרשת התאמה אחד-לרבים דו-כיוונית: כל חלק בכל אחת מהצורות יכול להתאים למספר חלקים בצורה השניה, אבל יחסי רבים-לרבים אינם מותרים. מאחר והאילוצים גמישים יחסית, ישנם בדרך כלל מספר פתרונות מתחרים וסותרים שמתאימים בסבירות גבוהה לאילוצים. במקרה כזה חלק מההתאמות בין החלקים עשויות להילקח מפתרון אחד בעוד חלק אחר מן ההתאמות נלקח מפתרון אחר וסותר, וכך מתקבלת התאמה שאינה עקבית. כפתרון, אנחנו מציעים אלגוריתם *התאמה ספקטרלית אדפטיבי (adaptive spectral matching)*, אשר מרחיב את הפתרון הפופולרי של התאמה ספקטרלית [Leordeanu and Hebert, 2005]. אלגוריתם זה מתקן את פונקציית האופטימיזציה באופן איטרטיבי בהתאם להתאמות שנבחרו בשלבים מוקדמים יותר. האלגוריתם האדפטיבי נותן עדיפות להתאמות ששייכות לאותו פתרון קוהרנטי.

תוצאות פרק זה פורסמו במאמר הבא:

Shed: shape edit distance for fine-grained shape similarity.

Yanir Kleiman, Oliver van Kaick, Olga Sorkine-Hornung, and Daniel Cohen-Or.

ACM Transactions on Graphics (SIGGRAPH Asia), 2015.

5 התאמה סימטרית באמצעות גרף חלקי הצורה

בפרק זה נחקור רעיון דומה של חלוקת הצורה למקטעים ומציאת התאמה בין המקטעים. הפעם נשתמש בטכניקה זו בכדי למצוא התאמות צפופות בין צורות ובכדי לזהות סימטריה פנימית של צורות. התאמה צפופה (dense או point-to-point correspondence) היא התאמה שבה עבור כל נקודה בצורה מוגדרת נקודת יעד בצורה השניה. לאחר חלוקת הצורה למקטעים באופן פרמטרי (כלומר, החלוקה היא לא בהכרח לכדי חלקים סמנטיים), אנו יוצרים גרף המתאר את חלקי הצורה (shape graph), שבו כל צומת מתאר מקטע ובין כל שני מקטעים סמוכים קיימת קשת. גרף חלקי הצורה מספק מידע על מבנה הצורה, שניתן להשתמש בו על מנת לייצב את ההתאמה ולשלול התאמות שאינן תואמות לשאר ההתאמות בין חלקי הצורה. בפרט, אנחנו משתמשים בגרף חלקי הצורה בכדי למצוא

מייצרים שאילתות נוספות בהתבסס על מיקום האובייקטים במרחב ההטמעה, והתהליך חוזר על עצמו עד לניצול תקציב השאילתות. אבחנה חשובה היא ששאילתות הכוללות אובייקטים הקרובים זה לזה הן בדרך כלל מדויקות יותר (למשל, קל לזהות תמונות דומות מאוד, אך קשה להעריך איזו מבין שתי תמונות קרובה יותר לתמונה שלישית כאשר כל השלוש שונות מאוד זו מזו). כמו כן, שחזור המרחקים בין כל האובייקטים בהינתן המרחקים הקצרים ביותר הוא יותר אמין משחזור המסתמך על מרחקים ארוכים. לכן, אנו דואגים בכל שאילתה לדגום תמונות שאמורות להיות קרובות זו לזו ככל הידוע לנו עד כה. לשם כך אנחנו בוחרים תמונה כלשהי במאגר ואז בוחרים n תמונות מתוך סביבתה הקרובה במרחב ההטמעה. בשלבים ראשונים של התהליך תמונות אלו יהיו רחוקות יחסית זו מזו, אך ככל שנוספות שאילתות מיקום האובייקטים במרחב ההטמעה מדויק יותר והמרחקים בין התמונות הנבחרות הולכים ומתקצרים.

ניתן לשלב בין מידת הדמיון שמחושבת באמצעות מיקור ההמונים ומדידות דמיון המבוססות על מאפייני התמונה\צורה. לדוגמה, ניתן לבחור תמונות מייצגות מתוך מאגר גדול, לחשב עבורן את הדמיון הסמנטי ולפעפע אותו אל שאר התמונות במאגר באמצעות מטריקות דמיון אוטומטיות.

תוצאות פרק זה פורסמו במאמר הבא:

Toward semantic image similarity from crowdsourced clustering.

Yanir Kleiman, George Goldberg, Yael Amsterdamer, and Daniel Cohen-Or.

The Visual Computer, 2016.

4 דמיון בין צורות באמצעות מרחק עריכת הצורה

בשני הפרקים הבאים נתמקד בצורות תלת מימדיות. מידות מרחק קיימות התרכזו עד כה בעיקר באבחנה בין צורות בקטגוריות שונות, או זיהוי צורות השייכות לאותה הקטגוריה, אך לא ייחסו חשיבות לדמיון הצורות בתוך אותה הקטגוריה. השיטות העדכניות ביותר מבוססות על מראה הצורה הכולל ולכן משקפות רק דמיון חסר הקשר סמנטי. שיטות אלו אינן משקפות דמיון בין צורות המונחות בפוזה אחרת, צורות שיש ביניהן דמיון חלקי (למשל כאשר חלק גדול מהצורה הוסר), או צורות שנערכו בהן שינויים ששינו את הפרופורציות בצורה (למשל מתיחה או כיווץ של חלקים). הן חסרות את ההקשר הסמנטי הנדרש על מנת לזהות צורות בעלות סגנון דומה או דרך שימוש דומה.

אנו מציגים את *מרחק עריכת הצורה (Shape Edit Distance)*, מידת דמיון בין צורות תלת מימדיות אשר מבחינה בדמיון בפרטים עדינים יותר של הצורה, בנוסף לדמיון מבני כולל בין הצורות. בפרט, מרחק עריכת הצורה מזהה דמיון בין צורות שעברו עריכה כגון הסרה, סיבוב, מתיחה או כיווץ של חלקים. בשם כך הוא מספק מידת דמיון סמנטית יותר מאשר שיטות מקבילות, שמשקפת במידה רבה יותר את הדמיון בין צורות כפי שהוא נתפס על ידי בני אדם. מטרתנו היא למדוד את כמות המאמץ הנדרשת בכדי להפוך צורה אחת לצורה השניה, על ידי ביצוע פעולות עריכה על חלקי הצורה. לצורך כך, אנחנו מחלקים כל צורה

במאגר, כך שניתן להשתמש במפות הדינמיות עבור כל מאגר של אובייקטים המיוצגים על ידי תמונות ובכל מידת דמיון שנבחר. לדוגמה, ניתן לשוטט במאגר של תמונות פנים באמצעות מידת דמיון שמבוססת על מאפיינים ספציפיים של זיהוי תמונה, או במאגר תמונות רפואיות על בסיס ניתוח רפואי של הנתונים. בעבודה זו מימשנו מפות דינמיות עבור מאגר של כ 4,500 אובייקטים תלת מימדיים ועבור מאגר של מיליון תמונות. כמו כן ערכנו סקרים מקיפים הבוחנים בין השאר את מידת שביעות הרצון של המשתמשים בעת ביצוע משימות שונות באמצעות מפות דינמיות לעומת שיטות חיפוש סטנדרטיות. התוצאות המלאות מופיעות בפרק 2 של החלק הלועזי של עבודה זו.

תוצאות פרק זה פורסמו במאמרים הבאים:

Dynamic maps for exploring and browsing shapes.

Yanir Kleiman, Noa Fish, Joel Lanir, and Daniel Cohen-Or.

Computer Graphics Forum (SGP), 2013.

DynamicMaps: Similarity-based browsing through a massive set of images.

Yanir Kleiman, Joel Lanir, Dov Danon, Yasmin Felberbaum, and Daniel Cohen-Or.

In *Proceedings of the SIGCHI conference on Human factors in computing systems*, *2015.*

3 דמיון בין תמונות באמצעות מיקור המונים

בפרק זה אנו מציגים שיטה לחישוב דמיון בין תמונות שהוא סמנטי לחלוטין, כלומר נובע מהקשר התמונה, כמו גם ממידע חיצוני לתמונה ואפילו הרגש שהתמונה מעוררת. כמו בפרק הקודם, גם כאן התמונה יכולה לייצג אובייקט כלשהו, כדוגמת צורות תלת מימדיות. הקשר סמנטי כאמור לעיל לרוב לא ניתן לזיהוי על ידי כלים אוטומטיים. תחת זאת, אנו מציעים לאסוף מידע מן הקהל על ידי שימוש בטכניקות של מיקור המונים. המטרה היא להתכנס במהירות יחסית לעבר מידת דמיון מדויקת ככל שניתן תוך כדי מזעור העלות, שנגזרת ממספר השאילתות הנדרש והסיבוכיות של כל שאילתה. ישנם שני אתגרים מרכזיים בפיתוח שיטה מסוג זה: ראשית, כיצד ניתן להשוות בין תמונות בצורה יעילה ומועילה, כלומר איזה סוג שאילתות רצוי לשלוח לקהל. שנית, איך כדאי לבחור את התמונות בכל שאילתה, וכיצד ניתן להשתמש בידע שהצטבר משאילתות קודמות על מנת לבנות שאילתות מועילות יותר.

אנו מספקים תשובות לשתי שאלות אלו, ומציגים שיטה שמבוססת על שאילתות חלוקה לקבוצות (clustering). על הקהל מוטלת המשימה לחלק אוסף של n תמונות ל k קבוצות. תוצאות החלוקה משמשות לשיפור הדרגתי של מידת הדמיון על ידי הטמעה של האובייקטים (embedding) במרחב ממימד נמוך. הטמעה זו פותרת קונפליקטים בתוצאות השאילתות ומאחדת את המידע כך שנוצרת מידה עקבית וקוהרנטית. לאחר מכן אנו

נקודה בצורה). אנו מגדירים התאמה מסוג חדש אשר נקרא לה *התאמה סימטרית (symmetry aware correspondence)*. בהתאמה זו כל מקטע יכול להתאים לקבוצת מקטעים סימטריים בצורה השניה. כמו כן אנו מציגים שיטה יעילה ויציבה למציאת התאמות מסוג זה, שפותרות את אי היציבות האופיינית לחישוב התאמות בין צורות סימטריות. אנו מראים שהתאמות סימטריות בין המקטעים יכולות לשמש לשיפור התאמות צפופות בין צורות ללא צורך במציאת התאמה חד-חד ערכית בין המקטעים.

בבדי למצוא התאמות בין מקטעים, יש לפתור למעשה בעיית התאמה בין גרפים. התאמה בין גרפים היא נושא רחב שנערך בו מחקר רב, אך רובן של העבודות מתרכזות בהתאמות חד-חד ערכיות או התאמות של אחד-לרבים. עבור השיטות שתוארו לעיל, נדרש מבנה התאמה מורכב יותר. לדוגמה, עבור חישוב מרחק העריבה בין צורות, נדרשת התאמה שבה כל חלק (בשתי הצורות) עשוי להתאים לכמה חלקים בצורה השניה, אך יחסים מסוג רבים-לרבים אינם מותרים. לשם כך, עבודות אלו כוללות תרומות ראויות לציון לפתרון בעיית ההתאמה בין גרפים: אנו מציגים שני עדכונים של שיטת התאמה ספקטרלית (spectral correspondence), שהיא שיטת נפוצה לחישוב התאמות בין גרפים. אלגוריתמים אלו מפורטים בחלקים 4.3 ו 5.3 של החלק הלועזי של עבודה זו.

2 שיטוט מבוסס דמיון על ידי מפות דינמיות

בפרק זה אנו מציגים *מפות דינמיות (Dynamic Maps)*, שיטה לסקירת מאגרי תמונות שמתבססת על הדמיון בין האובייקטים במאגר. הקלט לשיטה הינו מרחקים בין האובייקטים ותמונות מייצגות של האובייקטים במאגר, בין אם מדובר בתמונות, צורות תלת מימדיות, או כל אובייקט אחר אשר ניתן לייצג ע"י תמונה. התמונות המייצגות מסודרות במפה דו מימדית אינסופית על גבי רשת סדורה (כלומר ישנם רווחים שווים בין כל התמונות), הניתנת לשיטוט על ידי גרירה לכל כיוון (pan) ופעולות זום-אין וזום-אאוט. המפה היא רציפה ודינמית, וכל טלאי במפה נוצר רק ברגע שבו הוא נדרש.

תמונות וצורות תלת מימדיות הן אובייקטים ממימד גבוה מאוד, והדמיון ביניהן לא יכול להיות מיוצג כראוי במרחב דו מימדי רציף. לכן, כל מיפוי גלובלי של תמונות או צורות למפה דו מימדית חייב להכיל אזורים שאינם רציפים או חלקים. במפות דינמיות, לעומת זאת, מאחר וכל טלאי במפה מיוצר בפני עצמו ולא מבוסס על מפה גלובלית, ניתן לשמור על כך שכל טלאי יהיה רציף באופן מקומי ושהמפה תהיה עקבית. עבור המשתמש, שרואה בכל רגע נתון טלאי רציף שמהווה חלק קטן מן המפה, התחושה היא שקיימת מפה גלובלית שמתוכה נלקח הטלאי, וכך נוצרת חווית שיטוט רציפה וחלקה. יתרון נוסף של מפות דינמיות היא האפשרות להתאים את התמונות המוצגות לכיוון השיטוט במפה, כך שרצון המשתמש מיוצג טוב יותר, בניגוד למפות גלובליות שבהן היחסים בין האזורים השונים במפה מוכתבים מראש.

ניתן ליצור מפות דינמיות ביעילות רבה, ללא תלות במספר האובייקטים שבמאגר. למעשה, החלק המשמעותי ביותר בזמן יצירת המפה מוקדש לטעינת התמונות הנדרשות מהדיסק. מכאן, שמפות דינמיות יכולות לשמש לסקירת מאגרי תמונות עצומים המכילים מיליוני תמונות. כמו כן, אלגוריתם יצירת המפה מנותק לחלוטין מחישוב הדמיון בין האלמנטים

תקציר

1 הקדמה

מאגרים של אובייקטים תלת מימדיים שהיו קטנים ומועטים הפכו להיות גדולים ונפוצים יחסית. לעומתם, מאגרי תמונות עברו תהליך דומה אך בסדר גודל אחר; הם הפכו מגדולים ונפוצים לעצומים וזמינים בכל כיס. זמינות המאגרים מגדילה את הביקוש לדרכים יעילות, אמינות ואינטואיטיביות לארגון וסקירת מאגרים של תמונות וצורות תלת מימדיות.

עבודת זו מתרכזת במחקר של *דמיון סמנטי* בתוך אוספים גדולים של תמונות או צורות תלת מימדיות. אנו מעוניינים במדידתו של הדמיון סמנטי, כמו גם באפליקציות אפשריות של מידת דמיון מסוג זה. זיהוי תמונות דומות או צורות דומות מהווה בסיס למגוון רחב של שימושים, כגון מציאת תמונות וצורות (retrieval), סקירת מאגרי תמונות וצורות (exploration), וחלוקה שלהן לקטגוריות. לצורך סקירת מאגרי תמונות וצורות, אנו מציעים *מפות דינמיות (Dynamic Maps)*, שיטה שמתבססת על הדמיון בין תמונות\צורות ומאפשרת שיטוט אינטואיטיבי במאגרים גדולים ללא תלות במילות מפתח או סינון. עבור שיטה זו, כמו אפליקציות נוספות, חשוב שהדמיון בין התמונות\צורות ישקף מידע סמנטי ששייך לתמונה או לצורה, כגון דרך השימוש באובייקט, המקור שלו, מיקומו ועוד. עם זאת, שיטות עדכניות לחישוב דמיון מתבססות לרוב על מאפיינים מקומיים ולא משקפות מידע סמנטי כראוי.

מונעים מהצורך בשיטות למדידת דמיון שמשלבות מידע סמנטי, אנו מציגים שתי שיטות למדידת דמיון סמנטי. הראשונה מתבססת על מיקור המונים (crowdsourcing), ומציעה דרך תשאול שבה ניתן להפיק מידע רב מכל שאילתה, וכך להקטין את מספר השאילתות שנדרש לשלוח לקהל. השימוש במיקור המונים מאפשר לגלות מידע סמנטי אשר קיים מחוץ לתמונה או הצורה עצמה, למשל זיהוי של תמונות אשר צוירו בידי אותו אמן או חפצים ששייכים לסביבה דומה. השיטה השניה מודדת דמיון סמנטי בין צורות תלת מימדיות על ידי השוואת החלקים מהן מורכבות הצורות. על ידי שימוש בהתאמה בין החלקים השונים בכל צורה, ניתן להעריך אילו העתקות (transformations) נדרשות על מנת להפוך צורה אחת לשניה. אנו מגדירים את *מרחק העריכה של הצורות (Shape Edit Distance)* בסכום העלויות של ההעתקות הנ"ל. מרחק זה רגיש לדמיון בדרך השימוש של צורות (שכן לרוב בין צורות שנעשה בהן שימוש דומה יש דמיון מבני), ולדמיון בין צורות שהן תוצאות של עריכות שונות של אותה צורת מקור.

דרך החישוב של מרחק העריכה בין צורות מהווה דוגמה ליחס בין חלוקה של צורות למקטעים, התאמה בין צורות ודמיון בין צורות. שלושת הבעיות הללו הן נדבכי יסוד של תחום חקר הצורות (shape analysis), וכל אחת מהן זוכה למחקר רב ומעמיק בפני עצמה. עם זאת, קיים קשר חזק בין הבעיות, ופלט של אחת מהן יכול לשמש כקלט לאחרת. למשל, דמיון בין חלקי הצורה יכול לשמש לצורך חלוקה של הצורה למקטעים, החלוקה למקטעים יכולה לשמש לצורך חישוב התאמה בין הצורות, והתאמה בין צורות יכולה לשמש לחישוב דמיון סמנטי בין הצורות. אנו ממשיכים לחקור את הקשר בין התאמה ובין חלוקה של צורות, ומראים כיצד חלוקה למקטעים יכולה להיות לעזר בחישוב התאמה צפופה בין צורות (point-to-point correspondence, שבהן מסופקת התאמה מדויקת של כל

תמצית

בעבודה זאת, אנו חוקרים את המושג "דמיון סמנטי" בין צורות תלת מימדיות ובין תמונות. בפסקאות הבאות נתייחס לתמונות אך כל האמור תקף גם לגבי אובייקטים תלת מימדיים. מדידת הדמיון בין תמונות הוא אבן בניין באפליקציות רבות, כגון הדמיית היחסים בין התמונות, חלוקה שלהן לקטגוריות, מציאת תמונות מסוימות במאגר ושיטוט במאגר תמונות. כאפליקציה נוספת, אנו מציגים שיטה חדשנית לשיטוט במאגרי תמונות אשר מבוססת על הדמיון בין התמונות. אנו מניחים תמונות על גבי מפה חסרת קצוות, כך שתמונות קרובות במפה הינן דומות זו לזו. ניתן לשוטט במפה לכל כיוון כפי שמשוטטים על גבי מפה גיאוגרפית. כך נוצרת חווית שיטוט אינטואיטיבית וחלקה. השיטה היא יעילה ללא תלות בגודל המאגר, כך שהיא שימושית גם עבור מאגרים גדולים במיוחד של מיליוני תמונות.

עבור אפליקציה זו ועבור שימושים אחרים שצוינו לעיל, איכות הפתרון תלויה במידה רבה באיכות המדידה של הדמיון בין התמונות. על הדמיון המחושב בין התמונות לשקף את הדמיון הנתפס ביניהן באופן אינטואיטיבי על ידי בני אדם, כלומר דמיון סמנטי. על כן קיים צורך במחקר מעמיק על מנת לשפר שיטות קיימות לחישוב דמיון, שבמקרים רבים נעדרות הקשר סמנטי. לצורך כך, אנו מציגים כיצד ניתן לגלות דמיון סמנטי מורכב שלא ניתן לחישוב באופן אוטומטי על ידי שימוש בטכניקות מיקור המונים (crowdsourcing). שיטות מסוג זה יכולות לספק הקשר סמנטי חיצוני לתמונה ולשמש כהשלמה לשיטות חישוב דמיון אוטומטיות.

האמור לעיל תקף במידה שווה לגבי תמונות ולגבי צורות תלת מימדיות. עתה נעבור להתמקד בצורות תלת מימדיות בלבד. חלוקה של צורה למקטעים (segmentation), התאמה בין צורות, ודמיון בין צורות הינן בעיות מפתח בתחום, וכל אחת מהן זוכה למחקר מעמיק בפני עצמה. יחד עם זאת, קיים קשר חזק בין הבעיות הללו, ופלט של אחת מהן יכול לשמש כקלט לאחרת. אנו מציגים שיטה אשר מודדת דמיון בין צורות על ידי חלוקתן למקטעים ומציאת התאמה בין המקטעים. מידת דמיון זו משקפת יחסים סמנטיים בין הצורות כגון צורות השייכות לאותו הסגנון או צורות שדרך תפעולן דומה.

אנו ממשיכים לחקור את הקשר בין התאמה בין צורות ובין חלוקתן למקטעים, ומראים כיצד חלוקה למקטעים יכולה לשמש לשיפור התאמות צפופות בין צורות (שבהן מסופקת התאמה מדויקת של כל נקודה בצורה). כאשר מחשבים התאמה בין צורות סימטריות, ישנם כמה פתרונות סותרים לבעיית ההתאמה, אשר גורמים לחוסר יציבות של הפתרון הסופי. אנו מגדירים התאמה מסוג חדש אשר נקרא לה התאמה סימטרית (symmetry aware correspondence). בהתאמה זו כל מקטע יכול להתאים לקבוצת מקטעים סימטריים בצורה השניה. הגדרה זו מאפשרת לנו למצוא את התאמות בין צורות סימטריות ביעילות רבה. ההתאמה הסימטרית שאנו מוצאים הינה פחות פרטנית אך יותר מדויקת משיטות אשר מוצאות התאמה חד-חד ערכית. אנו מראים דרך לשימוש בהתאמה סימטרית בין מקטעים על מנת לשפר התאמות חד-חד ערכיות צפופות בין צורות, ללא צורך במציאת התאמה חד-חד ערכית בין המקטעים. כמו כן, חישוב התאמה חד-חד ערכית בין מקטעים על פי התאמה סימטרית יכול להתבצע כתהליך המשך במנותק ממציאת ההתאמה עצמה, ולכן צפוי שתהליך זה יהיה פשוט יותר.

TEL AVIV UNIVERSITY

אוניברסיטת תל-אביב

הפקולטה למדעים מדוייקים ע"ש ריימונד ובברלי סאקלר

ביה"ס למדעי המחשב ע"ש בלבטניק

דמיון והתאמה בין אובייקטים תלת מימדיים ובין תמונות

חיבור זה מוגש כחלק מהדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

יניר קלימן

עבודה זו בוצעה בביה"ס למדעי המחשב

תחת הנחייתו של פרופ' דניאל כהן-אור

הוגש לסנאט של אוניברסיטת תל-אביב

אוגוסט 2016

TEL AVIV UNIVERSITY

אוניברסיטת תל-אביב

הפקולטה למדעים מדוייקים ע"ש ריימונד ובברלי סאקלר

ביה"ס למדעי המחשב ע"ש בלבטניק

דמיון והתאמה בין אובייקטים תלת מימדיים ובין תמונות

חיבור זה מוגש כחלק מהדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

יניר קלימן

אוגוסט 2016