LOW AND MID-LEVEL SHAPE PRIORS FOR IMAGE SEGMENTATION

by

Alex Levinshtein

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Low and Mid-Level Shape Priors For Image Segmentation

Alex Levinshtein

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2010

Perceptual grouping is essential to manage the complexity of real world scenes. We explore bottom-up grouping at three different levels. Starting from low-level grouping, we propose a novel method for oversegmenting an image into compact superpixels, reducing the complexity of many high-level tasks. Unlike most low-level segmentation techniques, our geometric flow formulation enables us to impose additional compactness constraints, resulting in a fast method with minimal undersegmentation. Our subsequent work utilizes compact superpixels to detect two important mid-level shape regularities, closure and symmetry. Unlike the majority of closure detection approaches, we transform the closure detection problem into one of finding a subset of superpixels whose collective boundary has strong edge support in the image. Building on superpixels, we define a closure cost which is a ratio of a novel learned boundary gap measure to area, and show how it can be globally minimized to recover a small set of promising shape hypotheses. In our final contribution, motivated by the success of shape skeletons, we recover and group symmetric parts without assuming prior figure-ground segmentation. Further exploiting superpixel compactness, superpixels are this time used as an approximation to deformable maximal discs that comprise a medial axis. A learned measure of affinity between neighboring superpixels and between symmetric parts enables the purely bottom-up recovery of a skeleton-like structure, facilitating indexing and generic object recognition in complex real images.

Acknowledgements

First and foremost I would like to express my gratitude to my two supervisors, Sven Dickinson and Cristian Sminchisescu. They are not only great supervisors, but good people. Their combined knowledge and experience served to help me keep my mind on the big picture while not neglecting important practical details. Having worked with Cristian since my Masters and even longer with Sven, our interactions and brainstorming have become an important part of my academic career and my life. Despite having finished my PhD, I hope that we will continue to have similar interesting discussions in the future.

During my 4 years of PhD I have collaborated with many interesting people. I would like to thank Kaleem Siddiqi, who visited UofT in 2007, for a very thorough introduction to geometric flows and level set methods, resulting in our joint Turbopixels work. I also thank Kyros Kutulakos, David Fleet and Allan Jepson. Kyros and Allan have been an important presence throughout my studies, starting from introductory computer vision courses that I took way back during my Bachelors. David has not only given me a lot of great advice about my work throughout the years, but is also the funniest professor I know. I also thank James Elder, my external examiner. He was very knowledgeable and provided many useful comments for my thesis.

I extend my gratitude to Suzanne Stevenson, for piquing my interest in research and introducing me to Sven. I also thank the faculty and students with whom I have interacted during my academic career: Rich Zemel, Sam Roweis, Ilya Sutskever, Pablo Sala, Diego Macrini, Francisco Estrada, Anatoliy Kats, Marcus Brubaker, Sam Hasinoff, and many others. I also acknowledge NSERC for financial support.

Finally I thank my family and those close to me: my parents, my sister Maya, Natasha and my dear Yulia. There have been some hard times along the way and I could not have done it without you.

Contents

Chapter 1

Introduction

1.1 The Need For Perceptual Grouping

From our very birth we are able to parse visual scenes into meaningful components. Even before we are taught what are the particular objects in our world, we already have an inherent concept of what is an “object”. Take, for example, the image in Figure 1.1a. Arguably, without ever seeing zebras before humans have little trouble in recognizing them as coherent objects in this image. This uncanny human ability to organize image components into coherent collections without relying on object-specific knowledge is referred to as perceptual grouping [105, 74].

Perceptual grouping is an essential human ability that facilitates fast scene analysis and object recognition. Our world contains a vast collection of objects of varying appearance and shape. It is made up of countless scenes, each composed of different components. Perceptual grouping is the compass that allows us to navigate in this sea of potential scenes, quickly parsing it into its constituents which we can then recognize. In computer vision, perceptual grouping is also of the utmost importance. We deal with images that are the result of taking photographs of the real world with a camera. The contents of the images are a product of the imaging conditions and the scene composition. Some types of images can be parsed more

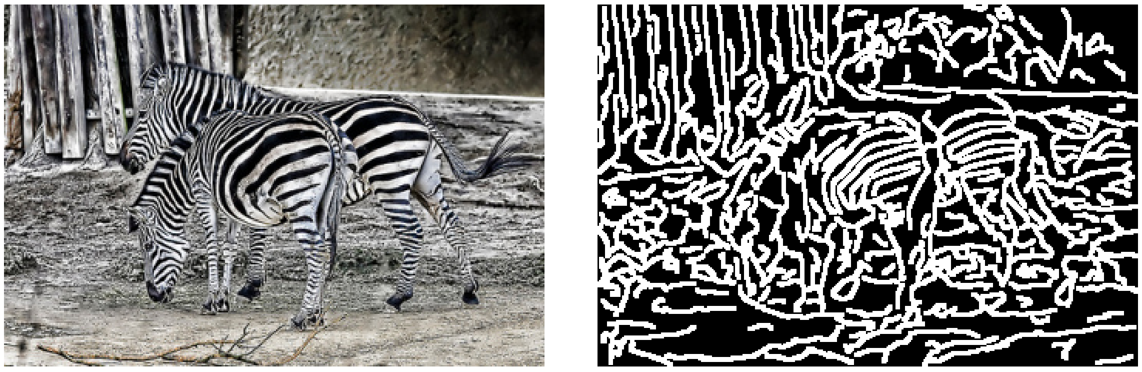


Figure 1.1: Perceptual grouping in action. Even without employing object level knowledge, perceptual grouping is used to group scene features into coherent components. While the original image (a) contains color, it often serves as a distractor due to heterogeneous internal appearance; image contours alone (b) may be sufficient for grouping.

easily than others due to restricted imaging conditions and scenes. For example, imaging mechanical parts on a conveyor belt in a factory is a constrained task. In this case, we know a priori that the scene is composed of a small number of very distinctive objects on a uniform, known background. However, in general, there is a prohibitive number of images that can be created when the imaging conditions and the scenes are unconstrained. Testing for the presence of all possible objects under all possible imaging conditions is intractable. Perceptual grouping is needed to deal with this complexity.

Having established the importance of perceptual grouping, let us now analyze this process applied to our example scene. First, note that while a real scene contains color (Figure 1.1a), shape alone (Figure 1.1b) is often sufficient and preferable for scene analysis. In fact, object appearance may be a distractor in cases of heterogeneous appearance, such as the zebra. Moreover, while appearance-based grouping is undoubtedly important, the appearance of objects within a given category may vary much more than their shape, making shape-based regularities more generic.

Let us examine the scene more closely. Zooming in on the zebra, we see that it is bounded

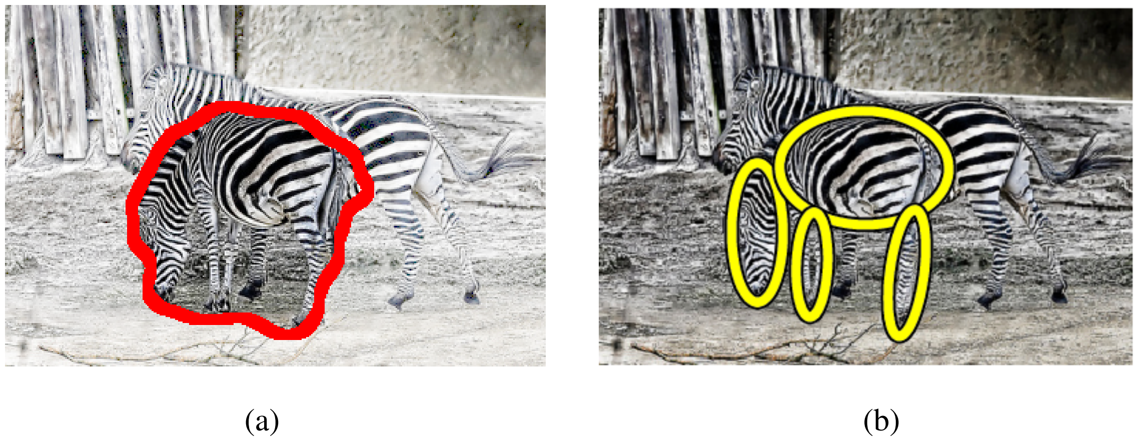


Figure 1.2: Using Closure and Symmetry for bottom-up grouping.

by strong continuous image edges. The whole object, with the exception of internal features and markings, can be represented by a single closed contour that is well supported by image edges (Figure 1.2a). This perceptual cue of closure helps us group image contours into larger, more meaningful components. Notice that the zebra is not just a single unordered blob, but is composed of parts. The head, the body, its legs and arms - all are coherent parts (Figure 1.2b). Besides being attached to the other parts at sharp contour concavities, the parts form elongated components that exhibit strong bilateral symmetry. Symmetry and parallelism, although not always present at the scale of the whole object, are often present at the scale of object parts [6, 77, 4]. Together with closure, these cues also help us parse the scene.

While humans successfully use closure and symmetry for grouping, as demonstrated by Gestalt phychologists, our goal is to employ these rules in computer vision. But how can we apply these large-scope grouping cues to find order in the sea of image contours in Figure 1.1b? To that end, we zoom in on the zebra and examine more local cues. First, while the appearance of a zebra, which is comprised of black and white stripes, is heterogeneous at an object scale, it is locally homogeneous within each stripe. Moreover, with the exception of a few concavities mainly between zebra parts, the bounding contour is mostly locally convex within a part (see zebra's back, belly, and neck, for example). Similar to the human visual system, that undoubtedly detects these regularities (local homogeneity and convexity) for grouping, we

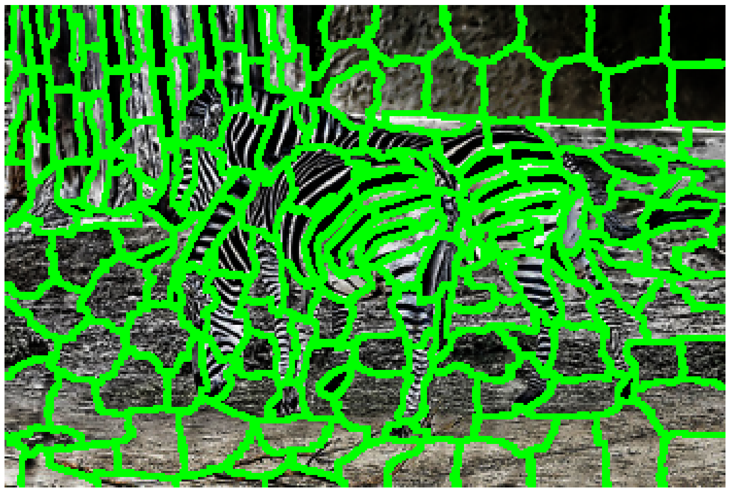


Figure 1.3: Superpixels as a basis for grouping.

can detect them using superpixels (Figure 1.3), with each superpixel capturing a local, homogeneous, compact patch. Being the product of local perceptual grouping, superpixels provide an ideal local scope for appearance and shape analysis. Using superpixels allows us to build algorithms that can efficiently recover closure and find symmetric parts in difficult scenes that are a challenge for even the human visual system (such as Figure 1.1a). Note that we do not claim that superpixels are a natural image representation, such as image contours. Instead, Chapters 4 and 5 show that choosing to use superpixels over image contours can aid many perceptual grouping tasks. In fact, most of the symmetric parts in Figure 1.2b can be automatically recovered by the system presented in Chapter 5, while Figure 1.2a is an actual output from the system in Chapter 4. The next section provides an overview of the thesis, describing a method for recovering compact superpixels and their use for closure and symmetry detection.

1.2 Thesis Overview

This thesis explores perceptual grouping at different scales. Chapter 2 starts with a review of perceptual grouping literature in computer vision. In the main body of the thesis, we start with local grouping and show how low-level cues can be used for important tasks such as superpixel segmentation (Chapter 3). Building on superpixels, we perform higher-level grouping tasks using cues such as closure and symmetry (Chapters 4 and 5, respectively), in an attempt to extract coherent objects in a bottom-up fashion. While the methods in Chapters 3–5 differ in the level of the perceptual grouping rules that they use, they are unified in their global goal of grouping with no prior scene or object knowledge. The overall work pushes the state-of-the-art in perceptual grouping and provides some insight into the mechanisms behind it. The thesis concludes with some discussion and ideas for extending the work (Chapter 6). Next we overview the work presented in this thesis and provide a summary of its contributions.

1.2.1 Computing compact superpixels

Similarity of appearance and proximity are perhaps two of the lowest-level perceptual grouping rules. Throughout the years, they have been used to design generic segmentation algorithms that can segment an image into disjoint regions based on appearance cues. Recently, some standard segmentation algorithms have been used to generate an oversegmentation of the image into superpixels, helping to alleviate the complexity of many high-level vision tasks. This relief in complexity comes at a price, since using superpixels instead of pixels is counterproductive if undersegmentation errors result. Most standard segmentation algorithms, while being very fast, do not constrain the superpixels to be compact or regular in size. Compactness and regularity impose a bound on superpixel size. In the absence of such constraints superpixels can grow to become quite large and can therefore result in a significant amount of undersegmentation (Figure 1.4c-e). One segmentation algorithm constrains the superpixels to be compact, but is very memory and time intensive (Figure 1.4b). The first contribution of this thesis (Chapter 3) is a new superpixel segmentation algorithm that reduces undersegmentation while being both fast and memory efficient (Figure 1.4a).

Building on the principles of curve evolution, multiple superpixel seeds are placed in a regular grid and are evolved in parallel until their boundaries collide. The speed of the evolution is partially governed by underlying image edges, producing superpixels that are homogeneous

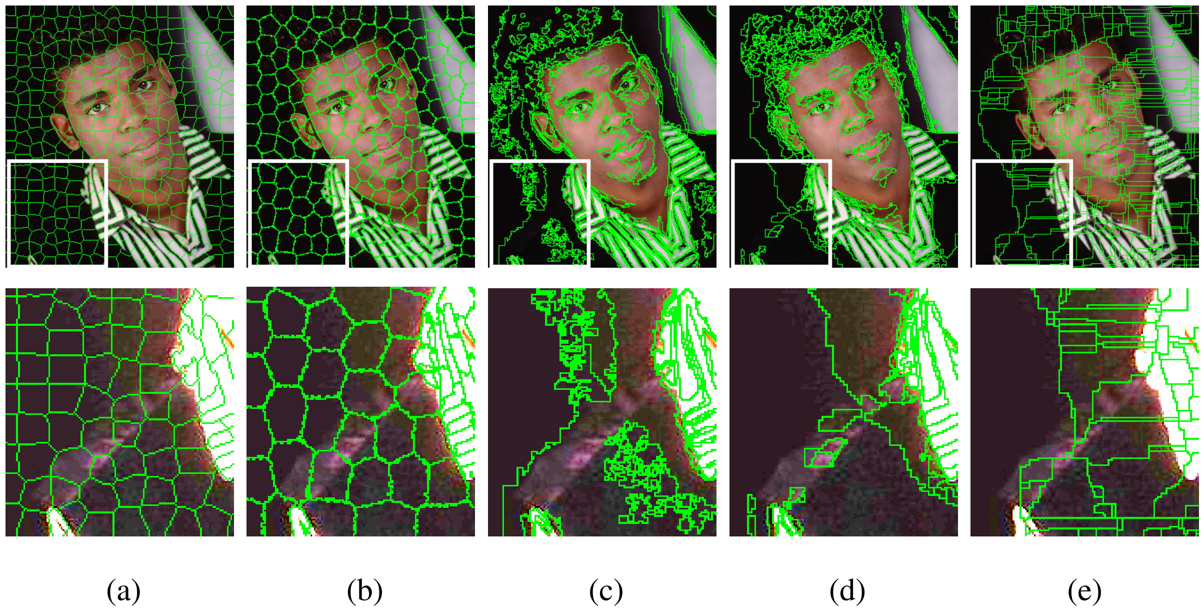


Figure 1.4: Over-segmentations obtained with five algorithms: (a) TurboPixels (b) N-Cuts[111] (c) Local variation [31] (d) Mean-shift [19] (e) Watershed [102]. Each segmentation has (approximately) the same number of segments. The second row zooms in on the regions of interest defined by the white boxes.

in appearance. Our initial grid-like seed placement implicitly ensures that superpixels are compact and regular in size. However, to further minimize undersegmentation, the speed of seed evolution explicitly pushes the superpixels to be compact, thereby making use of another low-level perceptual grouping prior. We show that the resulting algorithm compares favorably to the competition in terms of speed and accuracy. The next two chapters of the thesis build on superpixels, but instead of object-specific segmentation tasks, superpixels are used to build generic perceptual grouping algorithms. To the best of our knowledge, using superpixels in this manner is novel.

1.2.2 Finding closure

Moving further up the scale of prior information, we make use of closure for grouping (Chapter 4). Finding closure in images in a bottom-up fashion is one of the key challenges in percep-

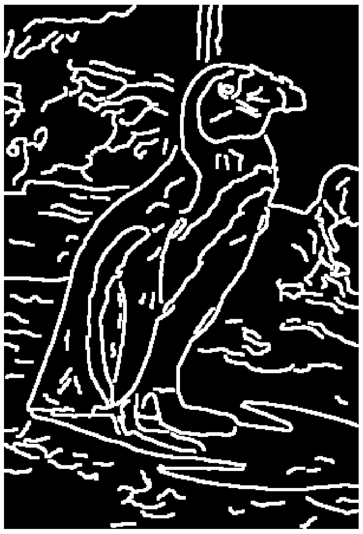


Figure 1.5: Finding contour cycles to recover closure. Recovering closure is commonly posed as a problem of finding cycles of contours that have good continuity and little overall gap between the contour fragments. In a real world image, the number of possible contour cycles is prohibitive, making a brute force approach to the problem intractable.

tual grouping. A successful solution to this important problem would significantly simplify the complexity of some high-level vision tasks. For instance, standard object recognition methods use a sliding window approach, where an object detector is tested on many image windows at different positions and scales. Instead, if a small number of image regions that are likely to correspond to objects could be obtained bottom-up using a closure finding method, such regions could be used as hypotheses for object recognition.

The problem of recovering closure is commonly posed as linking together a set of fragmented contours into a cycle that separates an object from its background. What makes the problem particularly hard is the intractable number of cycles that may exist in the contours extracted from an image of a real scene (Figure 1.5). Testing all possible cycles of contours is intractable without somehow reducing the complexity of the problem. Closure finding techniques commonly solve this complexity issue by either employing heuristics to prune the search space [44, 87, 29, 30] or finding a globally optimal solution using a restricted model of closure

[103, 99]. Instead, we try to avoid using heuristics to detect closures, while minimizing the amount of restrictions on our closure model. Thus, the second contribution of the thesis is a globally optimal method for detecting closure in images.

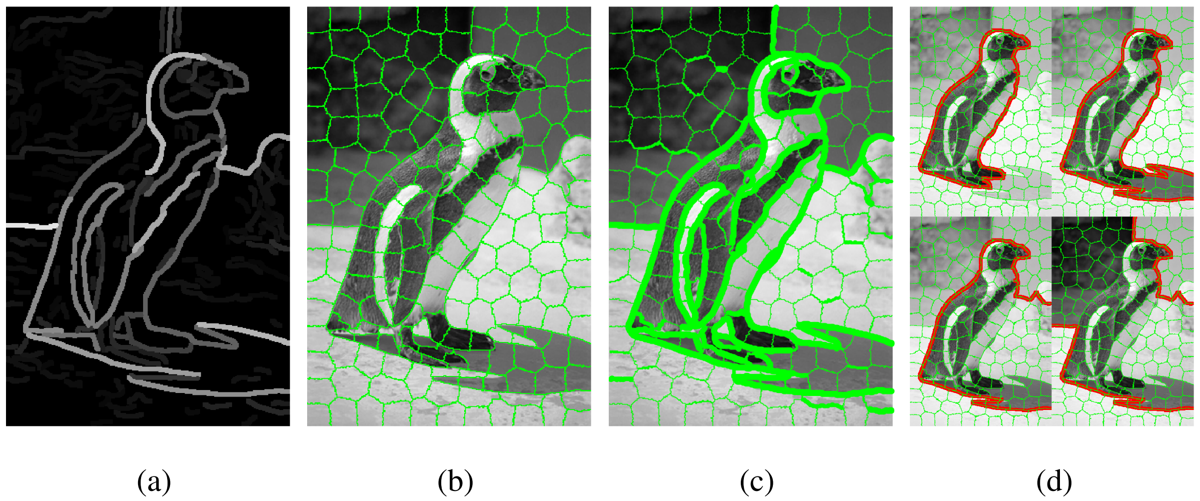


Figure 1.6: Overview of our closure detection approach: (a) contour image – while we take as input only this contour image, we will overlay the original image in the subsequent figures to ease visualization; (b) superpixel segmentation of contour image, in which superpixel resolution is chosen to ensure that target boundaries are reasonably well approximated by superpixel boundaries; (c) a novel, learned measure of gap reflects the extent to which the superpixel boundary is supported by evidence of a real image contour (line thickness corresponds to the amount of agreement between superpixel edges and image contours); (d) our cost function can be globally optimized to yield the largest set of superpixels bounded by contours that have the least gaps. In this case the solutions, in increasing cost (decreasing quality), are organized left to right, top to bottom.

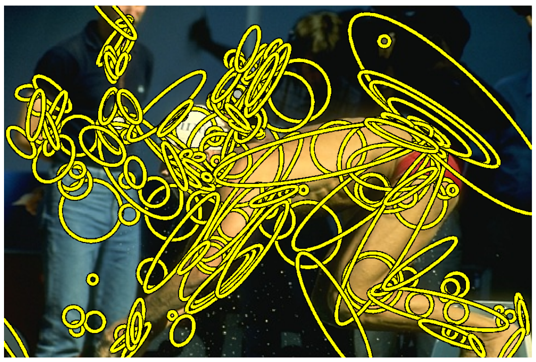
Given an image of extracted contours, we begin by restricting contour closures to pass along boundaries of superpixels computed over the contour image, thereby reformulating the problem of searching for cycles of contours as the problem of searching for subsets of superpixels whose collective boundary defines an appropriate cycle of contours (Figure 1.6a-b). We build a cost function over subsets of superpixels that measures the gap (a measure of missing image edges) of the region's boundary relative to its area. We learn the gap measure (Figure 1.6c) and finally show how to optimize the proposed closure cost in a globally optimal manner, returning a small number of solutions representing subsets of superpixels that have good closure (Figure 1.6d). In an extension to this work, we show how the same technique can be used to detect closure in spatiotemporal domains. We show that our closure detection method achieves state-of-the-art results.

1.2.3 Recovering and grouping symmetric parts

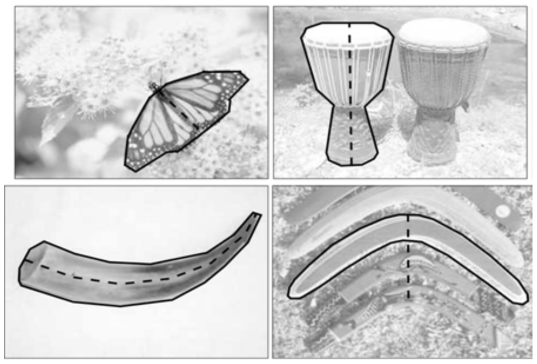
Recall that the underlying goal of this thesis is to perceptually group image features to ease the complexity of scene analysis. The more grouping we can do bottom-up, the less we rely on strong prior scene knowledge, which may be unavailable. However, bottom-up grouping that is performed with no prior scene knowledge can only take us so far. In the case of superpixels, the complexity of subsequent grouping or search may be diminished, but there are still a great number of superpixel groups that need to be considered. In the case of closure, we extend the grouping to recover entire objects. However, without any additional constraints, there is still a significant number of accidental groups that have good closure. Moreover, objects that have weak bounding contours are hard to detect using only the closure cue. The question is whether we can employ a stronger bottom-up prior for grouping, without using any prior object or scene information?

Stopping short of using object priors, we can use prior information at the part level. The underlying premise is that an object can be described using a small vocabulary of parts. Early computer vision researchers explored this idea [5] and focused particularly on parts that are based on symmetry. While symmetric parts cannot be used to describe all objects, a large subset of objects can be described in terms of concatenations of symmetric parts. But what is perhaps most significant is that exploiting local symmetry facilitates perceptual grouping under some very challenging conditions.

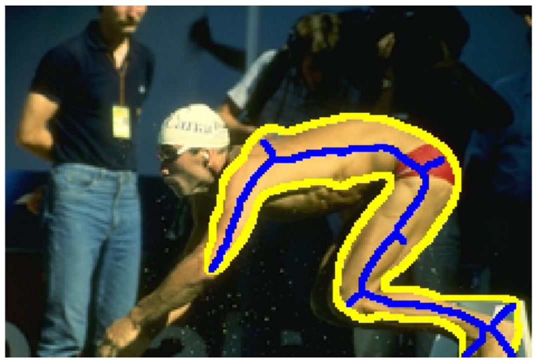
Existing approaches that attempt to extract and group symmetric parts can be divided into



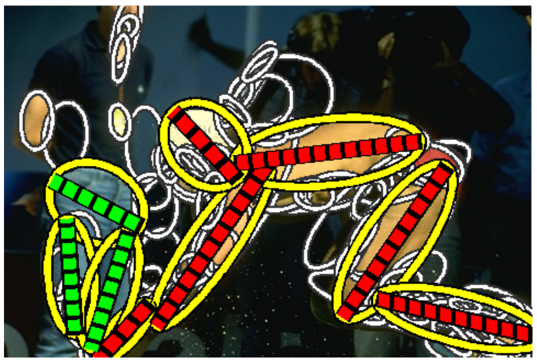
(a)



(b)



(c)



(d)

Figure 1.7: Approaches for symmetry detection. (a) Filter-based approaches analyze multiscale filter responses, (b) Contour-based approaches group contour fragments, (c) Skeleton-based methods analyze shape symmetry by using the figure-ground segmentation, (d) Our approach efficiently detects and groups symmetric parts bottom-up.

filter-based, contour-based, and skeleton-based. Filter-based approaches (Figure 1.7a) detect parts by analyzing different low-level filter responses. However, simply detecting parts as local maxima in a set of multiscale filter responses leads to many false positives and false negatives, suggesting that successful part extraction requires paying closer attention to image contours. Contour-based methods (Figure 1.7b) do just that, but are faced with the prohibitive complexity of selecting subsets of contours that exhibit symmetry. Skeleton-based approaches (Figure 1.7c) assume that the figure-ground segmentation task is solved, and analyze the symmetry of the figure, resulting in very powerful symmetric description. In our work, we try to improve on the above approaches. We use superpixels to detect parts that closely follow image evidence without the overwhelming complexity of grouping contours. As a result, we extract a skeleton-like part structure, but do so bottom-up, without requiring figure-ground segmentation (Figure 1.7d).

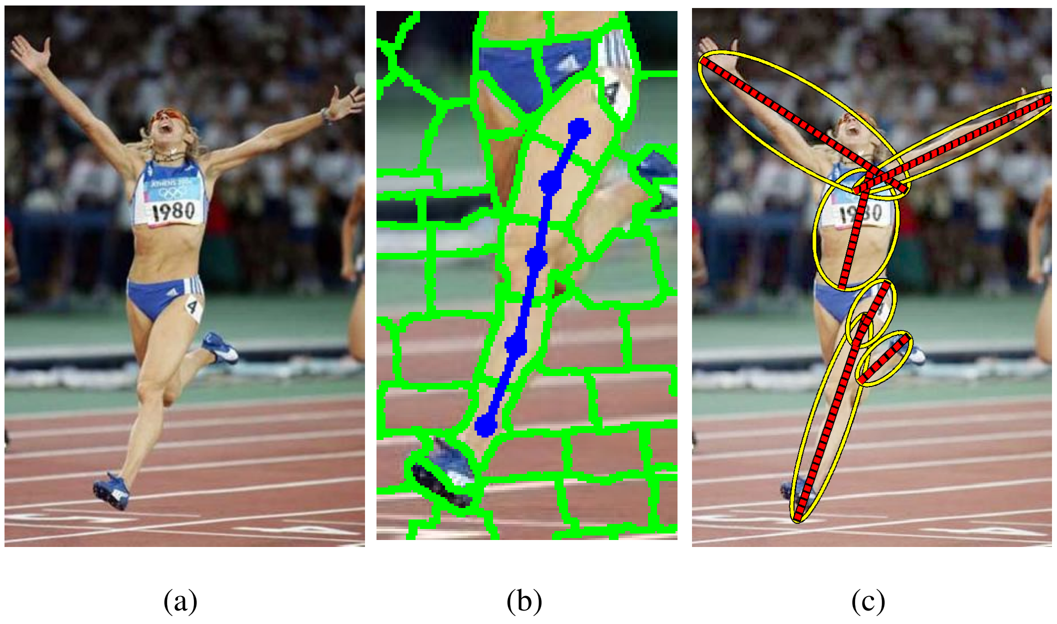


Figure 1.8: Our symmetric part detection and grouping. (a) An input image; (b) Once superpixels are extracted, we try to form chains of superpixels to detect parts; (c) The result is a skeleton-like representation, where the parts were detected and grouped bottom-up.

Drawing on the principle that a skeleton is defined as the locus of medial points, i.e., centers of maximally inscribed disks, our key idea is to hypothesize a sparse set of medial points at multiple scales by segmenting the image into compact superpixels at different superpixel resolutions, making object parts naturally correspond to chains of superpixels (Figure 1.8b). Superpixels are adequate for this task, balancing a data-driven component that is attracted to shape boundaries while maintaining a high degree of compactness. The superpixels (medial point hypotheses) at each scale are linked into a graph, with edges adjoining adjacent superpixels. Each edge is assigned an affinity that reflects the probability of two adjacent superpixels belonging to the same symmetric part (medial branch). A standard graph-based segmentation algorithm applied to each scale yields a set of superpixel clusters which, in turn, yield a set of regularized symmetric parts. In the second phase of our approach, we address the problem of perceptually grouping symmetric parts arising in the first phase. Like any grouping problem, our goal is to identify sets of parts that are causally related, i.e., unlikely to co-occur by accident. Again, we adopt a graph-based approach in which the symmetric parts across all scales are connected in a graph, with edges adjoining parts in close spatial proximity. Similar to the first phase, each edge is assigned an affinity, this time reflecting the degree to which two nearby parts are believed to be physically attached, and the same graph-based segmentation algorithm is applied to yield part clusters, each representing a set of regularized symmetric elements and their hypothesized attachments. The end result is a method that, starting from an image (Figure 1.8a), is able to detect and group symmetric parts in a bottom up fashion (Figure 1.8c).

1.2.4 Summary of Contributions

This thesis explores perceptual grouping using shape priors of different degrees, thereby making the following contributions:

- Chapter 3 introduces a new superpixel extraction algorithm.
  - Unlike most previous approaches, our method is geared towards extracting compact superpixels, thereby minimizing bleeding and undersegmentation.
  - Compared to an approach that does enforce superpixel compactness (Ncuts [93]), our algorithm is based on local cues and is therefore orders of magnitude faster achieving comparable performance for dense superpixels segmentations.
  - Our curve evolution framework for superpixel extraction is flexible enough to be extended to include additional shape priors besides compactness in the future.
- Chapter 4 introduces a new closure detection algorithm.

  - Instead of finding cycles of contours, we formulate the problem as finding subsets of superpixels, thereby significantly reducing the complexity of finding closures.
  - Unlike previous methods, the new closure detection approach minimizes a standard closure cost in a globally optimal fashion without resorting to the use of heuristics or simplification of the closure model.
  - Compared to competing approaches, which focus on returning a single best solution, we return a small set of promising shape hypotheses and outperform the competition, better serving high-level vision tasks.

- Chapter 5 introduces a new method for bottom-up recovery and grouping of symmetric parts.
  - Our main contribution is the new idea of approximating the maximal discs of a medial axis using compact superpixels.
  - A learned measure of superpixel and part affinity, trained on a set of noisy images, together with greedy grouping heuristics, enables us to overcome the prohibitive complexity of detecting and grouping symmetric parts in real images.
  - In contrast to competing approaches, our method does not require prior figure-ground segmentation and recovers precise symmetric parts, closely supported by underlying image contours.
  - Our resulting approach is able to efficiently recover a full skeleton-like structure in a purely bottom-up fashion, facilitating indexing and generic object recognition.

Chapter 2

Related Work

2.1 Introduction

Over the past 40 years, object recognition in computer vision progressed from restricted scenes imaged under restricted conditions to complex scenes captured under realistic conditions. One reason for this progression is the advancement in modelling techniques that now mostly rely on new machine learning techniques, which make models more robust and suitable to real world conditions. Another reason lies in the development of new highly distinctive local features that robustly encode the appearance of image patches, such as SIFT (Lowe [59]), giving rise to patch-based object recognition methods. As opposed to earlier recognition techniques, the dimensionality and robustness of these features allows modern recognition techniques to use small collections of such features for object recognition in real world scenarios. The features are particularly suitable for recognizing specific exemplars of objects, as they capture distinctive local appearance. In such exemplar recognition scenarios, it is even possible to cope with large databases of exemplars, since the presence of even a single feature is sometimes sufficient to recognize an exemplar object. However, while the distinctiveness of local appearance features enables the robust recognition of particular exemplars, the same distinctiveness makes it hard to create models for object categories, thereby hindering more generic recognition. As a result, until recently, most patch-based techniques focused on recognizing restricted categories where members of a category share a common set of distinctive local features; examples include faces, motorcycles, or the backs or sides of cars.

Recently, the community recognized the need for more generic object recognition and attempted to push these appearance-based methods to datasets with a significant number of objects which undergo significant viewpoint and appearance changes[1]. While the current state-of-the-art is continually advancing, performance still greatly lags human performance. An important question to ask is whether such models of local distinctive features can be extended to be used for more complex recognition tasks, and whether they have the potential to achieve human performance.

In [5], Biederman mentions that a typical adult human being is able to recognize roughly $30,000$ objects. In addition to the large number of possible object categories, there are additional challenges. First, one has to deal with the potential transformations of an object: location, orientation, size, and possibly non-linear deformations such as articulation or bending if the object is not rigid. Secondly, even if the exact location of the object is known, the object's appearance can vary due to illumination and occlusion effects. Lastly, even more significant appearance and shape variability can be present within a given category of objects. The above transformations pose great challenges to finding objects in images. Attempting to recognize objects under these conditions using highly distinctive local features will require creating many separate models per object category that depend on the selected exemplar, the view, and the other potential transformations mentioned above, creating a prohibitive number of models. It would be infeasible to directly match every one of these models to an image. Instead, one needs to first narrow down the number of candidate objects to a reasonable number and only then test those candidates.

Going back to the early 1980s, the computer vision community has already recognized the

[1]For datasets and related publications visit the Pascal Visual Object Classes homepage at `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`.

need for perceptual grouping to handle the overwhelming complexity of object recognition. Lowe [58] introduced various view-point invariant 2D relations over line segments, such as curve co-termination, allowing to make a partial inference about their 3D relations. As it is very unlikely that two arbitrary image curves would co-terminate, co-termination of curves in 2D suggests that they are likely to be related in 3D. Brooks [11] used similar relations over ribbons to provide constraints for object recognition. It was precisely such perceptual grouping that allowed these authors to sufficiently lower the complexity of object recognition by reducing the number of candidate groups that are matched with object models.

Narrowing down the number of potential candidate objects requires the ability to form large abstract groups of low-level image primitives in a bottom-up fashion. Bottom-up grouping can range from the formation of small local components to the extraction of whole objects. While the latter completely eliminates the need to account for external object transformations, such as an object's location and orientation, even local grouping eases the complexity of accounting for all possible transformations. In turn, the abstraction of the recovered groups introduces robustness to changes in imaging conditions and within-class object variability, further easing the complexity of generic object recognition. Obviously, further bottom-up processing will result in fewer object candidates, and decreases the computational burden on object recognition. Therefore, with the hope of extracting whole objects bottom-up, one may start by applying simple grouping cues such as local appearance similarity [102, 19, 31, 93, 104]. However, with the exception of a small number of objects that have a purely homogeneous appearance, most real world objects are unlikely to be extracted through the application of purely local low-level grouping cues. In order to extract large meaningful groups of image primitives, one therefore needs to impose stronger grouping constraints.

Stopping short of applying object-level models, as this would deflate the role of perceptual grouping, perhaps the strongest of grouping constraints amounts to abstract part models and their attachments. Using a small set of abstract parts facilitates concise modeling of a wide variety of objects under different transformations. In the early days of computer vision, several researchers proposed various vocabularies of such parts, like geons [4], superquadrics [77] and generalized cylinders [6]. Indeed, the use of such parts not only enables one to model a lot of object categories but also allows one to model significant within-class variability. Such concise representations are not only biologically motivated [5], but their simplicity (low numbers of parts and relations) enables us to cope with the complexity of having a large database of objects. However, recovering and grouping abstract parts from an image in a bottom-up fashion is a difficult problem and it suffers from some of the same complexity issues that plague object recognition. Moreover, while constraining objects to be part-based helps us to advance bottom-up grouping, overly restricting the set of parts limits the number of objects representable by such parts and hinders generic recognition. Backing away from abstract part-based object representations, but without going as far as using only local grouping rules, one can find middle ground by representing objects using smooth, closed curves [103, 27, 44, 47, 99, 29, 30, 107, 46]. Such a representation is more flexible than representing objects using abstract parts. However, being the less constrained model, it is more likely to fail in extracting whole objects due to a multitude of accidental groups of image contours well-captured by smooth closed curves.

In summary, perceptual grouping approaches in computer vision differ in the scope and the complexity of grouping rules that they employ. There is a clear tradeoff between the strength of the grouping rules vs. the flexibility in scene description and the complexity of applying such rules. The next section introduces the origins of perceptual grouping and reviews it at its different levels. Starting from the grouping of low-level primitives, we review methods that group image pixels into larger regions relying on appearance similarity. Proceeding to higher-level grouping, we show how primitives such as contour fragments can be extracted and grouped using principles of continuity and proximity. While the application of such low-level grouping significantly reduces the complexity of high-level tasks, it usually proves insufficient for generic object recognition scenarios. Thus, our review concludes with more complex mid-level perceptual grouping, such as parametric curve extraction, closure detection, and recovering symmetry, facilitating complex object categorization.

2.2 Perceptual Grouping

Perceptual grouping refers to the human ability to detect statistically significant relations between image primitives and group low-level image features into higher level structures without the (or with limited) a priori knowledge of the scene. In Section 2.1, we motivated the need for perceptual grouping, answering the question of **why** it is important to be able to group low-level image features bottom-up. This section will focus on a second question of **how** such grouping can be done without (or with limited) scene knowledge.

The research in this area began in the 1920's by the Gestalt psychologists [105]. The Gestaltists believed that object perception results from the overall structure of the visual information. Max Wertheimer, one of the founding fathers of perceptual organization, categorized the perceptual rules [105] based on a series of psychological experiments. Wertheimer used images of simple geometric primitives (dots and curves) to demonstrate his laws. Here are some of the commonly accepted perceptual grouping laws (see Fig. 2.1 for examples) :

- proximity - elements that are close together tend to be grouped
- similarity - elements that are similar in some properties (such as color, orientation or size) tend to be grouped
- common fate - all else being equal, items moving in the same direction are grouped

There were also additional rules that apply to line-like elements:

- continuity - edges that form smooth continuations of one another are grouped
- symmetry - symmetric edges are grouped
- parallelism - parallel edges are grouped
- closure - elements forming a closed figure are grouped

Some additional rules were proposed more recently by Palmer et al. (see [74]):

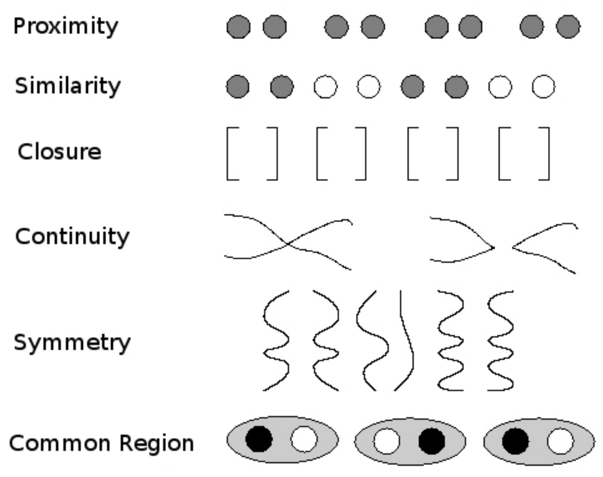


Figure 2.1: Examples of perceptual grouping rules

- synchrony - synchronously moving elements are grouped
- common region - elements in a common region of space are grouped
- element connectedness - elements that are connected by other elements tend to be grouped

The above laws are coordinated by the law of Pragnanz: “of several geometrically possible organizations that one will occur which possesses the best, simplest and most stable shape”. While this law is intuitively true, the Gestaltists provide no mathematical theory for the application of this law. What is a good, simple and stable shape? In computer vision, this question is sometimes answered through the application of heuristics. There are, however, two perceptual organization principles with a precise computational framework that are used in the literature [113]. One is the likelihood principle, first introduced by Helmholtz [38] and known in computer vision as the non-accidentalness principle [108, 57]. This principle assigns a high probability of grouping elements whose arrangement was unlikely to have occured accidentally. The second principle is the minimum description length (MDL) principle, giving a higher significance to simpler groupings of elements [83].

The principle of non-accidentalness was initially introduced into computer vision by Lowe [57], as well as Witkin and Tenenbaum [108]. Its goal is to find groups of features that are unlikely to occur by accident. This principle is the outcome of a simple application of Bayes' rule. Let *Causality* denote the event that a set of features are part of (or caused by) the same object and let *Organization* denote the event that the features have some organization among themselves. Then we are interested in the probability that the features come from the same underlying cause, given that we have found some organization among them. Using Bayes' rule, we obtain:

$$P(Causality|Organization) = \frac{P(Organization|Causality)P(Causality)}{P(Organization)} \tag{2.1}$$

Thus a particular grouping is more likely to be non-accidental if:

- It is not likely to occur arbitrarily ($P(Organization)$ is low)
- It is likely to occur if it is a result of a common cause - the features belong to the same object ($P(Organization|Causality)$ is high)
- Scenes have order (contain objects) - there is a significant number of feature groupings having a common cause ($P(Causality)$ is high)

The principle of finding non-accidental perceptually salient groups was heavily applied in the 80's and 90's. Lowe [57] applies the above principle for finding non-accidental line groups, facilitating the task of 3D object recognition. Mohan and Nevatia [67] use this principle on a larger scale and apply it to extracting and grouping different image primitives such as curves, symmetric curve pairs and closed ribbons. Many other approaches also incorporate this principle, but do so in a less explicit way, without explicitly modeling the above probabilities but simply providing a measure of goodness for a group. Jacobs [44], for example, uses the principle for grouping lines into convex groups, and while he does not measure the above probabilities explicitly, he proves that low values of his grouping cost function correspond to highly non-accidental convex line arrangements.

The second principle is the minimum description length (MDL) principle, giving higher significance to simpler groupings of elements [83]. This principle can be thought of as a scientific application of the Occam's Razor principle - the simplest explanation is the best. The goal in MDL is to select the best model of the data in terms of the length of its encoding. More concretely, we are first given a set of discretely defined models $M = \{M_1, M_2, \ldots, M_n\}$ or a family of models M defined by some continuous parameter vector θ, as well as our data X. Dealing with the continuous case, for example, the goal would then be to find the best parameters $\hat{\theta}$ that minimize the description length of the data under the chosen model $M(\hat{\theta})$, as well as the description length of M itself (i.e., the description length of $\hat{\theta}$). Under a Bayesian interpretation of the MDL principle (see [35]), this reduces to the Maximum A Posteriori (MAP) estimate to the likelihood. Specifically, we aim to find $\hat{\theta} = argmax_{\theta} P(\theta|X)$, which is equivalent to $\hat{\theta} = argmax_{\theta} P(X|\theta)P(\theta)$.

In perceptual grouping, and in computer vision in general, the MAP principle is used in many approaches that involve some statistical modeling of the problem. It is worth noting that there are other interpretations to MDL that yield better bounds for the description length, such as the Normalized Maximum Likelihood (NML) interpretation [83, 35]. While MDL can be thought of as an application of the MAP model selection principle, to the best of our knowledge its more rigorous formulation was first used in computer vision by Leclerc [51], who applied it to segmenting the image into piecewise smooth intensity regions. A more high-level task for which this principle was used was in the work of Pentland [78], who used MDL to select the simplest set of projections of volumetric parts. Nowadays, many approaches that include statistical modeling use the MAP principle, thereby implicitly employing MDL.

Comparing the MAP formulation of the MDL principle to that of the non-accidentalness principle illuminates some apparent similarities. Both techniques try to select the best set of elements by using Bayes' law. We argue that the two principles are in fact two approaches to the same idea. Take, for example, the problem of finding lines in an image by grouping image edgels. $P(Causality)$ would then measure the a priori probability of lines (perhaps vertical or horizontal lines are more frequent than others), which is exactly the same term as the model probability $P(\theta)$ in MDL. Given a particular line model, $P(Organization|Causality)$ as well as $P(X|\theta)$ measure the probability of the observed image edgels (how well the edgels fit a given line), while $P(Organization)$ and $P(X)$ in MDL measure the a priori probability of the edgel distribution in the image. The only difference is that the MDL principle has information-theoretic roots, while the principle of non-accidentalness has its roots in psychological human tests.

Combining perceptual grouping found by the Gestaltists with the above grouping frameworks opens the way for computational solutions to perceptual grouping. Early computer vision work, some of which is discussed in the above paragraphs, has already started to explore perceptual grouping. However, we will now take a more organized approach and review the work systematically. Sarkar and Boyer have a nice overview of perceptual grouping work in their 1999 editorial [9]. They categorize perceptual grouping work according to two dimensions: the complexity of features that are being grouped (points, edgels, regions, etc.) and the dimensionality of the space in which the features live (2D, 3D, or one of these with an additional motion dimension). We will mainly concentrate on work with 2D images without motion, though some work in 3D is also relevant.

Basing our categorization on the work of Sarkar and Boyer [9], we break our upcoming review into low- and mid-level perceptual grouping. The categories' boundaries are somewhat vague, but in general terms, low-level grouping corresponds to grouping pixels and edge points into larger components, such as regions and edge fragments. This level corresponds to the signal level from Table 1 in [9]. We also choose to include more complex primitives at this level, such as longer contour fragments composed of image edgels (primitive level in [9]). Mid-level grouping (structural level in [9]) corresponds to recovering higher-level components, such as closed regions corresponding to objects or object parts, or high-level geometric primitives such as smooth curves or ribbons. The input features for this level commonly consist of edge fragments and regions extracted with lower-level methods. The next sections will systematically review perceptual grouping work in these two categories, illustrating the perceptual grouping rules employed, as well as the applications for using the recovered groups.

2.3 Low-level grouping

Following the taxonomy of perceptual grouping work in [9], low-level perceptual grouping refers to grouping low-level image features, like pixels or edgels, using local cues, such as appearance similarity and continuity. The resulting groups correspond either to image regions or to edge chains and can provide a good starting point for object recognition. The ideal situation would be for the extracted regions or contours to correspond to whole objects, but it is unrealistic to hope to obtain these in a purely bottom-up fashion[2]. Therefore, the resulting low-level primitives are mainly used in the following two ways:

- Grouping - Working with low-level primitives instead of pixels simplifies the complexity of high-level vision tasks.
- Feature extraction - Low-level primitives provide a better scope for feature extraction and support the extraction of higher-level image features.

In the following sections, we review the work in region segmentation and edge detection, and show how they can be applied for further segmentation and object recognition.

2.3.1 Region segmentation

The goal behind region segmentation is to be able to group pixels into meaningful regions without the a priori knowledge about the scene. Such a task is very difficult even in the case of grouping points in range images [3], let alone grouping pixels in 2D images. Given the absence of high-level knowledge, most low-level region segmentation techniques resort to defining a local appearance-based pixel affinity for neighboring pairs of pixels and cluster the pixels based on this affinity. The segmentation methods mainly differ in the way this affinity is defined, as well as in the methods used for clustering.

[2]Note that while obtaining a single correct object segmentation bottom-up is still considered unrealistic, recent developments in figure/ground segmentation provide methods to obtain a small number of object hypotheses that capture all the objects in the image with high probability [14].

The simplest and fastest approaches define a very local affinity (usually just in the 4- or 8-connected neighborhood of a pixel) that is based squarely on the difference in local pixel appearance. Greedy or local clustering algorithms are then used to obtain the regions using this affinity. Due to their locality and lack of top-down regularization, the methods in this category are quite sensitive to local appearance changes. They work well for objects with homogeneous appearance, but can result in significant undersegmentation (bleeding) if this is not the case. The watershed algorithm [102] is one of the first examples of such a method. The authors provide a fast algorithm for image segmentation, where a grayscale image is treated as a surface that is slowly immersed in water. Under such a definition, the "dams" separating the minima of the image would correspond to region boundaries between the resulting regions. Even though this method is extremely fast, it suffers from the aforementioned bleeding effects. Moreover, in its original form it is applicable only to grayscale images, though it is possible to extend it to work with other image modalities.

Comaniciu and Meer [19] describe a mean-shift clustering approach to image segmentation. Image pixels are clustered using the mean-shift algorithm, where the neighborhood for mean-shift is defined both in terms of the local spatial neighborhood of each pixel and its color similarity to other pixels. Finally, Felzenszwalb and Huttenlocher [31] describe a simple greedy graph clustering algorithm that is popular due to its simplicity, speed, and competitive results. The pixels are grouped in a hierarchical manner using a local color affinity defined over the 4-connected neighbors of each pixel. A smartly chosen heuristic is used to determine whether to group the segments in question based on the degree of their internal similarity compared to the degree of their between-segment similarity. It is this condition that allows the authors to employ a greedy grouping technique and thus provide an efficient solution to the segmentation problem.

To cope with the limitations of greedy segmentation algorithms, there was a move to introduce more global solutions to the problem. Wu and Leahy [109] were the first to pose image segmentation as optimization of a global cost defined on undirected graphs. However, the technique had the undesirable effect of producing many small isolated homogeneous segments. This was one of the motivating factors for one of the most popular segmentation techniques today - the Normalized Cuts framework of Shi and Malik [93]. The authors pose the segmentation problem as the minimization of the normalized cuts cost that pushes the segments to be balanced and thus minimizes bleeding effects. While the affinity between the neighboring pixels is still a simple local function of appearance similarity, the authors resort to global methods for optimizing their NP-hard cost and provide a spectral clustering-based approximation to the global optimum. Though the resulting regions are shown to be more robust to bleeding, such balanced regions may not be suitable for all domains, since often the objects in the image are of different sizes. Moreover, the spectral clustering method is computationally expensive and cannot run on very large images. Finally, since it is still based on purely local pixel affinities, it cannot be expected to extract objects bottom-up based purely on this low-level grouping criterion. A different global method based on ratio cuts is introduced in [104]. Similar to normalized cuts, the normalization in ratio cuts enables the method to recover large connected regions with minimal amount of bleeding. For general graphs, finding a ratio cut is NP-hard problem. In this case, however, the graph is defined over regions and the edges are naturally defined based on region adjacency. Since the resulting graph is planar, the authors provide a polynomial algorithm for solving the problem. However, they still report running times that are on the order of minutes or hours per image.

Whereas the techniques in the previous paragraph employ more sophisticated clustering algorithms, they still use simple pixel affinities. In [111], Yu and Shi not only provide a more principled approach for solving the Normalized Cuts problem, but also employ a more sophisticated pixel affinity function based on “intervening contours”. It allows the assignment of better affinities to pixels in large neighborhoods based on the presence or absence of edges between them. In [33], Fowlkes et al. further improve the pixel affinity by using several features to compute it, as well as learning it by training a logistic classifier over a set of images.

These examples illustrate the evolution of low-level region segmentation techniques from greedy clustering algorithms using simple grayscale affinities to global clustering algorithms using elaborate affinities that are based on multiple image cues. Region segmentation is the subject of current research, with ongoing work on new cost functions, better optimization methods and stronger affinities. Despite the advances in low-level region segmentation, the methods in this category are united in their reliance on purely local appearance. This property makes it unlikely that, barring cases of homogeneous appearance, large semantically meaningful segments can be extracted using such approaches. Nevertheless, region segmentation is of the utmost importance and proves to be essential as a preprocessing step for many higher-level vision tasks.

Instead of having to deal with millions of pixels, it is now common practice to use small local regions as atomic primitives. Images are typically preprocessed with region segmentation tuned towards oversegmentation. The result is a much smaller number of primitives, typically numbering in the hundreds of regions. Further perceptual grouping rules are applied to group these elements bottom-up, or scene knowledge is used at this stage to detect objects or categories. In [80], Ren and Malik coined the term superpixels for such region oversegmentation. Instead of grouping pixels, they grouped superpixels in an attempt to find "good" segments, where goodness was a learned function of perceptual grouping features (such as homogeneity in appearance and closure). Employing superpixels not only significantly reduced the complexity of their task, but also provided an ideal scope for computing the features used to evaluate goodness. Later, superpixels started to become popular for image labeling problems. In such problems, the task of segmenting the image into K categories was defined as assigning a label to every superpixel. For example, He et al. [36] used superpixels to segment the image into a number of categories, such as vegetation, sky, and buildings. Such methods start by computing robust appearance features for superpixels and analyze the correlation between superpixel features and labels. Though superpixel appearance features can be as simple as color histograms, they can go as far as encoding appearance using robust local descriptors, such as SIFT [59]. Once such correlation is learned, superpixel labels are determined based on superpixel features and context.

Nowadays, many researchers make use of region segmentation in their work. Some frameworks [40] go beyond the use of single scale superpixels, relying on multiscale region segmentation for scene analysis. However, the underlying principle of using region segmentation remains constant across all approaches, restricting the use of low-level region segmentation to preprocessing stages rather than considering it a goal in itself.

2.3.2 Edge detection

Instead of grouping pixels, an alternative approach to finding coherent regions in an image is to trace their boundaries. Starting with the detection of image edges, representing locations of strong change in color or texture, one would attempt to link them together into closed contours. Not all edge detection systems attempt to group edges into closed contours. Most approaches limit themselves to the extraction of long continuous image curves (edgels), leaving their further grouping to higher-level processing stages. Similar to region segmentation, edge detection relies on the computation of local appearance similarity with edges being detected at locations of heterogeneous appearance, followed by grouping them into chains. The methods differ in the complexity of their similarity measures and in the scope of the grouping of individual edge points.

Though edge detection has been a part of the vision community since its infancy, Canny's [13] edge detection approach is perhaps the most enduring edge detection algorithm. Though the basic principle of edge detection lies in thresholding the pixels based on a function of image gradient, Canny's detector adds additional components to the pipeline. Through noise reduction, non-maxima suppression and edge following using hysteresis thresholding, Canny's edge detection algorithm outperforms its predecessors.

Canny's original algorithm detects edges at a single scale. However, in the image, edges occur at various scales. They can be very sharp, for in-focus objects with clear boundaries, or they can be blurry, for edges of shadows under multiple light sources, for example. Several authors have attempted to tackle this issue. Among them, Jeong and Kim [45] pose the edge detection problem of finding the scale that results in the best reconstruction of an image, while assuming that scale changes smoothly across the image. Other approaches for automatic scale selection of edges include Lindeberg [53] and Elder and Zucker [28].

Finally, just as the pixel affinity in region segmentation methods did not have to be only a function of color, edges can also be based on additional features. In [63], Martin et al. combine brightness, color and texture measures into a probability of boundary (Pb) measure. Pb is computed by training a logistic classifier over the aforementioned features. It significantly outperforms techniques that are based solely on image intensity. We would argue that this method and its extensions [61] are currently the standard approaches for obtaining image edges.

Having identified image edge points and locally grouped them into short edgels, the results can be used as a basis for extracting longer parametric curves, finding closures and building better larger-scoped features for object recognition. Many approaches start with edgels as the basis for detecting true occluding contours, attempting to fill gaps and remove edgels that do not correspond to object contours. Shashua and Ullman [92], for example, extract long, smooth image curves. It is precisely the use of edgels that enables the authors to employ the perceptual grouping cues of continuity and curvilinearity. One counter example that starts with plain edge pixels and nevertheless presents astounding results is the work of Williams and Jacobs [106]. Unlike [92], the authors use a more principled approach to fill gaps, computing a distribution over illusory contours. However, the system is tested on artificial images rather than exemplifying if it can correctly fill in the gaps in a real image. Some researchers attempt to extract larger-scope semantically meaningful edge chains. For example, Nelson and Selinger [71] group edgels into parametric curves, which are then used for object recognition. To be able to operate in more realistic imaging scenarios, edge grouping can employ additional grouping rules and better computational frameworks. Ren and Malik [82] also use edgels as a starting point for more complete contour extraction, but add higher order constraints such as T-junctions and object-level priors, formulating their problem as inference in a Conditional Random Field

(CRF) network. In Ren et al. [81], both superpixels and the edges between them are used as nodes in a CRF, providing even more constraints and context for contour detection. Finally, similar to spectral clustering approaches for region segmentation, Zhu et al. [112] propose to solve this hard grouping problem by embedding the edge fragments into polar coordinates such that closed contours correspond to circles in that space. Furthermore, the authors show that their approach is not restricted to the extraction of closed contours and results in a state-of-the-art method for generic contour detection.

Grouping short edge fragments or individual edge pixels instead of working with all image pixels decreases problem complexity and simplifies the application of some shape-based perceptual grouping rules. However, there are other advantages of using edges. Ever since the seminal paper by Lowe [59], SIFT and other variations of keypoint features [65] were heavily used for object recognition. These features encode the appearance in a local patch and can be used for robust matching. Similar constructions can also be used to encode local shape, which is where image edges come into play. For example, Mori et al. [69] use the shape context feature for object recognition. Originally coined by Belongie and Malik [2] for matching shapes, shape context in [69] is constructed by sampling image edge points. Instead of using individual edge points, Ferrari et al. [32] construct a higher-order feature and use edgels to compute their kAS descriptor. The descriptor robustly encodes the relative position and orientation of a group of k straight edgels, making it useful for more generic object recognition tasks. Finally, Nelson et al. [71] use even higher-level primitives (image curves) in their object recognition framework, though they resort to working with somewhat simplistic images in order to extract such features bottom-up.

Similar to region segmentation, the state-of-the-art is continuously being extended, resulting in approaches like [61] for edge detection or [112] for contour grouping. But the image primitives and the grouping rules used in both region segmentation and edge detection methods usually prove to be insufficient for the kind of grouping needed for generic object recognition. Building up on the primitives extracted at this level, we need to employ stronger grouping rules, giving us larger-scope, more meaningful groups. This brings us to the subject of mid-level grouping.

2.4 Mid-level grouping

Unlike most current recognition methods that work with real world scenes but constrain the recognition problem, early vision researchers set their sights on very generic object recognition [72, 11, 58]. This level of recognition typically modeled objects as compositions of more abstract image features. The models ranged from a collection of smooth surfaces or long parametric curves used for representing scene surfaces, to representing objects using closed contours or with a small set of abstract parts. However, extracting such features bottom-up from a real world image proved to be a significant challenge, forcing most methods to bridge this representational gap by working with simplified images. But even simplified images present a challenge for the extraction of these primitives. Nevertheless, the early vision community illustrated that using higher level image primitives and employing stronger, mid-level grouping rules and models for scene analysis is essential for generic object recognition. Understanding these mid-level rules and models, together with new algorithms and machine learning techniques would help us push perceptual grouping into the realm of real world imagery.

One of the most general models assumed for mid-level grouping is that the image is a projection of a scene made up of piecewise smooth surfaces [12]. Such a model is flexible enough for most scenes, but is hard to extract even from range images [3] where one has to distinguish perturbations in surfaces from surface discontinuities. The situation is much more complex in the case of 2D images, where appearance is not only a function of surface geometry but also depends on illumination and albedo. With the exception of a small number of approaches, such as Hoeim et al. [41] who address a constrained version of this problem, the majority of the community has tended to avoid this difficult problem. More constrained models need to be defined if we hope to extract meaningful components bottom-up, without relying on object-level knowledge. Two of the most prominent mid-level grouping rules fitting this bill are closure and symmetry. In this section, we will mainly concentrate on these rules and their application to object recognition.

2.4.1 Symmetry

Following the recognition-by-components insight of Biederman [5], symmetry can serve as a powerful basis for defining object parts. While not all objects can be represented using a small set of symmetric parts, various symmetry-based part vocabularies were shown to be sufficient to represent a large enough subset of objects. Restricting objects to be modeled using such vocabularies facilitates generic recognition by providing not only a necessary abstraction for object representation, but also the ability to extract object parts bottom-up.

In 3D, symmetry has served as a basis for a rich body of object recognition, with work applied to various recognition domains and experimenting with different part vocabularies. For instance, Binford [6] used generalized cylinders to represent object parts. Generalized cylinders can be defined with a space curve that acts as the axis and a cross section that is swept along the axis. As the name suggests, it is a generalization of a right circular cylinder (commonly refered to as cylinder) for which the space curve will be a straight line segment and the cross section will be a circle normal to the axis at any point. Other examples of symmetry-based parts employ more restrictive vocabularies, such as superquadrics [77] and geons [4]. These examples show that symmetric parts can provide a powerful representation for a significant collection of objects. We will stop here and not detail the methods for extraction of these volumetric parts from range images, as we are interested in work in 2D image domains.

In 2D, symmetry has been present in the vision community even longer. Blum's medial axis transform (MAT) [7] computed the skeleton of an object given its silhouette. The skeleton produced by the MAT is the set of all center points of the maximally inscribed discs of the given shape. The natural way to compute such skeletons is to compute the distance transform of the shape and find the locus of local maxima in the distance transform. Skeletons have been widely used in the recognition community. For example, Siddiqi et al. [97] computed the singularities (shocks) in the skeleton formation and organized them into a shock graph. Shock graphs for objects were then matched using a new graph matching algorithm comparing two different objects. Shock graphs have been used by others using different matching techniques. Some examples include Pellilo et al. [76] who matched shock graphs using association graphs, and Sebastian et al. [89] who computed a graph-edit distance-based measure between a pair of shock graphs.

One issue with skeletonization techniques is that the resulting skeleton is sensitive to small perturbations in the shape of the object. Some skeleton extraction algorithms ignore this problem, while others try to address the issue by smoothing either the original contour or the resulting medial axis. Recently, Macrini et al. [60] proposed a more principled approach for the regularization and abstraction of a skeleton by analyzing its ligature structure. While it is a significant step toward ultimate generic object recognition using part-based skeleton representations, an even more serious weakness of skeleton-based approaches is that a silhouette of an object is required as input, effectively requiring the object to be segmented ahead of time. Segmenting out objects bottom-up in a purely unconstrained fashion is, in general, an unsolved problem, and we will therefore continue this section by reviewing symmetric part extraction that does not require objects to be segmented a priori. Such approaches can be grouped into these three categories:

- Filter-based part extraction
- Contour-based part extraction
- Model-based part extraction

The remainder of this section reviews part extraction in each of these categories.

The filter-based category refers to methods that attack the problem by passing the image through a bank of low-level filters and analyzing the filter response to find symmetric regions. These regions, referred to as blobs or ridges, provide a crude segmentation of an image and could signal the presence of objects or parts of objects in the image domain with application to object recognition and/or object tracking. In other domains, such as histogram analysis, blob descriptors can also be used for peak detection with application to segmentation. Another common use of blob descriptors is as primitives for texture analysis and texture recognition. In more recent work, blob descriptors have found increasingly popular use as interest points for wide baseline stereo matching and to signal the presence of informative image features for appearance-based object recognition based on local image statistics.

The use of filter-based symmetry detection for multiscale abstract part extraction was proposed by Crowley [24], who detected peaks (rotational symmetries) and ridges (elongated symmetries) as local maxima in a Laplacian pyramid, linked together by spatial overlap to form a tree structure. Object matching was then formulated as comparing paths through two trees. Shokoufandeh et al. [94] proposed a more elaborate matching framework based on Lindeberg's multiscale blob model [54] (an extension of Lindeberg's older work [53]). Finally, Levinshtein et al. [52] demonstrate that multiscale blobs and ridges can serve as a powerful basis for automatically learning part-based generic shape models.

A more recent application of filter-based symmetric regions is as a basis for defining robust feature descriptors for keypoint features for stereo matching or exemplar object recognition. The original keypoint approach, still very popular today due to its robustness and applicability to many problems, is the approach of Lowe [59]. Motivated by the work of Lindeberg [53] where blobs are detected as the maxima of the scale-normalized Laplacian of the image, Lowe uses a difference of Gaussians (DoG) to recover the position and scale of his SIFT features. DoG is a good approximation to the Laplacian of an image and can be computed more efficiently. In [66], Mikolajczyk et al. compare a number of these and other region extractors. Alternatives to using a Laplacian or a difference of Gaussians include the determinant of the Hessian or the Harris operator for blob detection. The last detector uses the second moment matrix instead of the Hessian for blob detection. All of the aforementioned blob detectors can be adapted to detect ridges using a procedure described by Mikolajczyk and Schmid [64].

The latter is arguably the more successful application of filter-based symmetric regions. Current approaches employing interest points serve as a good example of a domain where an accurate delineation of local symmetric regions may not be necessary. In addition to being very efficient, coarse filter-based symmetry detection proves sufficient for the robust extraction and scope representation of stable image points in such domains. The same is not true for generic object recognition, where large parts with possibly heterogeneous appearance need to be extracted. Simply detecting parts as local maxima in a set of multiscale filter responses leads to many false positives and false negatives, suggesting that successful part extraction requires paying closer attention to image contours.

The second category of symmetry detection approaches is comprised of methods that find symmetry by grouping image contours. Most of the techniques in this category are similar in that they start with detecting edges and then group them into one or more symmetric regions. Unlike the approaches in the previous category that overcome the complexity of finding symmetric regions through the application of coarse filters, all contour-based approaches are faced with the task of finding symmetry in a vast collection of image contours. Coping with the overwhelming complexity of such a task forces contour-grouping techniques to consider a variety of grouping approaches and impose additional constraints on symmetry.

Early contour-grouping approaches, motivated by the early work in skeletonization, attempt to extract skeleton-like representations from real images. An early example of such a technique is the work of Brady and Asada [10], showing how smooth local symmetries (SLS) can be detected. Unlike the MAT-based skeletons that are composed of centers of maximally inscribed discs, SLS skeletons are made of midpoints of line segments forming the same angle with both sides of the bounding contour. However, while the definition of a skeleton differs between the two approaches, they are visually and conceptually very similar. In contrast to skeletonization approaches that assume the availability of an object's contour, Brady and Asada extract closed contours using the Canny edge detector. As their technique depends on precise contour tangents, they represent the extracted contour using circular arcs and straight line fragments, instead of working with the raw edges. Nevertheless, the final set of local symmetries can be quite noisy and fragmented. Connell and Brady [20] describe not only how to “clean” up this noisy set of symmetries by eliminating some and grouping others, but also introduce a system that uses the resulting symmetric groups to model objects. To group the local symmetries, they use a set of perceptual grouping rules including proximity of local symmetries, their continuation, change in cross-section length, and various other heuristics, some of which are similar to the principles used for skeleton analysis in [97]. In a further extension, Ponce’s [79] theoretical paper analyzes various skeleton formulations and shows, for example, that the MAT skeletons are a more constrained set than the SLS skeletons of Brady and Asada. Ponce’s analysis also results in yet another skeleton definition, accounting for skewed symmetries that can arise from 3D projection. However, despite the improvement of this and Connell’s approach on the original technique of Brady and Asada, all three methods avoid the complexity of grouping image edges arising from real scenes by working with simplified imagery. Working with more realistic scenarios would introduce significantly more edgels, making the extraction of closed contours, as well as the detection and grouping of symmetric regions, far more difficult.

Diving more into the realm of real world imagery, Saint-Marc et al. [84] show how symmetries can be extracted by building on a B-spline representation of image contours. They first illustrate how B-splines can be extracted by starting with standard edge detection (such as Canny), performing edge linking, and concluding with fitting splines to edge chains. Given this simplified image representation, they show how a variety of symmetries can be extracted by imposing constraints over pairs of B-splines. Unlike previous approaches, restricted to the detection of local symmetries, Cham and Cipolla [18] employ a similar B-spline representation, and provide a very fast and simple method for global skewed symmetry axis extraction. The Hough transform is first used to efficiently hypothesize groups of points that form local skewed symmetries. Once such groups are available, the hypotheses are checked using a global symmetry criterion proposed by the authors. In [17], the same authors describe a different approach for solving the same problem, this time using their measure of “geometric saliency”. In [55], Liu et al. propose a more principled approach to symmetry axis extraction. They formulate the symmetry axis finding problem as finding the best sequence of pairs of points in the image and solve the problem using Dijkstra's algorithm. Their method does not require the contour to be available a priori, but it does require an initialization with an initial pair of points and produces an open boundary. Moreover, no preprocessing into image curves is performed as done in some of the other methods. Therefore, to cope with the complexity of finding the best sequence among all pairs of points in the image, the authors resort to using hashing techniques.

Note that all the approaches described so far extract global unbounded symmetry axes, or extract symmetric sections by analyzing closed contours or pairs of image curves. The global symmetry recovered by the first set of approaches is useful for scene analysis but is insufficient for generic object representation. On the other hand, the kind of symmetries extracted by the second set of approaches are ideal for object representation, but are shown to operate on a very restrictive set of images, usually containing a single object with homogeneous appearance. In real images containing multiple objects with heterogeneous appearance imaged against a complex background, it is unlikely that meaningful closed contours could be recovered bottom-up by the suggested approaches or that objects (or their parts) would be bounded by a pair of extracted image curves. The more likely scenario is that an object's contour would correspond to far more than a single closed image contour or a pair of image curves, requiring more elaborate grouping strategies.

Ylä-Jääski and Ade [110] provide such a method by finding partial symmetries between straight edge segments and then grouping them together into complete axial descriptions. In [98], Stahl and Wang take a similar approach but use a much more principled grouping algorithm based on ratio cuts to obtain their symmetries. The authors start by extracting linear edge segments and construct symmetric quadrilaterals which are then used for grouping. The algorithm finds the best sequence of quadrilaterals that minimizes the gap in boundary edges, while maintaining a smooth symmetry axis as well as a compact internal region for the resulting symmetric part. Even though grouping is polynomial in the number of graph edges, the number of quadrilaterals, as well as the possible ways of filling the gaps between them, are prohibitive. This forces the authors to resort to heuristics to reduce the complexity of the problem. They also provide an iterative approach for extracting multiple symmetric regions, by finding the best region and repeating the process after removing all the quadrilaterals associated with that region. Still, quadrilaterals typically number in the thousands and the running time is on the order of several minutes per image.

Although great advances were made in contour-based symmetry detection, the complexity of contour grouping remains the main difficulty faced by all methods. Early work reduced this complexity by constraining the symmetry representation or working with simplified images, while recent approaches work under less constrained scenarios but have to rely on suboptimal grouping algorithms and/or grouping heuristics. Nonetheless, being very data-driven in nature, such approaches prove to be much more suitable for object analysis than top-down coarse filtering techniques, despite their problematic complexity.

The third category for bottom-up symmetry detection, called model-based grouping, refers to methods that employ a top-down deformable shape model during the extraction process. In fact, any filter-based technique can be seen as a model-based approach, as it detects parts by employing a top-down coarse shape model. Unlike the case of filter-based techniques, where model detection is approximated by analysing low-level filter responses, the models here are matched against image contours. Moreover, the notion of having a model is made much more explicit, with models ranging from complex parametric symmetric shapes to going beyond symmetry, and representing arbitrary shape. While most techniques under this category go as far as using specific object shape models, some address the domain of perceptual grouping. That said, even the more specific object detection approaches can sometimes be modified to work with more generic shape models, and thus provide useful insight into perceptual grouping.

Work with deformable shape models has its roots in Kass et al. [47]. Their model restricts the shape to have a smooth boundary with strong underlying image edge support. Given such a weak shape model, the method is more suitable to be used as a low-level grouping approach described in a previous section. Moreover, the method requires a rough global initialization of the model prior to image alignment. Unlike [47], Cootes et al. [21] use object specific deformable models in their work. However, the model still needs to be initialized close to the object for the approach to work. Examples of more practical techniques, attempting to find multiple instances of a deformable shape models automatically, are [78, 85, 88]. Pentland [78] solves the problem using a filter-like approach. Representing object parts using 3D superquadrics, part templates are constructed a priori for different part projections and different settings of the deformation parameters. The algorithm proceeds by accumulating evidence for each of the templates at different image locations. High-scoring templates, and consequently object parts, are selected using a global optimization procedure. Recently, Sala and Dickinson [85] employed a similar approach for symmetric part detection. They too represent 2D parts as projections of a small set of deformable 3D part vocabulary, but unlike Pentland who worked with range images, avoiding the additional complexity of dealing with heterogeneous object appearance, Sala and Dickinson are able to extract symmetric parts from real 2D images. In contrast to previous approaches, Sclaroff and Liu [88] define their shape model as a deformable polygon, and while they use it for segmenting specific objects, the model can be easily adapted for symmetric part extraction. They pose the grouping problem as finding a subset of oversegmented image regions that satisfy their shape model. However, the authors show that despite pruning the search space using various heuristics, a brute force approach still exhibits a prohibitive grouping complexity, forcing an approximate solution of the problem by using a greedy algorithm. There are many other approaches that employ deformable shape priors, but similar to contour grouping techniques, all such methods are faced with prohibitive algorithm complexity, this time arising from matching models to image data. Overcoming this issue, while accurately detecting symmetric regions, is the subject of ongoing research.

In summary, we reviewed three categories of symmetry detection approaches that illustrate the tradeoff between fast but inaccurate methods that rely on low-level filter responses, to high-complexity contour grouping or model-based approaches that are much more data driven.

Besides the individual weaknesses and strengths of the aforementioned techniques, most of them share another recurring weakness. With few exceptions, symmetric parts are usually not grouped together. For example, in skeleton-based approaches, a skeleton already corresponds to a whole object. However, in order to use it for efficient object recognition, it needs to be parsed into stable branches that correspond to object parts - a challenging task as skeletons are sensitive to small shape perturbations. Unlike the case of skeletons, bottom-up symmetry extraction techniques result in a disconnected set of symmetric parts. While grouping them together is perhaps easier than grouping low-level features, such as pixels, into whole objects, it is still the subject of ongoing research and not commonly addressed. Whole object skeletons or collections of unrelated symmetric parts undoubtedly simplify generic scene analysis, but symmetry alone is not enough. Objects parts need to be related and/or grouped together, calling for additional perceptual grouping rules to be used. One such rule is described in the next section.

2.4.2 Closure

The law of closure is another strong perceptual grouping rule, enabling us to form large-scope semantically meaningful components without the use of an object model. Unlike continuity and curvilinearity, that facilitate the extraction of smooth curves with strong edge support, closure is a more global constraint which results in finding **closed** contours in an image. A famous example of closure is shown in Figure 2.2, where we perceive a “hidden” triangle despite the absence of supporting image edges. In addition to relying on continuity and curvilinearity, the triangle is perceived because it corresponds to a single closed contour. Given such an intuitive concept of closure, let us now move on to a more precise mathematical definition.

There is no consensus on the exact definition of closure in the computer vision community. Most agree on two important properties for any closure cost: 1) the cost should be monotonic to the amount of contour gap (the lack of image edge support), and 2) the cost should not overly penalize larger regions or longer contours (the absolute amount of contour gap is likely

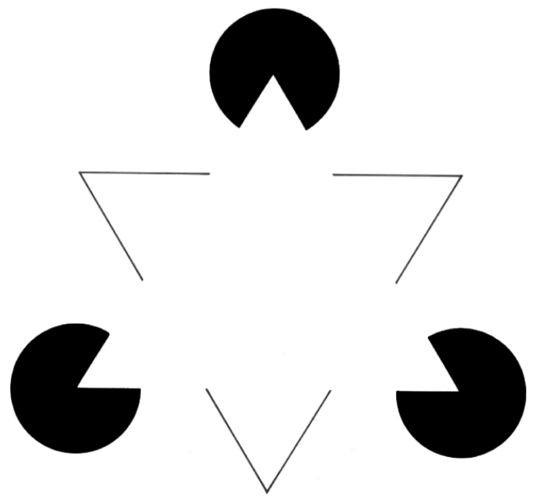


Figure 2.2: Kanisza Triangle. Despite contradictory information, the law of closure forces a perception of a closed object (triangle in the middle).

to increase with increase in region size or contour length). These two basic properties induce the closure cost of a closed contour to be a ratio of contour gap to a measure of contour size. However, using gap alone to evaluate closure is insufficient in the presence of many distracting contours. Real occluding contours exhibit additional properties, such as being smooth and continuous. Therefore, most closure formulations incorporate at least these additional properties. Other differences in cost formulations arise through different definitions of gaps and various measures of contour size. For instance, Elder and Zucker [26] show that a cost based purely on the total boundary gap is insufficient for perceptual closure. Though objects with smaller gap on their boundary are more easily detected by human subjects, this is not the only factor contributing to a successful perception of whole objects. They show that the distribution of gaps along the contour is also very important. Instead of a sum of the length of all gaps, they show that a Euclidean norm is a much better measure. Under such a measure, a single large gap would be significantly worse than many small gaps amounting to the same total length. In addition to different measures of gap, there is also the question of what is contour size? Most techniques resort to normalizing contour gap by its length, while some show that normalization by area instead of length results in better performance [99]. While area normalization helps to incorporate another important perceptual grouping cue, i.e., compactness, into the closure cost, not all objects are compact, in which case a cost based on length normalization could work better. Similar arguments could be made for other terms of the cost function, making the search for the "true" closure cost a subject of current research.

In conjunction with the issue of a precise closure definition, there is also the issue of finding closed contours by minimizing a closure cost. Since closure is a global property, the main problem associated with its use is being able to overcome the high computational complexity arising from considering a multitude of all possible closed contours in an image. Most approaches simplify the problem by starting from short edge fragments instead of raw pixels. Despite this simplification, a brute force approach for computing closure would have to consider all possible subsets of these edgels, resulting in exponential complexity. This forces most methods to impose further constraints on their closure cost or resort to the use of heuristics and/or greedy optimization approaches. In [44], Jacobs attempts to extract closed convex contours by grouping straight line segments. The grouping problem is formulated as a brute force search for subsets of line segments. Various heuristics are introduced to prune the search tree, the main one being a hard threshold on the measure of closure that the author proposes, not only for the final subsets but for all intermediate ones as well. The resulting search algorithm is still exponential in the number of line segments in the worst case, but is shown to have a reasonable average time performance in realistic scenarios. A less restrictive constraint is that of compactness, which can be attained by normalizing the gap by area (Estrada and Jepson [29, 30]), where the authors perform a brute force depth first search for the solution in a sea of edgels, constrained by several heuristics to make the search feasible. Elder and Zucker [27] provide a graph-theoretic solution, modeling the pairwise affinity between two adjacent contour fragments and reducing the problem of finding closed contours to finding shortest paths in a graph using Dijkstra's algorithm. Williams and Hanson [107] addressed the problem of perceptual completion of occluded surfaces, formulated as the problem of computing a labeled knot-diagram representing a set of occluded surfaces from observed image contours. While formulated as an elegant combinatorial optimization problem, for which an optimal solution was available, the approach was not tested on real scenes.

The above approaches all face the prohibitive complexity of grouping edgels into closed contours, forcing methods to impose additional constraints on the closure cost or use suboptimal grouping algorithms. Zhu et al. [112] propose a more general solution, providing not only a more global approach, but also the option to extract long image curves as well as contour cycles. The method works by embedding the edge fragments into polar coordinates such that closed contours would correspond to circles in that space. Computing several complex eigenvectors of the embedded edgels allows the authors to find multiple cycles in a principled way (one cycle per eigenvector). Jermyn and Ishikawa [46] show how closed contours can be extracted by optimizing a ratio cut cost. While in general, optimizing ratio cuts is NP-hard, working with a 4-connected planar graph over image pixels allows the authors to find closures in polynomial time. Wang et al. [103] take a similar approach, but define their graph over contour fragments, facilitating the computation of a better gap measure. Restricting closures to be alternating gap/non-gap cycles of fragments results in a globally optimal, polynomial algorithm. Finally, since normalization by area instead of contour length promotes compactness, which is often a desirable property, Stahl and Wang [99] modify the approach in [103], measuring area through the application of Green's theorem.

Working with edge fragments, instead of pixels, undoubtedly decreases the complexity of finding closures. However, the requirement of finding not simply long chains of contours, but closed cycles, puts an additional burden on contour-based grouping approaches. This added complexity can be avoided if closure finding is reformulated as the problem of grouping regions or fitting a top-down deformable shape model to an image. While such techniques are common when object-level knowledge is employed, such as labeling images using superpixels or using active shape models, they are seldom encountered in perceptual grouping literature. The work of Kass et al. [47] is one example of fitting a generic deformable shape to an image, however the cost in [47] contains many local minima corresponding to undesirable solutions and requires the manual initialization of the model close to real object boundaries. As an example of region-grouping techniques, one can take the work of Sclaroff and Liu [88], already reviewed in the previous section. Though the models in [88] employ object-level knowledge, it is conceivable that the same shape representation and algorithm could be used to recover more generic shapes. In perceptual grouping, one of the few region-grouping methods whose goal is to find closure is the work of Jermyn and Ishikawa [46]. However, the authors work with the most basic of region primitives, pixels, hindering their ability to compute a robust gap measure and to employ useful perceptual grouping rules such as continuity and curvilinearity. One of the most recent advances in this domain is our own work in Chapter 4, illustrating the benefits of working with small regions (superpixels) instead of edge fragments. Nevertheless, region grouping has its own downfalls, especially when the underlying regions are undersegmented. Whether or not this is a good alternative to contour-based closure extraction is arguable at this point, with the winning technique being the one that is able to extract most objects in the scene in a reliable fashion.

2.5 Conclusions

Perceptual grouping is an astounding human ability. In this review we have illustrated the high-level principles behind this ability and motivated the need for perceptual grouping in computer vision. We reviewed approaches starting from local grouping rules, such as proximity and continuity, operating on low-level image primitives, to more global rules, such as closure and symmetry. Low-level grouping techniques, while usually being fast and efficient, result in very local image components which are insufficient for generic recognition. On the other hand, mid-level grouping is shown to contribute towards the extraction of meaningful, large-scope components, facilitating generic recognition. However, methods employing these rules suffer from the prohibitive complexity of assembling low-level primitives into larger groups.

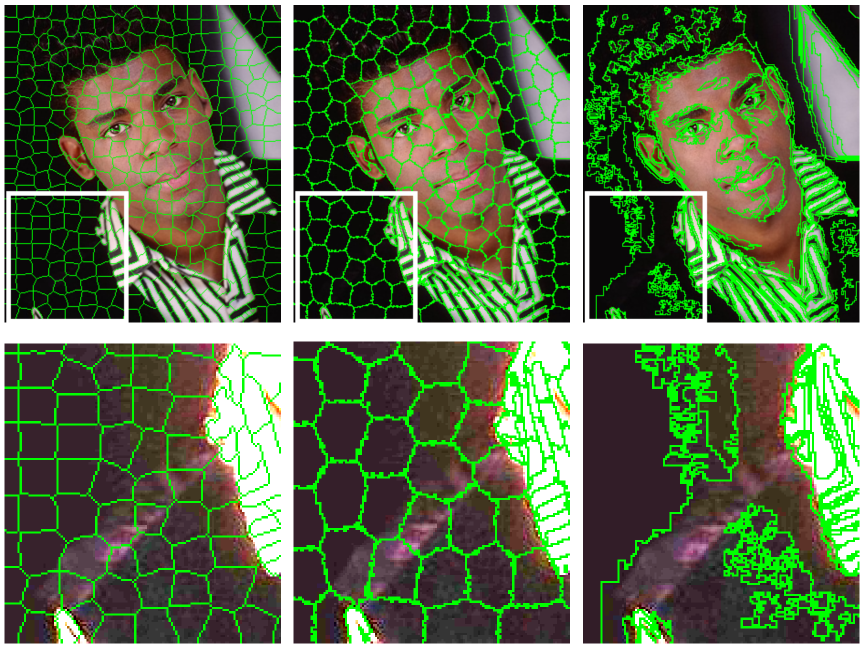
Between low and mid-level grouping the latter is more useful for generic object recognition; however, low-level primitives, such as superpixels or edge fragments are shown to be instrumental for many vision tasks. In fact, most successful mid-level grouping approaches rely on low-level primitives instead of operating on raw pixels, suggesting that continuing advances in all levels of grouping are essential. Moreover, even the mid-level grouping results achieved by current approaches are usually insufficient for the amount of grouping required for generic object recognition. Most current approaches focus on a single cue, such as closure or symmetry, but not both, providing promising results but failing to operate in a variety of different scenarios. Ultimate success in perceptual grouping lies in the integration of many perceptual grouping cues. Hierarchically building on local image primitives obtained through the application of low-level grouping rules, employing multiple mid-level constraints would propel perceptual grouping in computer vision to human capabilities.

Chapter 3

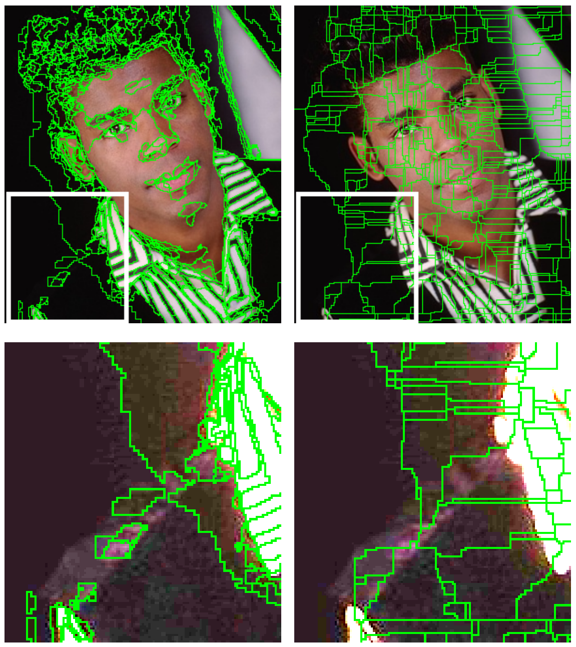
Turbopixels: Fast Superpixels Using Geometric Flows

3.1 Introduction

Superpixels [80] represent a restricted form of region segmentation, balancing the conflicting goals of reducing image complexity through pixel grouping while avoiding under-segmentation. They have been adopted primarily by those attempting to segment, classify or label images from labelled training data [37, 40, 41, 70, 80]. The computational cost of the underlying grouping processes, whether deterministic or stochastic, is greatly reduced by contracting the pixel graph to a superpixel graph. For many such problems, it is far easier to merge superpixels than to split them, implying that superpixels should aim to *over-segment* the image. Region segmentation algorithms which lack some form of compactness constraint, e.g., local variation [31], mean-shift [19], or watershed [102], can lead to *under-segmentation* in the absence of boundary cues in the image. This can occur, for example, when there is poor contrast or shadows. Algorithms that do encode a compactness constraint, including N-Cuts[111] and *TurboPixels* (the framework we propose), offer an important mechanism for coping with under-segmentation. Figure 3.1 shows the over-segmentations obtained using these five algorithms;



(a) (b) (c)



(d) (e)

Figure 3.1: Over-segmentations obtained with five algorithms: (a) TurboPixels (b) N-Cuts[111] (c) Local variation [31] (d) Mean-shift [19] (e) Watershed [102]. Each segmentation has (approximately) the same number of segments. The second row zooms in on the regions of interest defined by the white boxes.

the effect of a compactness constraint in limiting under-segmentation can be clearly observed in the results produced by *TurboPixels* and *N-Cuts*.

The superpixel algorithm of Ren and Malik [80] is a restricted graph cut algorithm, constrained to yield a large number of small, compact, quasi-uniform regions. Graph cut segmentation algorithms operate on graphs whose nodes are pixel values and whose edges represent affinities between between pixel pairs. They seek a set of recursive bi-partitions that globally minimize a cost function based on the nodes in a segment and/or the edges between segments. Wu and Leahy [109] were the first to segment images using graph cuts, minimizing the sum of the edge weights across cut boundaries. However, their algorithm is biased toward short boundaries, leading to the creation of small regions. To mitigate this bias, the graph cut cost can be normalized using the edge weights being cut and/or properties of the resulting regions. Although many cost functions have been proposed (e.g., [23, 46, 86, 104]), the most popular normalized cut formulation, referred to widely as N-Cuts, is due to Shi and Malik [93], and was the basis for the original superpixel algorithm of [80].

The cost of finding globally optimal solutions is high. Since the normalized cut problem is NP-hard for non-planar graphs, Shi and Malik proposed a spectral approximation method with (approximate) complexity $\mathcal{O}(N^{3/2})$, where N is the number of pixels. Space and run-time complexity also depend on the number of segments, and become prohibitive with large numbers of segments. In [91] a further reduction in complexity by a factor of $\sqrt{N}$ is achieved, based on a recursive coarsening of the segmentation problem. However, the number of superpixels is no longer directly controlled, nor is the algorithm designed to ensure the quasi-uniformity of segment size and shape. Cour et al. [22] also proposed a linear time algorithm by solving a constrained multi-scale N-Cuts problem, but this complexity does not take the number of superpixels into account. In practice, this method remains computationally expensive and thus unsuitable for large images with many superpixels.

There are fast segmentation algorithms with indirect control over the number of segments. Three examples include the *local variation* graph-based algorithm of Felzenszwalb and Huttenlocher [31], the *mean-shift* algorithm of Comaniciu and Meer [19], and Vincent and Soille's watershed segmentation [102]. However, as mentioned earlier, since they lack a compactness constraint, such algorithms typically produce regions of irregular shapes and sizes.

The above techniques are some of the more recent examples of region growing approaches, some of which date back to the 1970s (a good overview is provided in [114]). Typically, regions start growing from single pixel seeds, where the seeds are provided manually (through user input) or defined automatically (by considering every pixel a seed, for example). Regions are grown greedily by iterative addition of pixels and/or other regions based on various grouping heuristics, with some studies considering region splitting as well [42]. Grouping heuristics range from merging regions based on appearance similarity to merging regions that can be approximated by a small number of low-order polynomials [75]. Similar to the more recent approaches, early region growing methods have also struggled to incorporate more global knowledge into the segmentation framework through the use of better heuristics and merging methods.

The *TurboPixel* algorithm introduced in this chapter segments an image into a lattice-like structure of compact regions (superpixels) by dilating seeds so as to adapt to local image structure. It is similar in spirit to region growing approaches, but focuses on region boundary and uses level-sets for curve evolution. Computationally, the approach is rooted in the early curve evolution techniques in computer vision (e.g., [15, 49, 62]). In an approach that is similar in philosophy to the one we develop in this chapter, in [90] properties of the medial axis are used to modify the evolution of two simultaneously evolving contours, in application to carpal bone segmentation. In the reaction-diffusion space of [49], a constant motion (reaction) term was played off against a curvature term (diffusion) for shape analysis. This flow was subsequently adapted to the problem of segmentation in [62] and [15], via the inclusion of a multiplicative image gradient stopping term. These methods led to active contour models that could handle changes in topology in a natural way. Formal theoretical justification, as the gradient flows associated with particular weighted length or area functionals, followed [16, 48, 95]. A reaction-diffusion space of *bubbles* was further developed in [101], where instead of a single contour, multiple bubbles were simultaneously placed and grown from homogeneous regions of the image. The latter is perhaps the closest to the approach proposed here, however, we introduce a new skeleton-based stopping criteria to prevent seeds from merging due to our regularity and compactness requirements. Moreover, our seeds are initialized to avoid image edges (rather than using random initialization) and we employ a more advanced reaction-diffusion evolution strategy, making it possible to better capture object edges.

While there are many variations on the theme of dilating seeds using geometric flows (e.g., this idea has been used for segmenting vasculature in medical imaging [56]), none of these methods have been applied thus far to superpixel segmentation. Below we develop such a technique by combining a curve evolution model for dilation with a skeletonization process on the background region to prevent the expanding seeds from merging. We demonstrate that this technique advances the state of the art in compact superpixel computation by 1) being applicable to megapixel size images, with very high superpixel densities, and 2) providing comparable accuracy to N-Cuts, but with significantly lower run times.

3.2 Superpixels from Geometric Flows

The key idea in our approach is to reduce superpixel computation to an efficiently-solvable geometric flow problem. Our approach is guided by five basic principles:

- **Uniform size and coverage:** Superpixel segmentation should partition an image into regions that are approximately uniform in size and shape (compactness), minimizing region under-segmentation, provided that superpixel size is comparable to the size of the smallest target region. We achieve this by designing a geometric flow that dilates an initial set of uniformly-distributed seeds, where each seed corresponds to one superpixel. The seeds behave initially like reaction-diffusion bubbles [101].
- **Connectivity:** Each superpixel should represent a simply-connected set of pixels. Our

dilation-based flow combined with its level-set implementation, ensures that this constraint is always satisfied.

- **Compactness:** In the absence of local edge information, superpixels should remain compact. Our flow begins from circular seeds and assumes no prior bias on the location of superpixel boundaries. To maximize compactness, we include a term that produces constant motion in the direction of the outward normal in regions of uniform intensity. This term maximizes the rate of area growth, while retaining the minimum possible isoperimetric ratio, which is 4π for a circular region.
- **Smooth, edge-preserving flow:** When growth stops, superpixel boundaries should coincide with image edges. This requires a geometric flow formulation with three properties: (1) it should slow down boundary growth in the vicinity of edges; (2) it should be attracted to edges; and (3) it should produce smooth boundaries. To do this, we borrow ideas from work on geometric active contours [15, 16, 48, 49, 62]. Such formulations provide an easy way to incorporate image-based controls on boundary growth, and include both a "doublet" term for attraction and a curvature term for shape regularization.
- **No superpixel overlap:** A superpixel segmentation should assign every pixel to a single superpixel. Therefore, boundary evolution should stop when two distinct dilating seeds are about to collide. To achieve this we incorporate a simple skeleton-based mechanism for collision detection in the background.

These considerations lead to a geometric flow-based algorithm, that we call *TurboPixels*, whose goal is to maintain and evolve the boundary between the *assigned region*, which contains all pixels that are already inside some superpixel, and the *unassigned region*, which contains all other pixels. At a conceptual level, the algorithm consists of the following steps, as illustrated in Fig. 3.2:

1. place initial seeds

2. iterate over the following basic steps until no further evolution is possible, i.e., when the

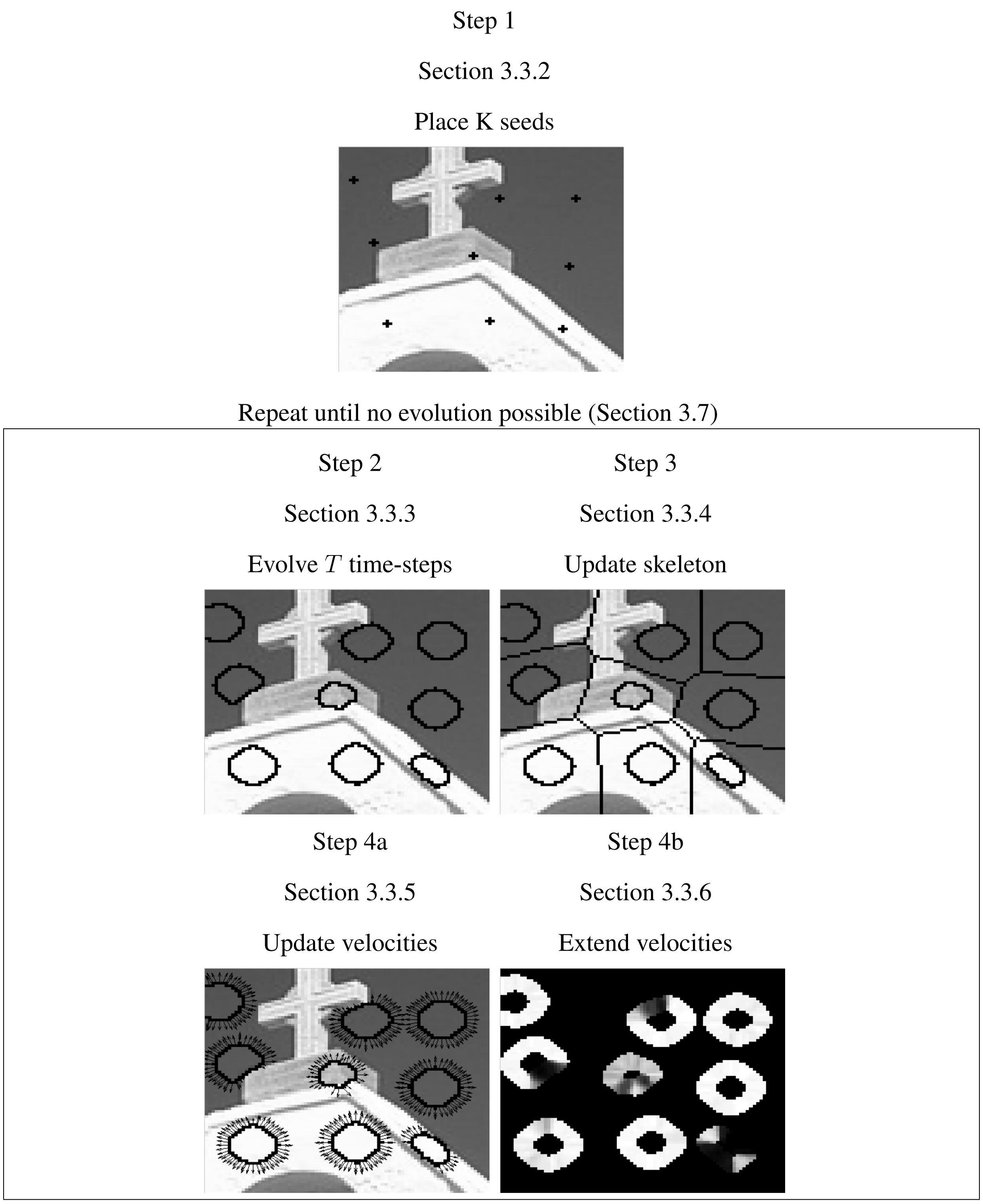


Figure 3.2: Steps of the TurboPixel algorithm. In Step 4a the vectors depict the current velocities at seed boundaries. Where edges have been reached the velocities are small. In Step 4b the magnitude of velocities within the narrow band is proportional to brightness.

speed at all boundary pixels is close to zero.

(a) evolve this boundary for T time steps;

(b) estimate the skeleton of the unassigned region;

(c) update the speed of each pixel on the boundary and of unassigned pixels in the boundary's immediate vicinity.

See Algorithm 1 for a pseudocode summary of these steps, each of which is discussed in detail below.

3.3 The TurboPixel Algorithm

3.3.1 Level-Set Boundary Representation

Geometric flows of the type associated with the TurboPixel algorithm are commonly implemented using level-set methods [73]. The basic idea is to devise a flow by which curves evolve to obtain superpixel boundaries. Let $\mathbf{C}$ be a vector of curve coordinates parameterized by p, a parameter which runs along the curve, and t, a parameter to denote evolution in time. Let $\mathbf{N}$ represent its outward normal and let each point move with speed S by a curve evolution equation $\frac{\partial \mathbf{C}}{\partial t} = S\mathbf{N}$. This curve evolution equation is implemented by first embedding $\mathbf{C}$ as a level set of a smooth and continuous function $\Psi : \mathbf{R}^2 \times [0, \tau) \to \mathbf{R}$ and then evolving this embedding function according to:

$$\Psi_t = -S\,||\nabla\Psi||\,. \tag{3.1}$$

In practice we define Ψ over the image plane as the signed Euclidean distance of each image pixel to the closest point on the boundary between assigned and unassigned (background) regions. A pixel's distance is positive if it is in the unassigned region and negative if it is not, with the boundary represented implicitly as the zero level set of Ψ. Since we are only interested in the zero level set of Ψ, we maintain an accurate representation of Ψ only in a narrow band around its current zero level set (typically 4 pixels wide on each side of the boundary). This narrow band is computed using the Fast Marching implementation in LMSLIB[1]. The superpixel boundaries can be computed at sub-pixel resolution by interpolation.

3.3.2 Initial Seed Placement

One of our key objectives is to compute superpixels that are evenly distributed over the image plane. Given a user-specified value K of superpixels, we place K circular seeds in a lattice formation so that distances between lattice neighbors are all approximately equal to $\sqrt{\frac{N}{K}}$, where N is the total number of pixels in the image. This distance completely determines the seed lattice, since it can be readily converted into a distance across lattice rows and columns. In our implementation the initial seed radius is 1 pixel.

The above strategy ensures that superpixels in a uniform-intensity image will satisfy the uniform distribution objective exactly. In practice, of course, images are not uniform and this deterministic placement may cause some seeds to accidentally fall on or close to a strong edge, inhibiting their early growth. To avoid this we perturb the position of each seed by moving it in the direction of the image gradient as a function of the gradient magnitude (see Sec. 3.3.5), with the maximum perturbation determined by the seed density.

3.3.3 Numerical Level Set Evolution

We use the following first-order discretization in time of Eq. (3.1):

$$\Psi^{n+1} = \Psi^n - \ S_I S_B \parallel \nabla \Psi^n \parallel \Delta t \ \ . \tag{3.2}$$

Each application of Eq. (3.2) corresponds to one "time step" Δt in the evolution of the boundary. We apply this equation until any point on the evolving boundary reaches the edge of the narrow band. The key term controlling the evolution is the product of two speeds $S_I S_B$,

[1]$LSMLIB$ is a library of level set routines written by K. Chu (`http://www.princeton.edu/~ktchu/software/lsmlib/`).

which lie at the heart of our TurboPixel algorithm. The first term (S_I) depends on local image structure and superpixel geometry at each boundary point, and the second (S_B) depends on the boundary point's proximity to other superpixels. We detail the computation of these velocities in Sections 3.3.5 and 3.3.4, respectively.

In theory, the velocities in Eq. (3.2) are defined at every point on the zero level set. In practice, we compute this term for a small band of pixels in the vicinity of the zero level set at iteration n. We discuss this process in Sec. 3.3.6. For notational simplicity, we omit the parameter n from Ψ^n in the following sections, except where it is explicitly needed.

3.3.4 Proximity-Based Boundary Velocity

The proximity-based velocity term ensures that the boundaries of nearby superpixels never cross each other. To do this, we use a binary stopping term that is equal to 0 on the 2D homotopic skeleton of the unassigned region and is is equal to 1 everywhere else, i.e., $S_B(x, y) = 0$ if and only if (x, y) is on the skeleton. This formulation allows the boundary of each superpixel to be guided entirely by the underlying image, until it gets very close to another superpixel boundary.

Since the regions between evolving curves change at each iteration of our algorithm, the skeleton must be updated as well. We do this efficiently by marking all pixels in these unassigned regions (i.e., those with $\Psi(x, y) > 0$) and then applying a homotopy preserving thinning algorithm [96] on them to compute the skeleton. The thinning algorithm removes pixels ordered by their distance to the boundary of the region with the constraint that all digital points that can be removed without altering the topology are removed.

3.3.5 Image-Based Boundary Velocity

Our image-based speed term combines the reaction-diffusion based shape segmentation model of [15, 62, 101] with an additional "doublet" term provided by the geodesic active contour

Algorithm 1: TurboPixel Algorithm

```
   Input: Image $I$, number of seeds $K$
   Output: Superpixel boundaries $B$
1  Place $K$ seeds on a rectangular grid in image $I$;
2  Perturb the seed positions away from high gradient regions;
3  Set all seed pixels to "assigned";
4  Set $\Psi^0$ to be the signed Euclidean distance from the "assigned" regions;
5  assigned_pixels $\leftarrow \sum_{x,y} [\Psi^0(x,y) >= 0]$;
6  Compute the pixel affinity $\phi(x,y)$;
7  $n \leftarrow 0$;
8  while Change in assigned_pixels is large do
9      Compute the image velocity $S_I$;
10     Compute the boundary velocity $S_B$;
11     $S \leftarrow S_I S_B$;
12     Extend the speed $S$ in a narrow band near the zero level-set of $\Psi^n$;
13     Compute $\Psi^{n+1}$ by evolving $\Psi^n$ within the narrow band;
14     $n \leftarrow n + 1$;
15     assigned_pixels $\leftarrow \sum_{x,y} [\Psi^n(x,y) >= 0]$;
16 $B \leftarrow$ homotopic skeleton of $\Psi^n$;
17 return $B$
```

[16, 48] to attract the flow to edges:

$$S_I(x,y) = \underbrace{[1 - \alpha\,\kappa(x,y)]\,\phi(x,y)}_{\text{reaction-diffusion term}} - \underbrace{\beta\,[\mathbf{N}(x,y) \cdot \nabla\phi(x,y)]}_{\text{“doublet” term}} \;. \tag{3.3}$$

The reaction-diffusion term ensures that the boundary's evolution slows down when it gets close to a high-gradient region in the image. It is controlled by three quantities: (1) a "local affinity" function $\phi(x,y)$, computed for every pixel on the image plane, that is low near edges and high elsewhere; (2) a curvature function $\kappa(x,y)$ that expresses the curvature of the boundary at point (x, y) and smoothes the evolving boundary; and (3) a "balancing" parameter α that weighs the contribution of the curvature term.

Intuitively, the doublet term ensures that the boundary is attracted to image edges, i.e., pixels where the affinity is low. Specifically, when a point (x, y) on the boundary evolves toward a region of decreasing affinity (an image edge), its normal $\mathbf{N}(x, y)$ will coincide with the negative gradient direction of ϕ, and the term acts as an attractive force. If the boundary crosses over an edge these two vectors will point in the same direction and cause a reversal in the boundary's direction of motion.

Local affinity function Our algorithm does not depend on a specific definition of the function ϕ, as long as it is low on edges and is high elsewhere. For almost all the experiments in this chapter, we used a simple affinity measure based on the grayscale intensity gradient:

$$\phi(x, y) = e^{-E(x,y)/\nu}, \ E(x, y) = \frac{\| \nabla I \|}{G_\sigma * \| \nabla I \| + \gamma} \ . \tag{3.4}$$

Our affinity function ϕ produces high velocities in areas with low gradients, with an upper bound of 1. Dividing the gradient magnitude in $E(x, y)$ by a local weighted sum of gradient magnitudes provides a simple form of contrast normalization. The support width of the normalization, controlled by σ, is proportional to the expected initial distance between seeds. This normalization allows weak but isolated edges to have a significant effect on speed, while suppressing edge strength in dense texture. The constant γ ensures that the effect of insignificant signal gradients remains small. We note that whereas our $E(x, y)$ is a simple measure of grayscale image gradient, the implementation of N-Cuts we use for comparison ([111]) in our experiments in Section 3.4 employs a more complex measure of intervening contours computed using a texture-based edge map.

Normal and curvature functions The outward normal of the zero level set of Ψ at a point (x, y) is given by the derivatives of Ψ, i.e., $\mathbf{N} = \nabla\Psi / \| \nabla\Psi \|$. The curvature of the zero level set, at a point (x, y), is given by [73]:

$$\kappa = \frac{\Psi_{xx}\Psi_y^2 - 2\Psi_x\Psi_y\Psi_{xy} + \Psi_{yy}\Psi_x^2}{(\Psi_x^2 + \Psi_y^2)^{\frac{3}{2}}} . \tag{3.5}$$

As is standard for diffusive terms, the derivatives of Ψ used for κ are computed using central difference approximations. Central difference approximations are also used for all other calculations with the exception of $\parallel \nabla\Psi^n \parallel$ in the level set form for the reaction term ($\phi(x, y)$) in Eq. 3.2, for which upwind derivatives [73] must be used since it is a hyperbolic term.

Balancing parameters The balancing parameters α and β in Eq. 3.3 control the relative contributions of the reaction-diffusion and doublet terms. Higher values of α prevent "leakage" through narrow edge gaps, but also prevent sharp superpixel boundaries that may be sometimes desirable. High values of β cause better stopping behavior of seeds on weak edges, but also slow down the evolution of seeds elsewhere.[2]

3.3.6 Speed Extension

The velocity terms S_I and S_B have meaning only on the current superpixel boundaries, i.e., the zero level set of Ψ. This leads to two technical difficulties. First, the zero level set is defined implicitly and, hence, it lies "in between" the discrete image pixels. Second, each time we invoke a level set update iteration (Eq. 3.2), the boundary must move by a finite amount (i.e., at least a sizeable fraction of a pixel).

Speed extension gives a way to solve both problems and is common in existing curve evolution implementations [62]. Here, we extend ϕ and $\nabla\phi$, the only image-dependent terms, in the same narrow band we use to maintain an accurate estimate of Ψ (see Sec. 3.3.3). To each pixel (x, y) in this narrow band, we simply assign the ϕ and $\nabla\phi$ values of its closest pixel on

[2]Based on empirical observation, the values $\alpha = 0.3$ and $\beta = 1$ were chosen. These values limit the amount of leakage during seed evolution, without slowing down the evolution in other regions. In the future, we intend to learn the optimal values for these parameters automatically by evaluating the performance of the algorithm on a training set of images.

the boundary[3].

3.3.7 Termination Conditions & Final Segmentation

The algorithm terminates when the boundaries stop evolving. Since in theory the boundaries can evolve indefinitely with ever-decreasing velocities, the algorithm terminates when the relative increase of the total area covered by superpixels falls below a threshold. We used a relative area threshold of 10^{-4} in all our experiments.

After termination, the evolution results are post-processed so that the superpixel boundaries are exactly one pixel in width. This is done in three steps. First, any remaining large unassigned connected regions are treated as superpixels. Next, very small superpixels are removed, making their corresponding pixels unassigned. Finally, these unassigned regions are thinned, as in Sec. 3.3.4, according to the algorithm in [96]. The thinning is ordered by a combination of Euclidean distance to the boundary and a ϕ-based term, in order to obtain smooth superpixel contours that are close to edges.

3.3.8 Algorithm Complexity

The complexity of our algorithm is roughly linear in the total number of image pixels N for a fixed superpixel density. At each time step, all elements of the distance function Ψ are updated (see Eq. 3.2). Each update requires the computation of the partial derivatives of Ψ and evaluation of $S_I S_B$. Thus each update takes $\mathcal{O}(N)$ operations.

The speed extension and homotopic skeleton computations are not linear in image size. Both actions are $\mathcal{O}(N \log N)$ but can be made faster in practice. If b is the number of pixels in the narrow band (which is linear in the number of pixels that lie on zero-crossings), then the complexity of speed extension is $\mathcal{O}(N) + \mathcal{O}(b \log b)$. The term $\mathcal{O}(b \log b)$ arises from the computation of the distance function Ψ in the narrow band. While b can approach N in theory,

[3]The algorithm that efficiently computes Ψ for all pixels within the narrow band provides their closest boundary pixel as a byproduct, so no extra computations are necessary.

it is usually much smaller in practice.

The homotopic skeleton computation is $\mathcal{O}(N) + \mathcal{O}(k \log k)$ [96], where k is the number of unassigned pixels. Again, $\mathcal{O}(k \log k)$ is the complexity of distance function computation inside the unassigned region. In practice, $k \ll N$, especially toward the end of the evolution when few unassigned pixels remain.

It now remains to take into account the number of iterations of the algorithm. Under ideal conditions, all the curves evolve with maximal speed until they meet or reach an edge. Since the expected distance between seeds is D_n initially (see Sec. 3.3.2), it will take $\mathcal{O}(\sqrt{\frac{N}{K}})$ iterations for the algorithm to converge. Hence, the algorithm converges more slowly for larger images, and more quickly as the superpixel density increases. Thus for a fixed superpixel density (keeping D_n constant), the number of iterations will be constant, making the overall complexity roughly $\mathcal{O}(N)$.

3.4 Experimental Results

We evaluate the performance of the TurboPixel algorithm (both with gradient-based and Pb-based affinity) by comparing its accuracy and running time to four other algorithms: Normalized Cuts (Ncuts), Superpixel Lattice (Lattice)[4] and square blocks (Sb), all of which encode a compactness constraint, and Felzenszwalb and Huttenlocher (Felz), which does not. The TurboPixel algorithm was implemented in Matlab with several C extensions[5]. For Ncuts, we use the 2004 Ncut implementation based on [111][6], while for Sb, we simply divide the image into even rectangular blocks, providing a naive but efficient benchmark for accuracy (other algorithms are expected to do better). All experiments were performed on a quad-core Xeon 3.6

[4]The Superpixel Lattice algorithm of Moore et al. [68] was developed in parallel to this work and thus was not thoroughly evaluated against in this study.

[5]A beta-version of our code is available at `http://www.cs.toronto.edu/~babalex/turbopixels_supplementary.tar.gz`; the default parameter values are the same as those used for the experiments in this article.

[6]We use Version 7 from Jianbo Shi's website `http://www.cis.upenn.edu/~jshi/software/files/NcutImage_7_1.zip`

Ghz computer. We use the Berkeley database, which contains 300 (481×321 or 321×481) images. In our experiments, the image size is defined as the fraction of the area of the full image size of 154401 pixels. In all experiments, performance/accuracy is averaged over at least 25 images and in most cases over a larger number.[7] Finally, the gradient-based affinity function of a grayscale image (Eq. 3.4) was used for the TurboPixel algorithm, a Pb affinity based on brightness and texture was used for TurboPixel Pb as well as for Lattice, a difference in image intensity was used as affinity in Felz, and a more elaborate (intervening contours) affinity was used for Ncuts.

3.4.1 Under-segmentation Error

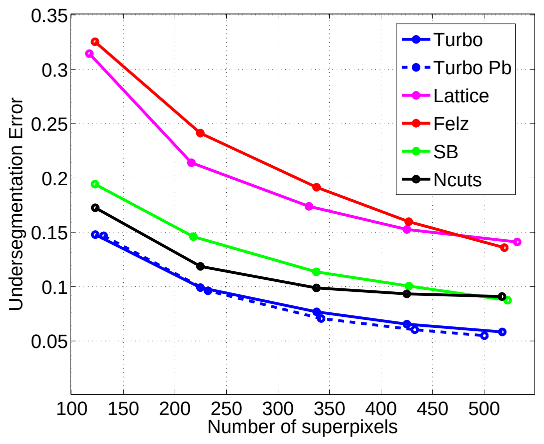
As stated in Section 3.1, algorithms that do not enforce a compactness constraint risk a greater degree of under-segmentation. Given a ground-truth segmentation into segments $g_1, \ldots, g_K$ and a superpixel segmentation into superpixels $s_1, \ldots, s_L$, we quantify the under-segmentation error for segment g_i with the fraction

$$\frac{\left[\sum_{\{s_j \mid s_j \cap g_i \neq \emptyset\}} Area(s_j)\right] - Area(g_i)}{Area(g_i)}. \tag{3.6}$$

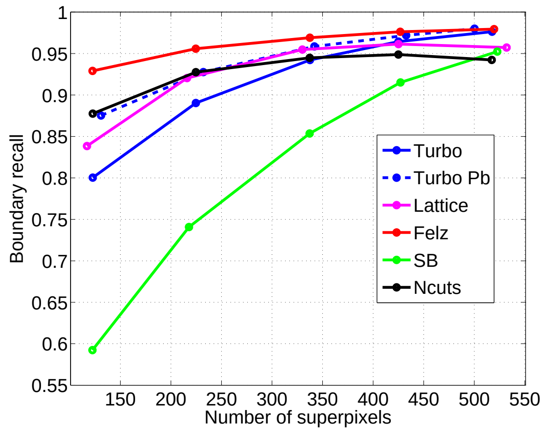
Intuitively, this fraction measures the total amount of "bleeding" caused by superpixels that overlap a given ground-truth segment, normalized by the segment's area.

To evaluate the under-segmentation performance of a given algorithm, we simply average the above fraction across all ground-truth segments and all images. Figure 3.3(a) compares the algorithms using this metric, with under-segmentation error plotted as a function of superpixel density. The inability of Felz to stem the bleeding is reflected in the significantly higher under-segmentation error over all three algorithms that encode a compactness constraint. Note that Lattice also suffers from bleeding, but it is possible to change the parameters of this method to force stronger conformity with a grid structure, thereby improving the undersegmentation

[7]Due to the long running time and large memory requirements of Ncuts, using the entire database was prohibitively expensive.



(a)



(b)

Figure 3.3: Under-segmentation error (a) and accuracy (boundary recall) (b) as a function of superpixel density.

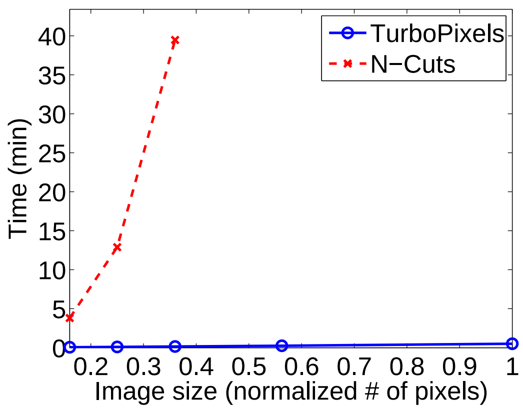
error at the expense of boundary recall. Of these methods, the TurboPixel and TurboPixel Pb algorithms achieve the least under-segmentation error, where the improved performance of the latter is due to the more powerful affinity function.

3.4.2 Boundary Recall

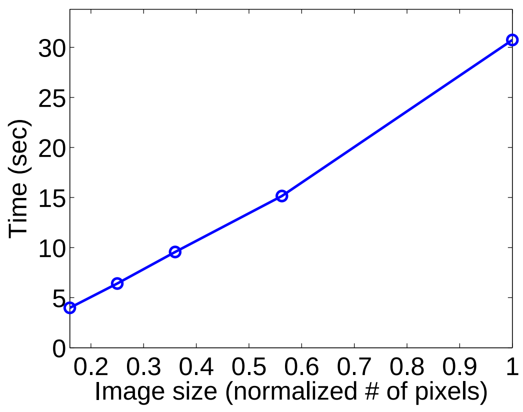
Since precise boundary shape might be necessary for some applications, we adopt a standard measure of boundary recall (what fraction of the ground truth edges fall within a small distance threshold (2 pixels in this experiment) from at least 1 superpixel boundary. As shown in Fig. 3.3(b), Felz offers better recall at lower superpixel densities, while at higher superpixel densities, Felz and TurboPixel are comparable, with both outperforming Ncuts and Sb. The fact that Felz does not constrain its superpixels to be compact means that it can better capture the boundaries of thin, non-compact regions at lower superpixel densities.

3.4.3 Timing Evaluation

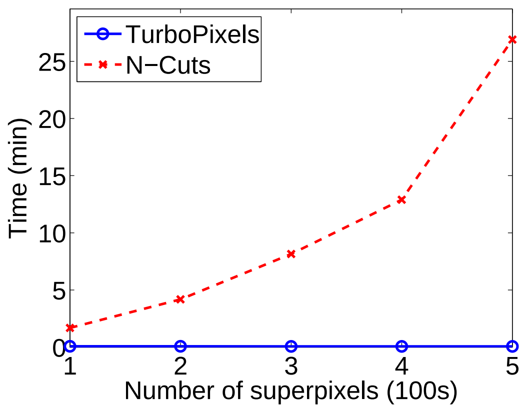
With the exception of the naive and clearly inferior Sb algorithm, the cost of enforcing a compactness constraint (Ncuts, TurboPixel) is significant; for example, Felz is, on average, 10



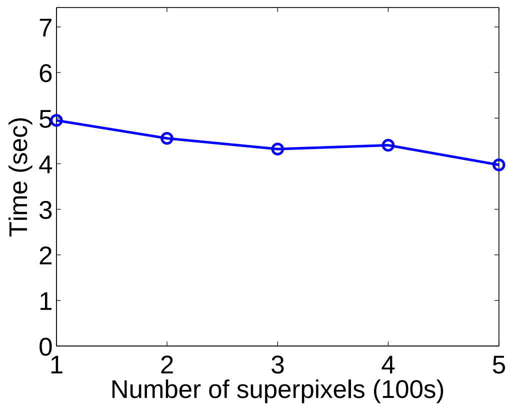
(a)



(b)



(c)



(d)

Figure 3.4: Timing evaluation. (a) Running time vs. image size. (b) An expanded version (a) to show the behavior of the TurboPixel algorithm. (c) Running time vs. superpixel density. (d) An expanded version of (c) showing the behavior of the TurboPixel algorithm.

times faster than TurboPixel. Lattice is even faster than Felz when affinity function computation is excluded. For our timing analysis, we restrict our comparison to TurboPixel and Ncuts, the two primary competitors in the class of algorithms with a compactness constraint. For any superpixel algorithm, it is appropriate to increase the number of superpixels as the image size increases, so that the expected area (in pixels) of each superpixel remains constant. Fig. 3.4 (a and b) shows the running time of the two algorithms as a function of increased image size. The expected size of a superpixel is kept fixed at about 10$\times$10 pixels.

The TurboPixel algorithm is several orders of magnitude faster. It is almost linear in image size compared to Ncuts, whose running time increases non-linearly. Due to “out of memory” errors, we were unable to run Ncuts for all of the parameter settings used for the TurboPixel results. Fig. 3.4 (c and d) show running time as a function of superpixel density, with the image size fixed at 240×160 (one quarter of the original size). The running time of Ncuts increases in a non-linear fashion whereas the running time of the TurboPixel algorithm decreases as the density of the superpixels increases. This is due to the fact that the seeds evolve over a smaller spatial extent on average and thus converge faster.

3.4.4 Qualitative Results

Fig. 3.5 gives a qualitative feel for the superpixels obtained by the TurboPixel algorithm for a variety of images from the Berkeley database. Observe that the superpixel boundaries respect the salient edges in each image, while remaining compact and uniform in size.[8] Obtaining superpixels under such conditions using Ncuts is prohibitively expensive. Fig. 3.6 provides a qualitative comparison against the results obtained using Ncuts. The TurboPixel algorithm obtains superpixels that are more regularly shaped and uniform in size than those of Ncuts.

The TurboPixel algorithm is of course not restricted to work with affinity functions that are based strictly on image gradient, as discussed in Section 3.3.5, and hence more refined measures can be used for superpixel boundary velocity. Fig. 3.7 shows the performance of the algorithm when the boundary velocity incorporates the Pb edge detector [63]. Note how the edge between the leopard and the background is captured much better when a Pb-based affinity is used. Moreover, the shapes of the superpixels inside the leopard are more regular for the latter case.

[8]Supplementary material (`http://www.cs.toronto.edu/~babalex/turbopixels_supplementary.tar.gz`) contains additional results of the TurboPixel algorithm on megapixel sized images with superpixel densities in the thousands.

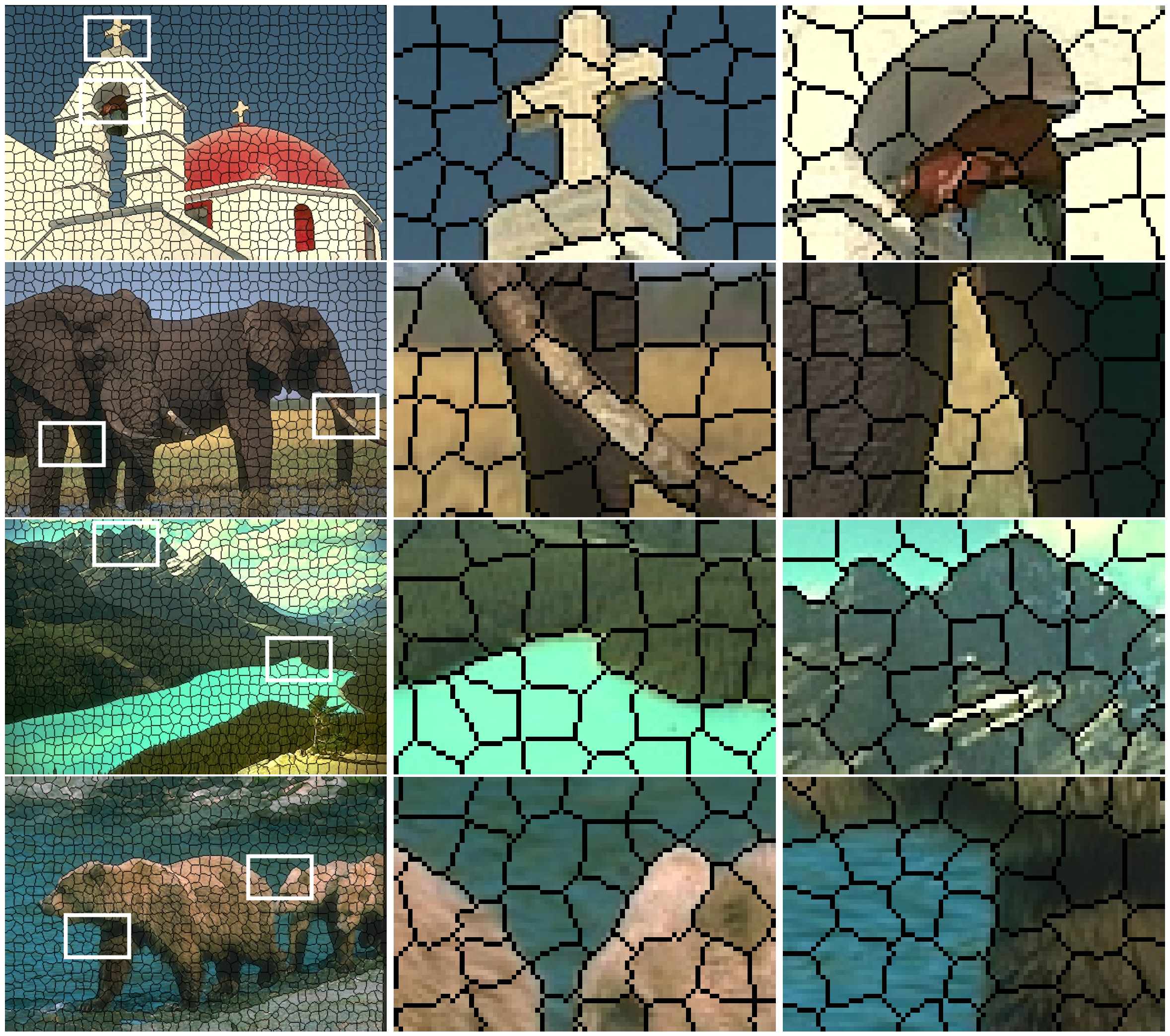


Figure 3.5: TurboPixel results on a variety of images from the Berkeley database, with a zoom-in on selected regions in the middle and right columns.

3.5 Limitations and Future Work

Our current system suffers from a number of limitations. First, note that in case of textured images, our current Pb-based affinity enables us to obtain better superpixels, but is just a first stab at better handling texture. We see that the leopard's top boundary in Figure 3.7, for example, is not being captured as well as the object boundaries we have shown before, where our original, gradient-based affinity was used. Handling texture is not the only issue faced by a superpixel algorithm. Edges in an image can also be present at multiple scales. In Figure 3.8,

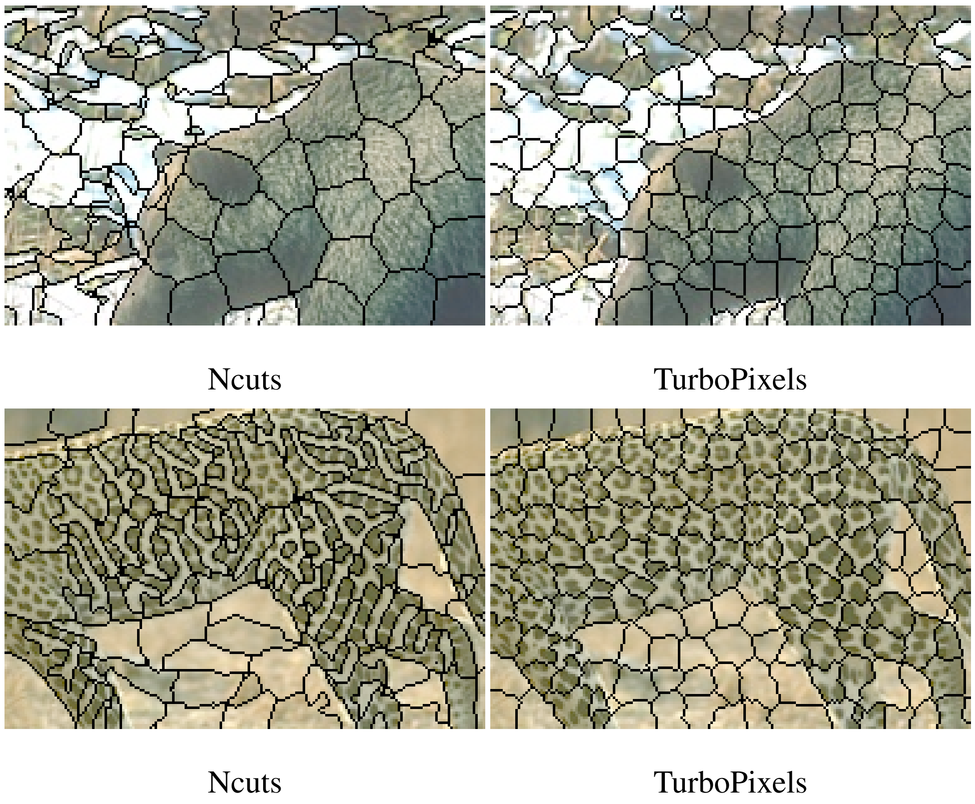


Figure 3.6: A qualitative comparison of TurboPixel results with gray-level gradient-based affinity compared to results with Ncuts.

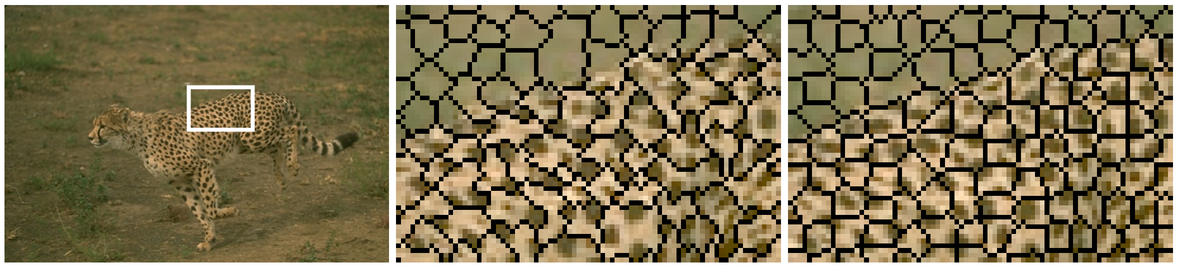


Figure 3.7: Qualitative results of the TurboPixel algorithm using gradient-based (middle) and Pb-based (right) affinity functions.

unlike the sharp boundaries of the deer, the edge separating the field from the trees is blurry as it is out of focus. Running our algorithm on the original resolution with our gradient-based affinity results in bleeding (Figure 3.8 middle). However, subsampling the image and detecting edges at a coarse scale (lowering the superpixel density accordingly), results in a significant

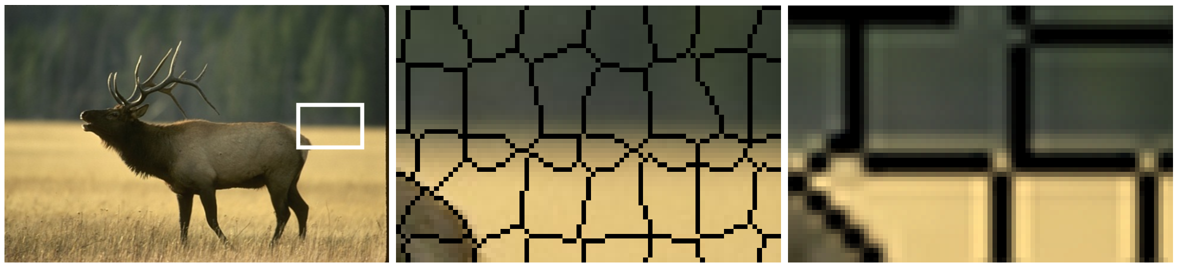


Figure 3.8: Effect of multiple edge scales on performance. Running TurboPixels on the original image (middle) fails to capture strong, but blurry edges. Running on a subsampled image (right) results in better recall of blurry edges.

improvement (Figure 3.8 right). To summarize the pixel affinity issue, our system provides a black-box for fast superpixel recovery and the quality of the results directly depends on the provided pixel affinity. While we have illustrated that the system works reasonably well with two example affinities, there is room for significant improvement in that regard, which we leave for future work.

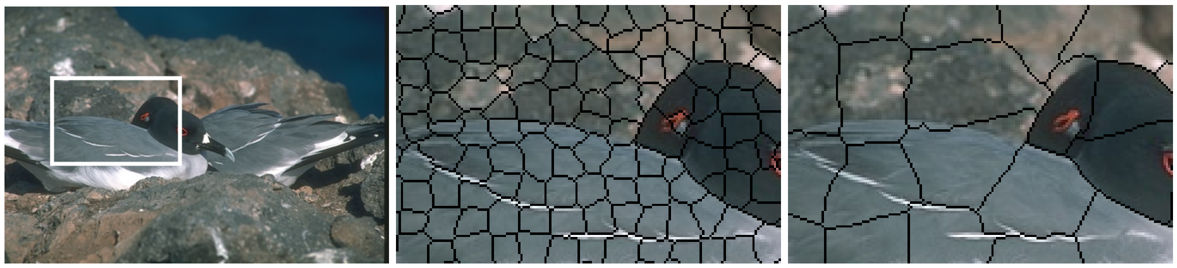


Figure 3.9: Running TurboPixels at different superpixel resolutions. High density superpixel resolution (middle), in addition with compactness constraints, is often enough to prevent bleeding. Running TurboPixels at coarse superpixel resolution (right) often causes undersegmentation.

The strongest points of our approach, i.e., the use of a local segmentation technique and enforcing superpixel compactness, are also its main downfalls. Like any local segmentation technique, our method can also suffer from bleeding effects in regions of low contrast. Enforcing compactness and maintaining high superpixel density helps to minimize bleeding, but is far from being a complete solution for undersegmentation. Given the same affinities, a global approach, such as Ncuts, is expected to do better, especially at coarser superpixel resolutions. The effects of weak contrast were apparent in previous examples. Notice the bleeding on the bottom part of the cross and the elephant tusk in Figure 3.5. The situations worsen at coarser superpixel resolutions. Figure 3.9 illustrates that, while the top boundary of the bird is captured well at a fine resolution (middle), our greedy approach may not be suitable for coarser superpixel resolutions (right). Finally, while compactness helps to avoid bleeding, especially when using a local segmentation approach, it may negatively affect performance when thin, non-compact shapes are present (such as the stick in Figure 3.10). Our regular seeding not only makes it challenging to place a sufficient number of superpixel seeds inside thin structures, but even seeded correctly such seeds are likely to bleed through object boundaries. In future work, we will explore other seeding procedures, placing more seeds in regions of high importance or inside thin structures, while using coarser resolutions inside homogeneous regions like the sky. Moreover, our framework could potentially be extended to encode additional shape constraints besides compactness. For instance, we could enforce superpixels to be elliptical inside narrow structures and thereby further minimize bleeding in such cases.

3.6 Conclusions

The task of efficiently computing a highly regular over-segmentation of an image can be effectively formulated as a set of locally interacting region growing problems, and as such avoids the high cost of computing globally optimal over-segmentations (or their approximations), such as N-Cuts. Combining the power of a data-driven curve evolution process with a set of skeletal-based external constraints represents a novel, highly efficient framework for superpixel segmentation. The results clearly indicate that while superpixel quality is comparable to the benchmark algorithm, our algorithm is several orders of magnitude faster, allowing it to be applied to large megapixel images with very large superpixel densities.

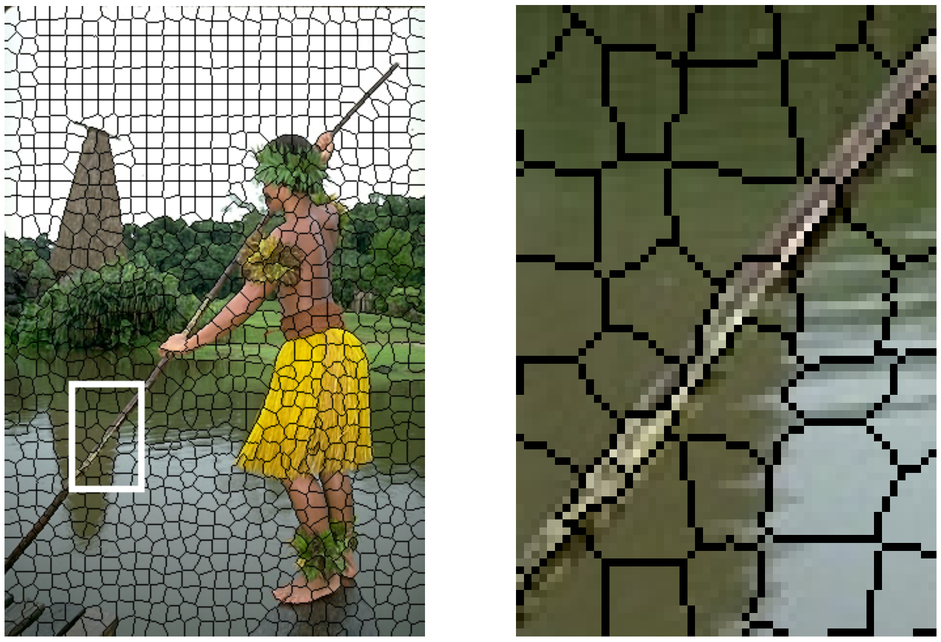


Figure 3.10: Negative effect of superpixel compactness on capturing thin objects.

The framework is general and, like any region segmentation algorithm, is based on a user-defined measure of affinity between pixels. While our experiments have demonstrated the use of intensity gradient-based and Pb-based affinities, other more complex affinity measures, perhaps incorporating information from multiple scales, are possible. Selecting the appropriate affinity measure is entirely task dependent. We offer no prescription, but rather offer a general framework into which a domain-dependent affinity measure can be incorporated.

It is also important to note that we have intentionally skirted several important domain-dependent problems. One global issue is the fact that our framework allows the user to control the superpixel shape and density. On the issue of density, our approach is very generic, and one could imagine that with domain knowledge, seeds could be placed much more judiciously. And depending on the task, seeds could be placed with varying density at the cost of lower superpixel uniformity. In some domains, varying seed density may be more desirable. In textured images, for example, seeds could be placed to capture the individual texture elements better (like the spots of the leopard in Figure 3.6). Moreover, our framework allows us to guide superpixels to have a certain shape. Currently, in the absence of edges, the superpixels would grow in a circular manner. However, one could imagine growing superpixels to be elliptical instead. Both of the above extensions should prove useful in cases of narrow structures (Figure 3.10). Still, as shown in the experiments, the use of a compactness constraint clearly minimizes under-segmentation at a significantly higher computational cost. If both under-segmentation and irregularly shaped superpixel boundaries can be tolerated, the Felzenszwalb algorithm is clearly the better choice, offering a tenfold speed-up as well as improved boundary recall at lower superpixel densities.

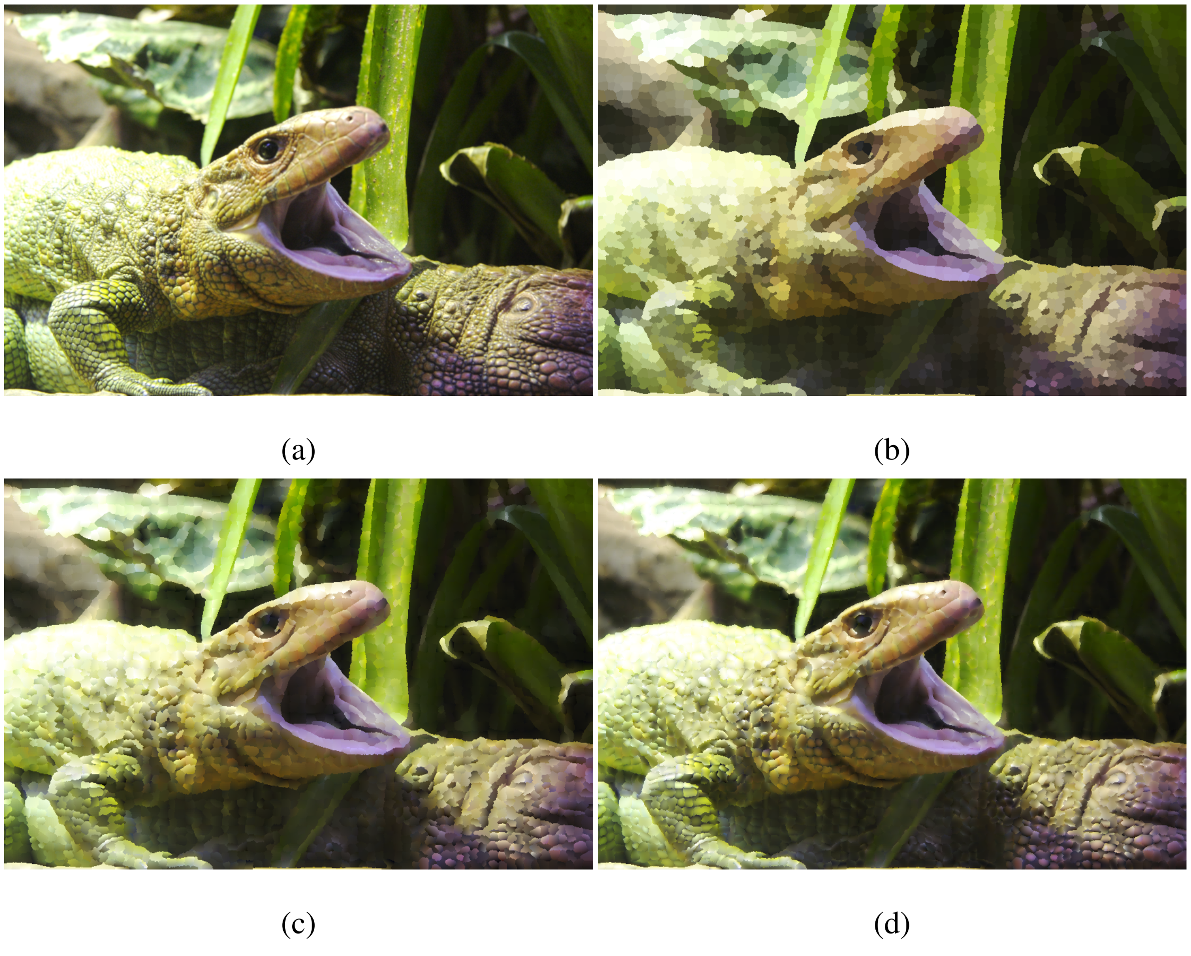


Figure 3.11: Image representation using superpixels. Each superpixel from the original image (a) is colored with: (b) The average color of the original pixels in it. (c) The best linear fit to the color of the original pixels in it. (d) The best quadratic fit to the color of the original pixels in it.

Perhaps the most important issue is what to do with the resulting superpixels. Currently, superpixels are mainly used for image labeling problems to avoid the complexity of having to label many more pixels. In the same manner, superpixels can be used as the basis for image segmentation. In the graph cuts segmentation algorithm, the affinity can be defined over superpixels instead of over pixels, resulting in a much smaller graph. Superpixels can also be considered as a compact image representation. To illustrate this idea, in Figure 3.11 each superpixel's color is approximated by three polynomials (one per channel). Note that whereas the mean and the linear approximations seem poor, the quadratic approximation approaches the quality of the original image. In the next two chapters, we use compact superpixels as a basis for perceptual grouping. In Chapter 4, we group compact superpixels to separate figure from ground. In Chapter 5, we use them to approximate the maximal discs of a skeletal branch and group them to form symmetric parts.

Chapter 4

Optimal Contour Closure by Superpixel Grouping

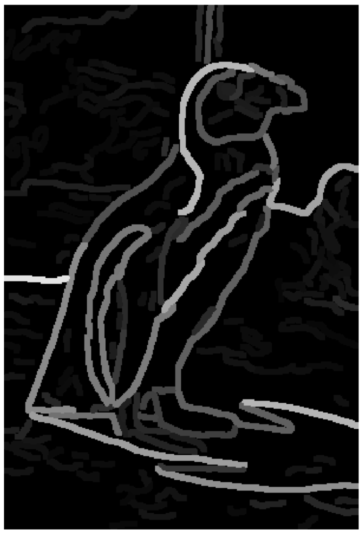
4.1 Introduction

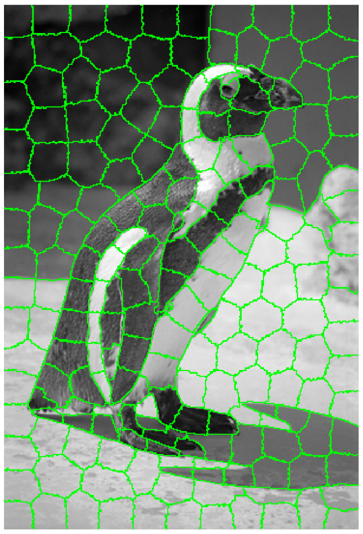
One of the key challenges in perceptual grouping is computing contour closure, i.e., linking together a set of fragmented contours into a cycle that separates an object from its background. What makes the problem particularly hard is the intractable number of cycles that may exist in the contours extracted from an image of a real scene. Early perceptual grouping researchers [105] identified a set of nonaccidental contour relations, such as symmetry, parallelism, collinearity, co-curvilinearity, etc., that can be used to link together causally related contours. Such nonaccidental grouping rules can serve as powerful heuristics to help manage the complexity of greedily searching for a contour closure that is unlikely to have arisen by chance [29, 30]. However, the space of possible closures is still overwhelming, particularly when one allows larger and larger boundary gaps in a closure. Finding an optimal solution is intractable without somehow reducing the complexity of the problem.

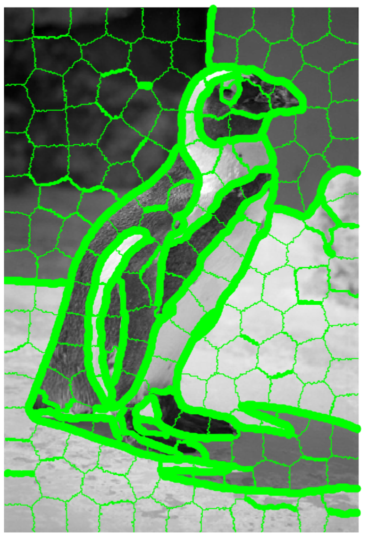
In this chapter, we introduce a novel framework for efficiently searching for an optimal closure. Figure 4.1 illustrates an overview of our approach. Given an image of extracted contours (Figure 4.1(a)), we begin by restricting contour closures to pass along boundaries of superpixels computed over the contour image (Figure 4.1(b)). In this way, our first contribution is to reformulate the problem of searching for cycles of contours as the problem of searching for a subset of superpixels whose border has strong contour support in the contour image; the assumption we make here is that those salient contours that define the boundary of the object (our target closure) will align well with superpixel boundaries. However, while a cycle of contours represents a single contour closure, our reformulation needs a mechanism to prefer superpixel subsets that are spatially coherent.

Spatial coherence is an inherent property of a cost function that computes the ratio of perimeter to area. We modify the ratio cost function of Stahl and Wang [99] to operate on superpixels rather than contours, and extend it to yield a cost function that: 1) promotes spatially coherent selections of superpixels; 2) favors larger closures over smaller closures; and 3) introduces a novel, learned gap function that accounts for how much agreement there is between the boundary of the selection and the contours in the image. The third property adds cost as the number and sizes of gaps between contours increase. Given a superpixel boundary fragment (e.g., a side of a superpixel) representing a hypothesized closure component, we assign a gap cost that's a function of the proximity of nearby image contours, their strength, their orientation, and their curvature (Figure 4.1(c)). It is in this third property that our superpixel reformulation plays a second important role – by providing an appropriate scope of contour over which our gap analysis can be conducted.

In our third and final contribution, the two components of our cost function, i.e., area and gap, are combined in a simple ratio that can be efficiently optimized using parametric maxflow [50] to yield the global optimum. The optimal solution yields the largest set of superpixels bounded by contours that have the least gaps (Figure 4.1(d)). Moreover, parametric maxflow will yield the top k solutions (see [14], for example). In an object recognition setting, generating a small set of such solutions can be thought of as generating a small set of promising shape hypotheses which, through an indexing process, could invoke candidate models that could be


(a)


(b)


(c)

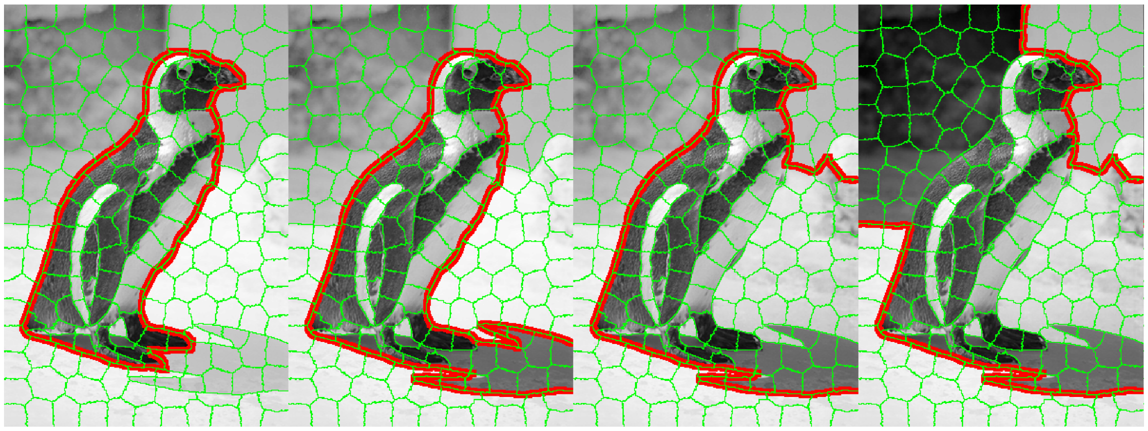

(d)

Figure 4.1: Overview of our approach: (a) contour image – while we take as input only this contour image, we will overlay the original image in the subsequent figures to ease visualization; (b) superpixel segmentation of contour image, in which superpixel resolution is chosen to ensure that target boundaries are reasonably well approximated by superpixel boundaries; (c) a novel, learned measure of gap reflects the extent to which the superpixel boundary is supported by evidence of a real image contour (line thickness corresponds to the amount of agreement between superpixel edges and image contours); (d) our cost function can be globally optimized to yield the largest set of superpixels bounded by contours that have the least gaps. In this case the solutions, in increasing cost (decreasing quality), are organized left to right.

verified (detected).

In the following sections, we begin by reviewing related work on contour closure (Section 4.2). Next, in Section 4.3, we introduce our problem formulation that transforms the problem of finding optimal cycles of contour fragments into the problem of finding an optimal subset of superpixels. It is here that our cost function is described. In Section 4.4, we describe our process for learning our gap function from training data, and in Section 4.5, we present an efficient procedure for finding the global minimum of our cost function using parametric maxflow. In Section 4.6, we evaluate our framework, comparing it to two competing approaches for computing closure, and discuss the strengths and weaknesses of our approach. Section 4.7 illustrates several extensions to the work. First, we show how our grouping framework can easily be augmented to include both contour and region information, thereby emphasizing the third important role of our superpixel reformulation of providing an appropriate scope over which appearance can be analyzed. Secondly, we show how the framework can benefit from the use of multiscale superpixel segmentation. Finally, we show how it can be naturally extended to finding closures in spatiotemporal domains. In Section 4.9, we draw conclusions and outline our plans for future work.

4.2 Related Work

Detecting closed contours in an image has been addressed by many researchers in different ways. One possible taxonomy for categorizing related work is based on the nature of the prior information used to constrain the grouping process. We will stop short of reviewing methods which assume object-level priors, for it is unclear how to make such methods scale up to very large databases. Instead, we focus on methods that make no assumptions about scene content, although as we will see, many make assumptions about the nature of parts that make up the objects in the scene. In fact, some methods incorporate low-, mid-, and high level shape priors, as exemplified by Ren et al. [82]. We will also stop short of reviewing methods focused solely on contour completion, e.g., Ren et al. [81] and Williams and Jacobs [106], although the regularities exploited by such approaches can clearly play a powerful role in detecting closure.

Many researchers have exploited the classical Gestalt cues of parallelism and symmetry to group contours. Lowe's [57] early work on perceptual grouping was one of the first to develop a computational model for parallelism, collinearity, and proximity. Many computational models exist for symmetry-based grouping, including Brady and Asada [10], Cham and Cipolla [17], Saint-Marc et al. [84], Ylä-Jääski and Ade [110], and more recently, Stahl and Wang [98]. One significant challenge faced by these systems is the complexity of pairwise contour grouping to detect symmetry-related contour pairs. The work in Chapter 5 attempts to overcome this computational complexity limitation by constraining the symmetric parts to be collections of superpixels. This chapter, draws on this idea of grouping superpixels, but will relax the symmetry constraint and focus on the more generic perceptual grouping rule of closure.

Further down the spectrum of prior knowledge are methods based on weaker shape priors than parallelism and symmetry. For example, Jacobs [44] uses convexity as well as gap to extract closed contours by grouping straight line segments. A less restrictive measure is that of compactness, which can be attained by normalizing the gap by area (Estrada and Jepson [29, 30], Stahl and Wang [99]). Some measure of internal homogeneity can also be used (Estrada and Jepson [30], Stahl and Wang [99]), provided that the inside of the region is easily accessible.

Finally we come to the most general methods that compute closure using only very weak shape priors, such as continuity and proximity. The most basic closure-based cost function uses a notion of boundary gap, which is a measure of missing image edges along the closed contour. Elder and Zucker [27] model the probability of a connection between two adjacent contour fragments, and find contour cycles using a shortest path algorithm. Wang et al. [103] optimize a measure of average gap using the ratio cut approach. However, a measure based purely on the total boundary gap is insufficient for perceptual closure, and Elder and Zucker [26] argue that the distribution of gaps along the contour is also very important. Williams and Hanson

[107] addressed the problem of perceptual completion of occluded surfaces, formulated as the problem of computing a labeled knot-diagram representing a set of occluded surfaces from observed image contours. While formulated as an elegant combinatorial optimization problem, for which an optimal solution was available, the approach was not tested on real scenes.

All the above methods suffer from the high complexity of choosing the right closure from a sea of contour fragments. To cope with this complexity, they either resort to heuristics to prune the search (e.g., [44]) or constrain the search space by other means (e.g., restricting the closure to alternating gap/non-gap cycles [99]). Zhu et al. [112] propose to solve this hard grouping problem by embedding the edge fragments into polar coordinates such that closed contours correspond to circles in that space; however, their goal is to better detect object contours, and they stop short of grouping the contours into closed boundaries. The method of Jermyn and Ishikawa [46] is perhaps the closest to our work. Similar to [103, 99], they minimize closure costs using ratio cuts, but unlike [103, 99] who operate on contour fragments, [46] works directly with pixels in a 4-connected image grid. It enables the authors to minimize many different closure costs (including our own) by globally minimizing ratio cuts in a simply connected planar graph. However, individual pixels provide poor scope for gap computation. In contrast, our superpixels not only provide greater scope for gap computation (which in our case is learned), but provide greater scope for the incorporation of internal appearance-based affinity. Finally, while their solution is optimal, it does not provide a set of optimal solutions that capture closures at multiple scales.

In this chapter, our goal is to find closed contour groups in an efficient manner. To that end, we use superpixels to constrain the search space of the resulting closures. Superpixels also provide an easy way to access internal region information (such as region area). Moreover, superpixel boundaries provide better scope for gap computation, as opposed to most previous methods that linearize the output of an edge detector and fill the gaps with straight line fragments. On the optimization side, we show that parametric maxflow [50] can be used not only to recover the global optimum of closure costs similar to that of Stahl and Wang [99] and Jermyn and Ishikawa [46], but can also be used to recover a multiscale set of closure hypotheses.

4.3 Problem formulation

As mentioned in Section 4.1, our framework reduces grouping complexity by restricting closure to lie along superpixel boundaries. Given a contour image $I(x, y)$[1], we first segment it into N superpixels using a modified version of the superpixel segmentation method of Mori et al. [70] ([70] uses the Pb edge detector [63], while we use globalPb [61]). If we let X_i be a binary indicator variable for the i-th superpixel, the vector $\vec{X}$ yields a full labeling of the superpixels of I as figure (1) or ground (0). Recall that our goal will be to select a maximal set of superpixels which have high spatial coherence and whose boundary has strong contour support in the image. Drawing on Stahl and Wang [99], we define our closure cost to be $C(\vec{X}) = \frac{G(\vec{X})}{A(\vec{X})}$, where $G(\vec{X})$ is the boundary gap along the perimeter of (the "on" superpixels of) $\vec{X}$, and $A(\vec{X})$ is its area. Boundary gap is a measure of the disagreement between the boundary of $\vec{X}$ and is defined to be $G(\vec{X}) = P(\vec{X}) - E(\vec{X})$, where $P(\vec{X})$ is the perimeter of $\vec{X}$ and $E(\vec{X})$ is the "edginess" of the boundary of $\vec{X}$. Out of the total number of pixels along the boundary of $\vec{X}$, $P(\vec{X})$, edginess is the number of edge pixels, with the edginess of image boundary pixels defined to be 0.

In order to facilitate the optimization of this cost using an optimal graph cut-based approach (see Section 4.5), we must decompose the cost function into unary and pairwise terms of the variables in X. Let P_i be the perimeter length of superpixel i and let P_{ij} be the length of the shared edge between superpixels i and j. Similarly, let E_i be the edginess of superpixel i's boundary, and E_{ij} be the edginess for the shared boundary between superpixels. Let the superpixel and shared superpixel edge gaps be $G_i = P_i - E_i$ and $G_{ij} = P_{ij} - E_{ij}$ respectively.

[1]The contour image takes the form of a globalPb image [61].

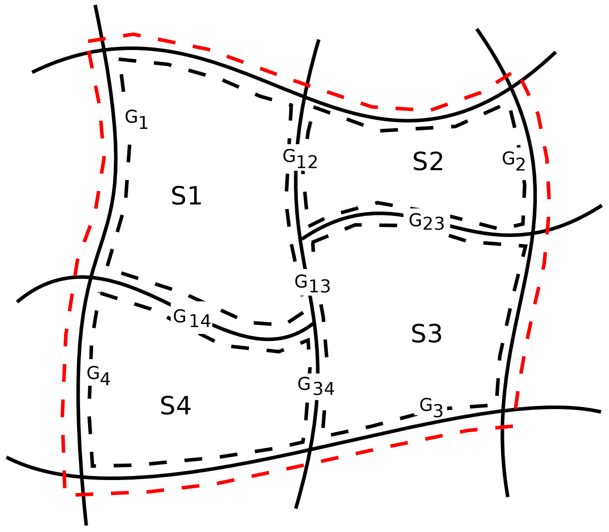


Figure 4.2: Boundary gap computation over superpixel graph. S_1, S_2, S_3, and S_4 correspond to superpixels that were selected. G_i and G_{ij} are the boundary gap of superpixel i and the gap on the edge between superpixels i and j respectively . The gap along the boundary of the selection (red) is then $G_{1234} = G_1 + G_2 + G_3 + G_4 - 2\,(G_{12} + G_{13} + G_{14} + G_{23} + G_{34})$.

Finally, let A_i be the area of superpixel i. Our closure cost becomes:

$$C(\vec{X}) = \frac{\sum_i G_i X_i - 2\sum_{i<j} G_{ij} X_i X_j}{\sum_i A_i X_i} \tag{4.1}$$

The denominator in the above ratio simply adds the individual areas of all the superpixels that were selected. Normalization by area not only promotes spatial coherence[2] but also promotes compactness; as we shall see in Section 4.6, given two possible paths (with strong edge support) a closure may take, it will prefer a compact path over one with deep concavities. The numerator in the above cost is more complicated. To compute the gap along the perimeter, we first add the individual gaps of all the selected superpixels. However, for selected superpixels that share boundaries, adding individual superpixel gaps would add gaps that are not on the boundary of the selection. For every internal boundary, the gap over that boundary was counted twice (once for each of the superpixels that share the boundary). Therefore, we subtract the gap

[2]While spatial coherence is promoted, it is not guaranteed. Since minimizing Eqn. 4.1 can occasionally result in disconnected sets of superpixels, we further guarantee connectedness by selecting the largest-area connected component of $\vec{X}$.

twice for all internal boundaries. Note that if two superpixels do not have a shared boundary, then both P_{ij} and E_{ij} (and thus G_{ij}) will be 0. Figure 4.2 gives an example of gap computation over a simple superpixel graph. In the next section, we introduce our gap measure, and show how it can be learned from training data.

4.4 Learning the Gap Measure

Most approaches to detecting contour closure (e.g., [99]) typically define gap as simply the length of the missing contour fragments, i.e., the length of that portion of the closure for which no image edges exist. In order to ground our gap measure using image evidence, as well as incorporate multiple contour features for gap computation, we choose to learn the gap from ground truth. Remember from Section 4.3 that for a pair of superpixels i and j, the gap on the edge between them is $G_{ij} = P_{ij} - E_{ij}$. Specifically, if $\vec{EP}_{ij}$ is the set of pixels on the superpixel edge (i, j), then $P_{ij} = |\vec{EP}_{ij}|$ and $E_{ij} = \sum_{p \in \vec{EP}_{ij}} E^p_{ij}$, where $E^p_{ij} = \left[P(\vec{f^p}) > T_e \right]$ is an edge indicator for pixel p ($P(\cdot)$ is a logistic regressor and $\vec{f^p}$ is a feature vector for the pixel p). Notice that we threshold the edginess measure instead of using it directly. T_e is a necessary threshold on the edginess measure. Since the distribution of edges in the training set is not necessarily the same as that for test images, this parameter controls the contribution of weak edges. Moreover, T_e lets us control the relative (to the area) effect of the gap on the closure cost (similar to α in [99]). Decreasing it results in many smaller structures being detected and causes more potential solutions to be generated. We analyze the performance of our method as a function of this parameter in Section 4.6.

Given a pixel p on the superpixel boundary, the feature vector $\vec{f^p}$ is a function of both the local geometry of the superpixel boundary and the detected image edge evidence in the neighborhood of the superpixel boundary pixel. This feature vector consists of four features (see Figure 4.3):

1. Distance to the nearest image edge; closer edges provide stronger evidence.

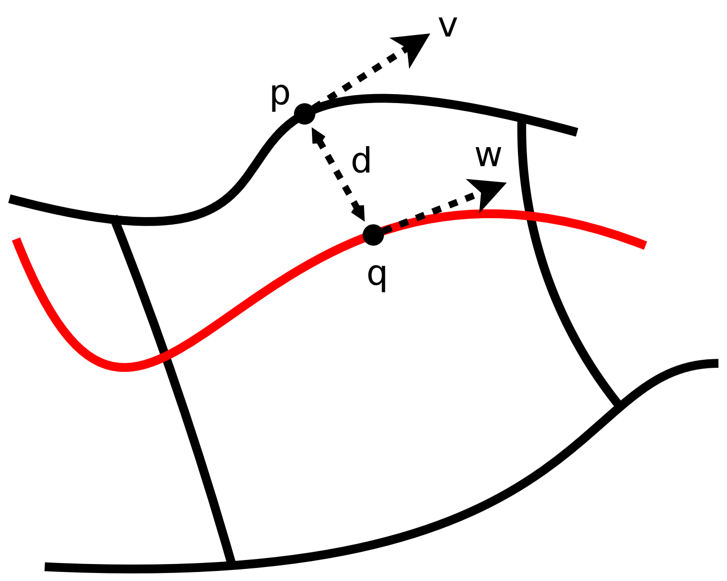


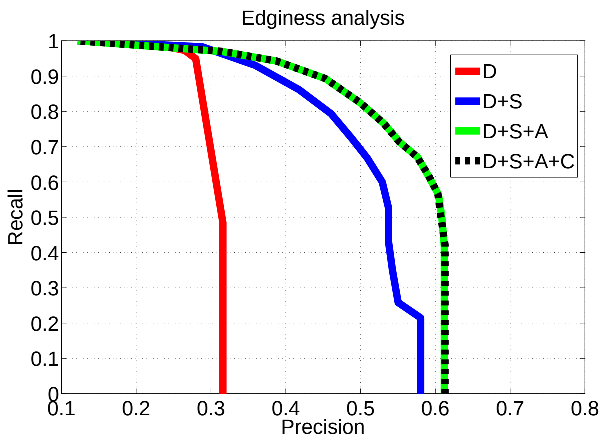
Figure 4.3: Contour features for learning the gap measure. Black curves correspond to superpixel boundaries, while the red curve corresponds to detected image edges. The features that are used for edge weight computation at superpixel boundary pixel p are: 1) distance d between p and q, where q is the closest point to p among the detected image edges; 2) image edge strength at q; 3) the alignment, computed as the absolute value of the cosine of the angle between v and w; and 4) the smoothness, computed as the squared curvature at p.

2. Strength of the nearest image edge; stronger edges provide stronger evidence.

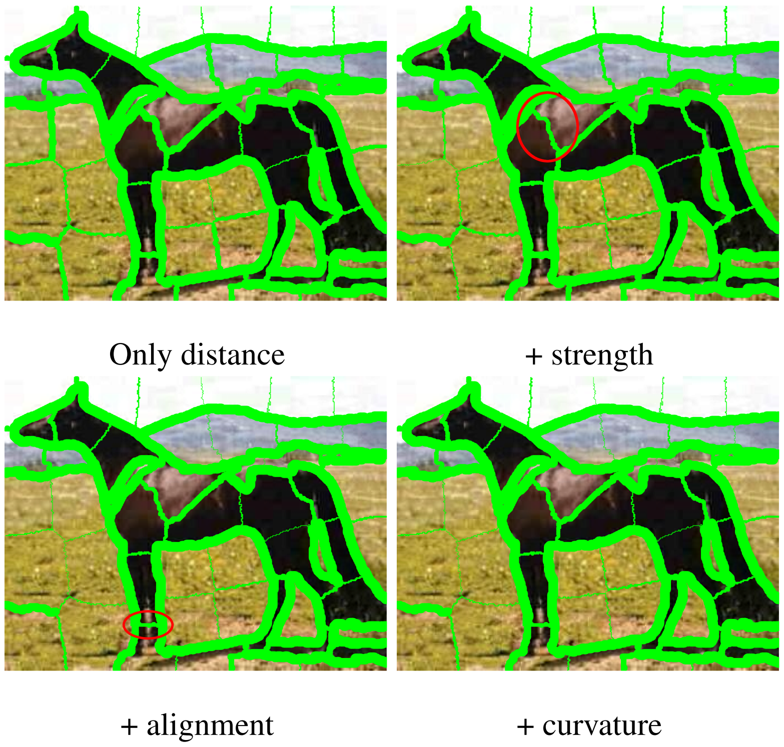
3. Alignment between the tangent to the superpixel boundary pixel and the tangent to the nearest image edge; aligned edges provide stronger evidence.

4. Squared curvature of the superpixel edge at a point p.

Given a dataset of images with manually labeled figure/ground, we map the ground truth onto superpixels. Our training set is composed of all the pixels falling on superpixel boundaries and is used to train a logistic classifier over a feature vector $\vec{f^p}$. In addition to learning from all four of the above features, we tried learning from subsets of the features. Figure 4.4 illustrates the effect of incrementally adding more features; the thickness of each superpixel edge in Figure 4.4(b) corresponds to the average edge probability of its superpixel boundary pixels.



(a)



(b)

Figure 4.4: Effect of different features on gap. (a) Quantitative evaluation - precision/recall of contour points using different features (D - Distance, S - Strength, A - Alignment, C - Curvature); (b) Qualitative evaluation (ordered left to right, top to bottom). For example, the superpixel edges that cross the legs become weaker as alignment is added and the shadow edge on the body becomes weaker as strength is added (red ellipses mark edges where the change is particularly visible).

Using all four features results in the best performance[3], in terms of retaining object boundary edges while suppressing other edges.

4.5 Optimization framework

It has been known for some time that ratios of real variables that adhere to certain constraints can be minimized globally [25]. Instead of minimizing the ratio $R(x) = \frac{P(x)}{Q(x)}$ directly, one can minimize a parametrized difference $E(x, \lambda) = P(x) - \lambda Q(x)$. It can be shown that the optimal λ corresponds to the optimal ratio $\frac{P(x)}{Q(x)}$. The constraints on the ratio guarantee that the resulting difference is concave and thus can be minimized globally.

In the case of binary variables, ratio minimization can be reduced to solving a parametric maxflow problem. Kolmogorov et al. [50] showed that under certain constraints on the ratio $R(x)$, the energy $E(x, \lambda)$ is submodular and can thus be minimized globally in polynomial time using min-cuts. Converting our closure cost $C(\vec{X})$ in Equation 4.1 to a parametrized difference results in a submodular cost $C(\vec{X}, \lambda)$, making the method in [50] applicable for minimizing the ratio $C(\vec{X})$.

In fact, the method in [50] does not simply optimize the ratio $R(x)$, but finds all intervals of λ (and the corresponding x) for which the solution x remains constant. The interval boundaries are called breakpoints, and while the smallest breakpoint λ_0 corresponds to the optimal ratio $R(x)$, consecutively larger breakpoints $\lambda_1, \lambda_2, \ldots$ are also related to ratio optimization. Kolmogorov et al. show that the optimal solution x^* of $E(x, \lambda)$ in the interval $[\lambda_i, \lambda_{i+1}]$, is also an optimal solution of $\min_{Q(x) \geq T} R(x)$, where $T = Q(x^*)$. In case of optimizing the closure cost in Equation 4.1, using parametric maxflow results in a multiscale set of optimal closure solutions under increasing area thresholds.

In general, the method in [50] can be exponential if the number of breakpoints is exponential, but is polynomial for obtaining a global optimum. For monotonic parametric maxflow

[3] Adding the curvature feature has a very marginal effect as can be seen both in Figure 4.4(a) and Figure 4.4(b).

[50], as is the case with the closure cost function in Equation 4.1, the number of breakpoints is at most $N + 1$, where N is the number of superpixels. Thus in our case finding all the breakpoints is polynomial in the number of superpixels. Moreover, the solutions exhibit nestedness, such that solutions for larger λs contain the solutions for smaller λs, resulting in hierarchical closure hypotheses. In our experiments, a solution is obtained in a fraction of a second for a superpixel graph of 200 superpixels[4], as there are typically less than 20 breakpoints. In Section 4.6 we analyze the number of breakpoints as a function of superpixel density and edginess thresholds.

4.6 Results

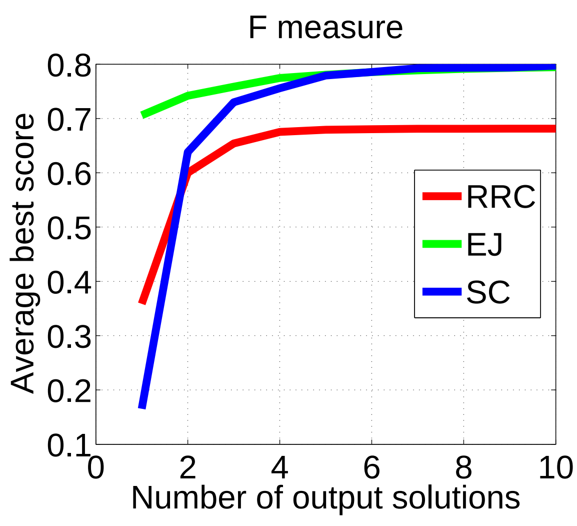
We compare our work, which we refer to as *superpixel closure*[5] (SC), to two other contour grouping methods: Estrada and Jepson (EJ) [30] and a version of ratio contours (RRC) from Stahl and Wang [99]. We provide a qualitative evaluation on various images (see Figure 4.7), as well as quantitative evaluation on two datasets, including the Weizmann Horse Database (WHD) [8] and the Weizmann Segmentation Database (WSD) [1]. Learning the gap measure (Section 4.4) is accomplished on the first 30 images from WHD. For testing, we use 170 additional images from WHD and all 100 images from WSD.

4.6.1 Quantitative Evaluation

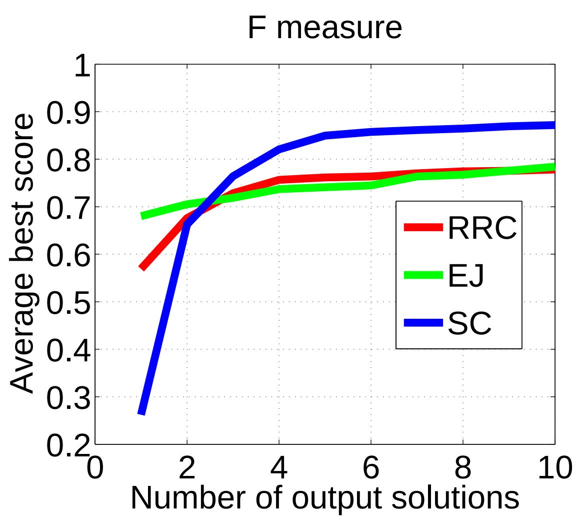
For a quantitative evaluation of the results, we use the **F-measure**, $F = \frac{2RP}{R+P}$, where R and P are recall and precision, respectively, of the solution relative to the ground truth. Specifically, if A is the set of pixels corresponding to the solution and A_{gt} is the ground truth, then $R = \frac{|A \cap A_{gt}|}{|A_{gt}|}$ and $P = \frac{|A \cap A_{gt}|}{|A|}$. Given K solutions, we select the solution with the best F-measure relative to

[4]In case of tens of thousands of superpixels, as is the case for spatiotemporal closure detection in Section 4.7.3, a solution can still be obtained in a matter of seconds.

[5]Matlab code for our method is available at `http://www.cs.toronto.edu/~babalex/closure_code.tgz`.



WHD



WSD

Figure 4.5: Quantitative results. We compare our results (SC) to two other algorithms: Estrada and Jepson [30] (EJ) and Ratio Contours [99] (RRC).

the ground truth. We average the "per-image" F-measure for all the images (and three ground truth segmentations in WSD) in a dataset and report the result.

Figure 4.5 shows the results of the three methods for increasing values of K. We chose the parameters that lead to the best performance for $K = 10$ for all three algorithms and fixed them for the entire experiment. For EJ, we used a Normalized Affinity Threshold (τ_{affty}) of 0.01, with the line segments generated by fitting the globalPb output. For RRC, we used $\lambda = 0$ and $\alpha = 1$. Here, we could not give the algorithm globalPb-based line segments, and thus use the method's own line segments generated from a Canny edge response. For our method, we fixed the number of superpixels to 200 and set $T_e = 0.05$, giving us best performance at the high range of K. Since the resulting solutions can be thought of as shape hypotheses for object recognition, we believe that the performance for some reasonably small value of $K > 1$ is more important than aiming to obtain a single best contour ($K = 1$)[6]. For $K = 10$, SC

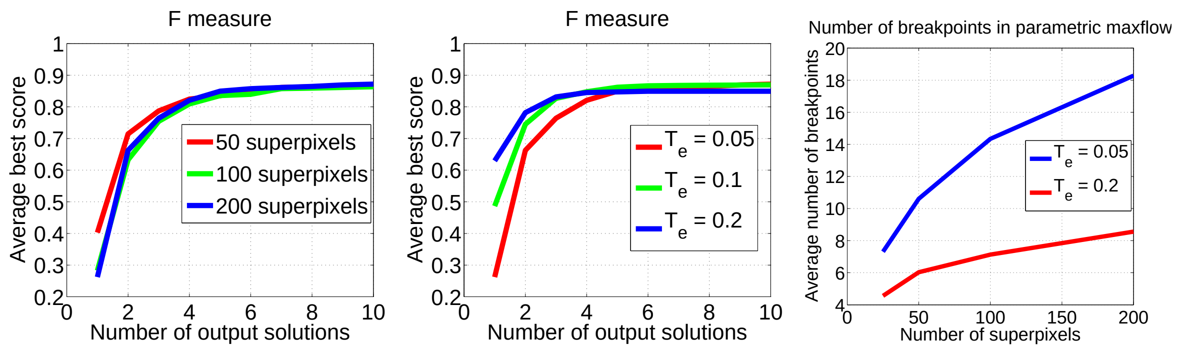
[6]SC can be tuned (see Figure 4.6) to perform better for $K = 1$ at a small expense of performance for higher K's.

(EJ, RRC) obtains an average F-measure of 79.72% (79.44%, 68.13%) on WHD and 87.19%[7] (78.44%, 77.82%) on WSD.

We outperform the competing approaches on both datasets for a setting of $K = 10$ (obtaining a comparable performance to EJ on the horses dataset), which we attribute to the superpixel formulation, as well as the optimal closure finding method in our framework. On the WHD, both SC and RRC perform significantly worse than on WSD, while EJ performs the same. This is likely due to the lower compactness of objects in the horse dataset (average isoperimetric ratio of 0.15, compared to 0.4 in WSD). Moreover, in many images there is a more compact path that includes the gap between the horse's legs due to shadow or ground edges. In addition, a significant number of images in the horse dataset have a picture frame boundary around the image. These boundaries provide the largest and most compact solutions, and are therefore found by SC instead of finding the horse. Finally, an interesting fact is that EJ still performs well on the horse dataset (unlike SC and RRC). This is most likely due to its reliance on internal appearance, which is definitely homogeneous in the case of horses. Since T_e is set so low ($T_e = 0.05$), we detect many small structures, capturing texture elements or object parts whose closure cost is lower than that of the actual figure object. As a result, our performance is poor for low values of K, but it is better at the high range of K.

Figure 4.6 shows the change in performance of our algorithm as we change the number of superpixels and vary T_e. Note that our dataset contains mostly large objects with a relatively strong boundary support. Therefore, coarse superpixel resolutions, which prevent the detection of small structures, lead to good performance for low values of K (Figure 4.6(a)). In general, higher superpixel density results in a very marginal performance gain for large values of K. Similar effect can be observed when changing the threshold T_e, since objects in our dataset typically have stronger edge support compared to other structures. Increasing the threshold T_e (Figure 4.6(b)) reduces the detection of small objects and improves performance at the

[7]For WSD, there are three ground truth segmentations per image. If we instead choose the closest of the three ground truth segmentations per image (as opposed to taking the average), our score on WSD improves to 88.76%.



(a) (b) (c)

Figure 4.6: Varying the parameters of our method (evaluation on WSD). (a) Varying the number of superpixels for a fixed edge threshold $T_e = 0.05$. (b) Varying the edge threshold T_e for a fixed number of superpixels (200). (c) Number of breakpoints as a function of superpixel density and edge threshold T_e.

low range of K, but also hurts the detection of objects with weak edges and thus results in slightly poorer performance at the high range of K. Nevertheless, in cases of large figure objects with strong boundary edges, coarser superpixel resolutions and higher edge thresholds are preferable.

The complexity of our approach is directly proportional to the number of breakpoints returned by parametric maxflow. Figure 4.6(c) shows how the number of breakpoints varies as a function of superpixel resolution and edge threshold. We also compare the running times of the three methods on WSD (average image size of 300×290 pixels). On a 2.6GHz Dual Core Intel CPU with 4GB of memory, setting the methods to retrieve $K = 10$ best contours, the average running times per image are: SC (not including globalPb edge detection and superpixel segmentation) – 1.3 sec, EJ (not including globalPb edge detection) – 23 sec, RRC – 59 sec.

4.6.2 Qualitative Evaluation

In addition to the quantitative evaluation, we also provide a qualitative evaluation of our method by testing it on images from the two datasets, as well as other images obtained from the in-

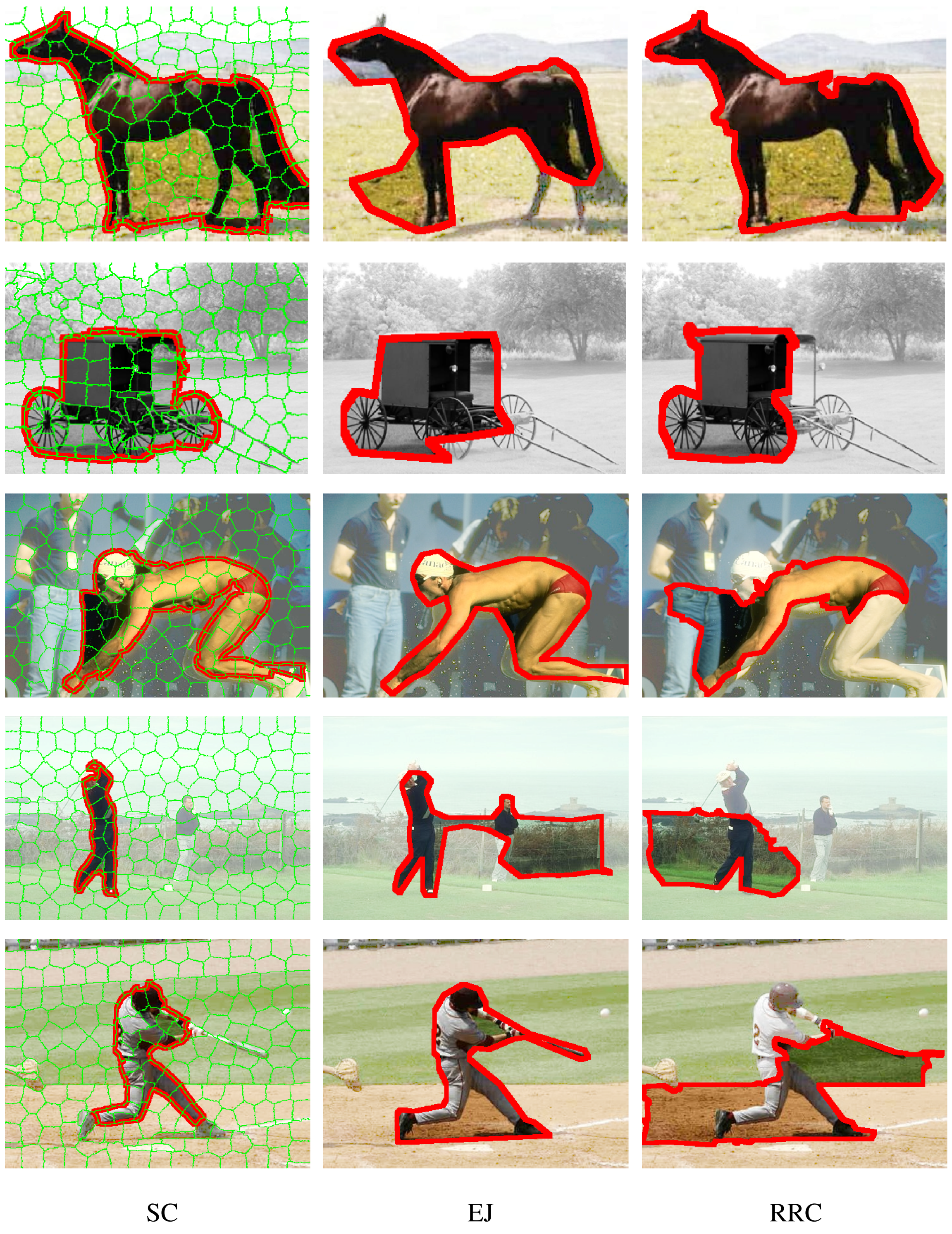


Figure 4.7: Qualitative results. We compare our results (left) to two other algorithms: Estrada and Jepson [30] (middle) and Ratio Contours [99] (right).

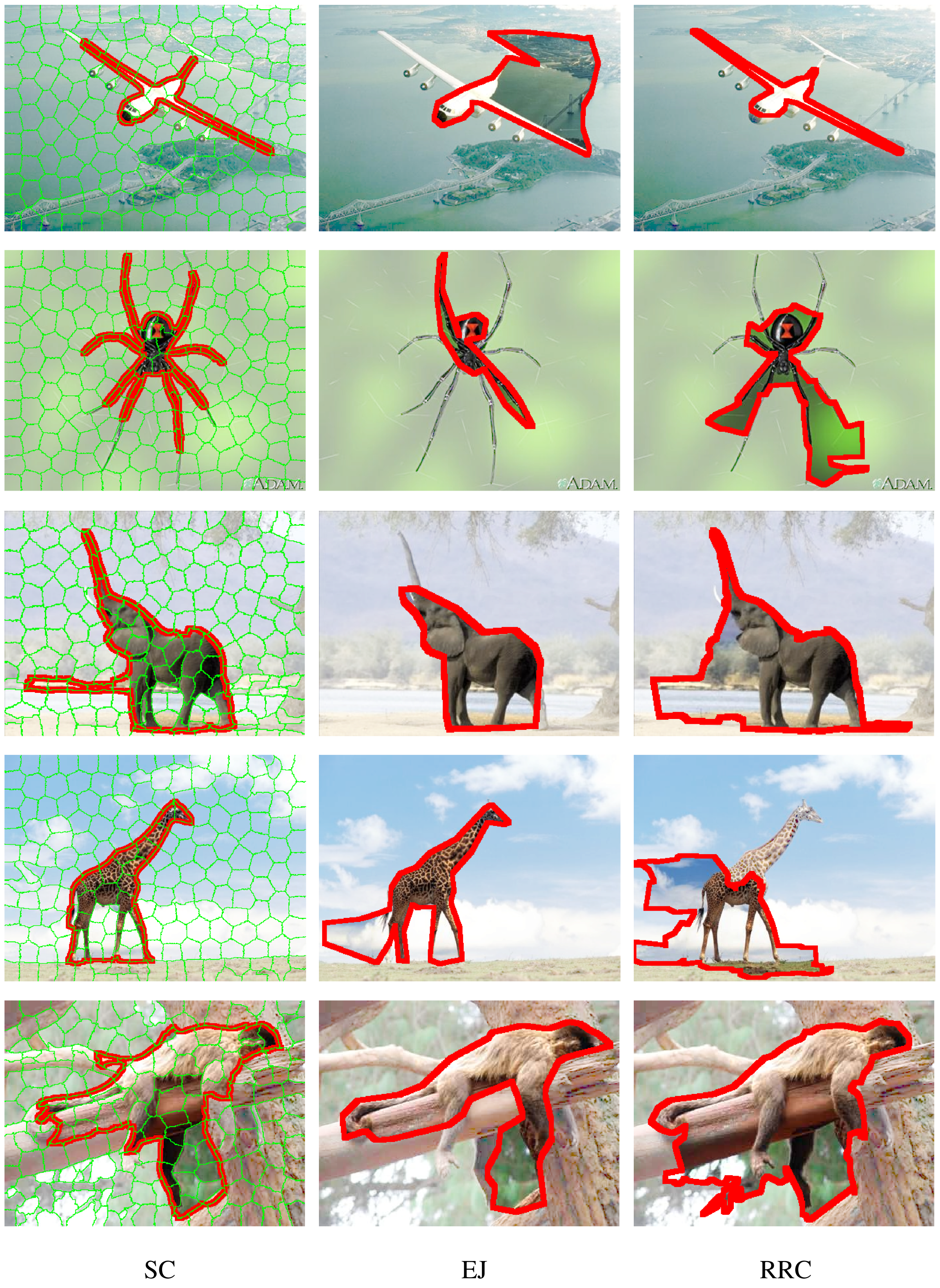


Figure 4.7

ternet. Figure 4.7 illustrates the performance of our method compared to the two competing approaches[8]. We manually select the best of 10 solutions for each method. Notice that the detected contours in our framework lie closer to the true object contours since the superpixel edges, even in the presence of a gap, lie closer to object edges than the linearized contours detected by the other algorithms. We pleasantly observed that our framework is not constrained to obtain compact solutions as is usually the case when one is normalizing perimeter by area. This is clearly visible in the image of a spider, where very thin legs are segmented since that represents the best closure solution. However, this is not always the case, for if there is a more compact contour that is not losing on gap, it will be preferred. This is the reason for the filled gap between the horse's legs or the filled gap between the carriage's wheels in the first two images. Note that for the horse image, EJ obtains a better solution by relying on the homogeneous appearance inside the horse. Finally, our method relies on superpixels to oversegment the object, which might not be the case for thin structures or when weak object contours are present. We still detect thin structures, such as the spider's legs, if good superpixels were found due to strong image edges. For weaker edges, however, thin structures are harder to capture (the bat of the baseball player, for example). Weak edges are also the cause for bleeding seen in the elephant example, where the upper portion of the front leg has a weak edge w.r.t. the background.

4.7 Extensions

4.7.1 Using internal homogeneity

As mentioned in Section 4.1, our superpixel formulation also facilitates the incorporation of appearance information, when it is both available and appropriate. The cost function in Equation 4.1 can be easily modified to incorporate a term which reflects the degree to which adjacent

[8]Supplementary material (`http://www.cs.toronto.edu/~babalex/closure_supplementary.tgz`) contains the results of our algorithm for all the images in both datasets.

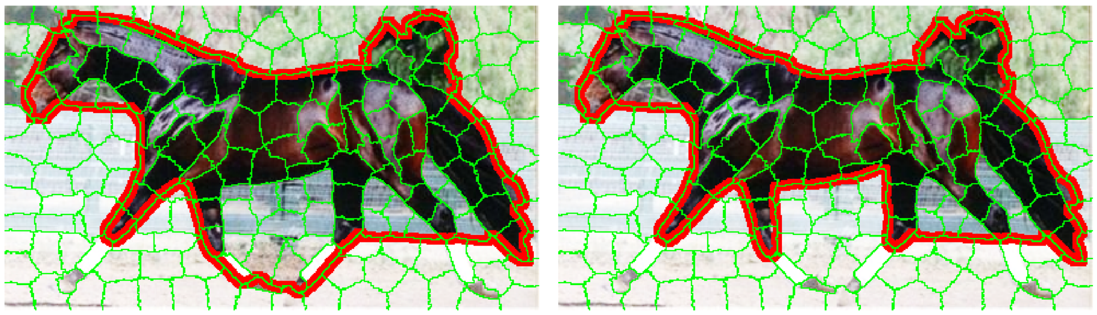


Figure 4.8: Using internal appearance homogeneity. For objects with strong internal homogeneity of appearance, optimizing the cost in Eqn. 4.2 results in better performance (right) than optimizing the cost in Eqn. 4.1 (left). Note that the gap between the horse's legs was not included on the right due to its heterogeneous appearance w.r.t. the rest of the horse.

superpixels *inside* the superpixel selection, i.e., inside the closed contour, have high affinity. Assuming that we are given an affinity matrix W, such that W_{ij} is the affinity between two superpixels i and j, we modify our closure cost to be:

$$C_{affty}(\vec{X}) = \frac{\sum_i G_i X_i - 2\sum_{i<j} G_{ij} X_i X_j}{\sum_{i<j} W_{ij} X_i X_j} \tag{4.2}$$

Compared to the cost in Equation 4.1, the numerator remains the same while the denominator changes to an internal homogeneity measure instead of the total object area. Minimizing this ratio results in minimizing the gap while maximizing the total affinity between the selected superpixels. Figure 4.8 shows an example where better results were achieved by exploiting appearance homogeneity.

4.7.2 Multiple superpixel scales

Though it might seem that the more superpixels we use, the better our method will perform, it is not always so. As seen in Figure 4.6(a), coarser superpixel scales constrain the solution more and thus perform better for low values of K. However, there is one additional advantage of using coarser superpixel scales. Since our superpixel algorithm does not produce hierarchical superpixels (since new superpixel boundaries may be introduced from finer to coarser scales), it

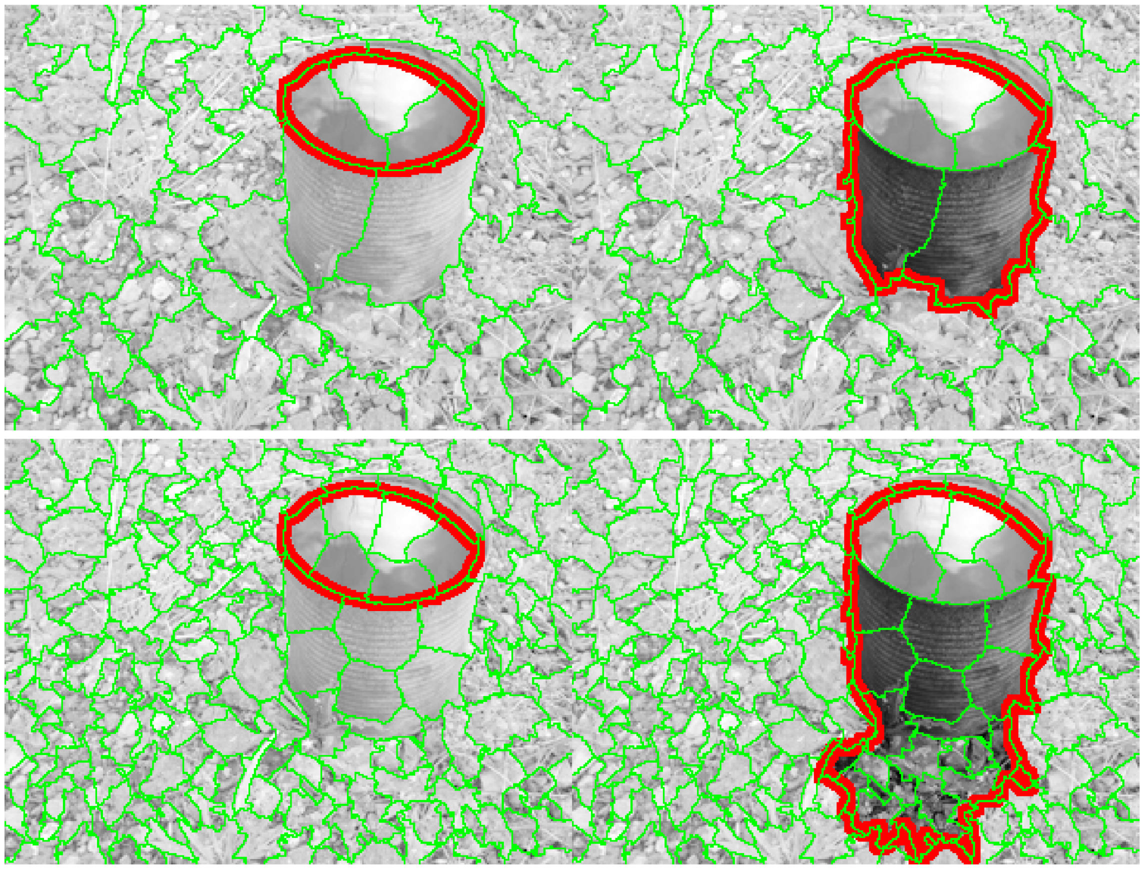


Figure 4.9: Multiscale results. Choosing the $K = 2$ top solutions yields better results in the case of 50 superpixels (top) than in the case of 200 superpixels (bottom).

is possible to occasionally have less undersegmentation at coarser scales. Figure 4.9 illustrates a situation where an object was segmented better at a coarser scale and consequently detected by our algorithm.

We tried a simple multiscale version of our algorithm where we merge the results from all scales. Specifically, we run our algorithm at four superpixel scales, obtaining 25, 50, 100, and 200 superpixels for each image. Setting $K = 10$ for each scale results in 40 solutions once the results are merged together. Since the performance of our method for a given scale does not significantly vary for $K > 10$, we do not select 10 of 40 solutions for the multiscale version, but instead retain all 40. Using the multiscale version increases the performance on WSD from 87.19% to 89.53%.

4.7.3 Spatiotemporal Closure

Introduction

Finding closures bottom up in 2D images is of the utmost importance, as doing so facilitates the implementation of higher-level vision tasks. The same is true for detecting closures in spatiotemporal domains, where closure hypotheses can serve object or action recognition in videos. This section extends our 2D closure detection approach to the spatiotemporal domain.

The most straightforward extension of our approach to 3D would have been to extract 3D superpixels (supervoxels) and define a 3D version of the closure cost in Equation 4.1. This would correspond to minimizing a measure of gap of a closed surface relative to the volume of that surface. However, such an approach requires the extraction of supervoxels. Unlike superpixels, the use of supervoxels is not popular in computer vision and we know of no robust approaches for extracting them. Moreover, a straightforward extension to 3D assumes a single measure of gap for all surface points. This assumption may not hold for spatiotemporal volumes, as one may require different measures of gap in space and time. For the above reasons, we extract superpixels for every frame and form a spatiotemporal graph where edges between superpixels encode superpixel affinity. This way, affinity can be thought of as a coarse approximation to the "edginess" measure defined for the 2D case, and low affinity would correspond to large gaps. Instead of minimizing Equation 4.1, we minimize the unbalanced normalized cuts cost, making the spatiotemporal closure detection closer to the method of Carreira and Sminchisescu [14].

Similar to the 2D closure extraction, our final goal in this extension is to obtain a small number of figure hypotheses, where each hypothesis corresponds to a coherent segment in space-time (Figure 4.10). To that end, we first segment each video frame using a modified superpixel framework presented in Chapter 3, taking special measures to make the superpixels more temporally coherent. We form a superpixel graph, where each superpixel is connected to its spatial neighbors within a frame as well as superpixels in temporally neighboring frames.

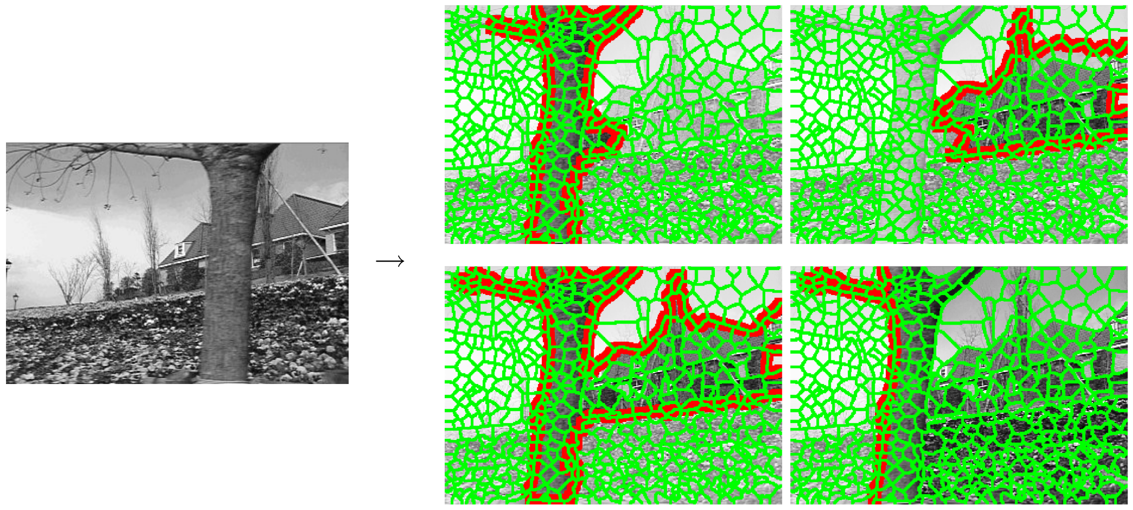


Figure 4.10: Generating multiple space-time figure/ground segmentations from a video.

We define affinities between adjacent superpixels in space and in time, incorporating both appearance and motion cues. We then use the parametric maxflow framework to obtain several unbalanced normalized cuts of this graph. We run our method on several test videos and show that it obtains a small number of figure/ground "tubes" that usually contain the objects of interest. We conclude with an analysis of the results and ideas for future work.

Problem formulation

Our goal is to find coherent space-time segments in a video. We formulate the segment-finding problem as a superpixel selection problem. Out of an exponential number of superpixel subsets we will select subsets with small unbalanced normalized cuts costs and show that they correspond to coherent spatiotemporal segments.

Given a superpixel segmentation of every frame in a video, we start by building a superpixel graph with spatial and temporal connections. Let $\vec{X}$ be an indicator vector for all the superpixels across all frames, with each element being in the set $\{0, 1\}$. We connect each superpixel to its spatial and temporal neighbors and define an affinity W_{ij} for each pair of neighboring superpixels i and j, encoding the similarity of the two superpixels. Setting $D_i = \sum_j W_{ij}$, we optimize the following closure cost:

$$C(X) = \frac{cut(\vec{X})}{volume(\vec{X})} = \frac{\sum_{ij} X_i(1 - X_j)W_{ij}}{\sum_i D_i X_i} = \frac{\sum_i D_i X_i - 2\sum_{i<j} X_i X_j W_{ij}}{\sum_i D_i X_i} \tag{4.3}$$

The above is called the unbalanced normalized cuts cost and, unlike the standard normalized cuts cost, can be minimized efficiently using parametric maxflow [50]. This cost is very similar to the 2D closure cost (Eqn. 4.1), with the exception that the numerator measures the cut instead of the gap and is normalized by affinity volume instead of by area. That said, we will show that the affinities W_{ij} can also include the length of the boundary between superpixels or their area to give larger superpixels a greater influence.

Algorithm details

Our algorithm consists of several stages. We start by extracting the superpixels for each frame of the video. Afterwards, we construct a superpixel graph where each superpixel is connected to its spatial and temporal neighbors. Each superpixel edge is assigned an affinity that measure the degree of superpixel similarity. Once the graph is built, we find optimal cuts using parametric maxflow. Finally, we post-process the solutions to detect connected components, remove similar or spurious results, and generate other potentially good solutions. The following subsections describe each of these stages[9].

Superpixel Extraction

We begin by extracting superpixels from every frame using the TurboPixels approach in Chapter 3. Instead of using the algorithm in its raw form, we modify it to obtain more temporally coherent superpixels. We start by extracting superpixels in the first frame using the original form of the superpixel algorithm in Chapter 3. Instead of reseeding the superpixels in the next frame on a regular grid, we use the current frame's superpixel to drive the seeding procedure.

[9]See the Approach Overview section at `http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html` for a graphical overview of the method.

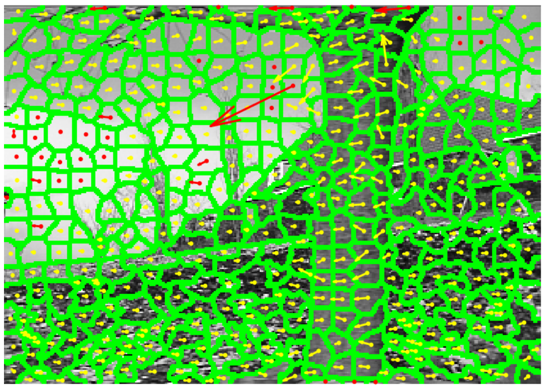


Figure 4.11: Superpixel flow. The arrow within each superpixel indicates the motion flow vector of this superpixel. Yellow arrows indicate reliable flows, while red arrows correspond to unreliable flows.

To that end, we first compute the optical flow using the Lucas-Kanade (LK) algorithm. The LK algorithm returns the flow for every pixel in every frame, together with a measure of reliability for each pixel flow. For every superpixel, we compute a weighted average of the flow over all the reliable pixels, where pixels that are closer to the superpixel centroid have larger weights. Superpixels with an insufficient number of reliably flowing pixels get a flow of $(0, 0)$. The result is a *superpixel flow*, with motion flow vector $\vec{V_i}$ for every superpixel i (Figure 4.11).

Taking the superpixel flow for every superpixel, we project the center of each superpixel to the next frame according to the computed flow. These projected centers serve as the initial seeds for the superpixel evolution in the next frame. We repeat this process for all the frames in the video, giving us a much more temporally stable superpixel segmentation. In addition, we also modify the superpixel algorithm to use a Pb-based [63] affinity rather than the original grayscale gradient-based affinity proposed in Chapter 3. Figure 4.12 shows the extracted superpixels for two temporally adjacent frames of the flower garden sequence using this modified approach and compares it to independent superpixel extraction at every frame using the original algorithm. Note that using the modified superpixel algorithm results in more temporally

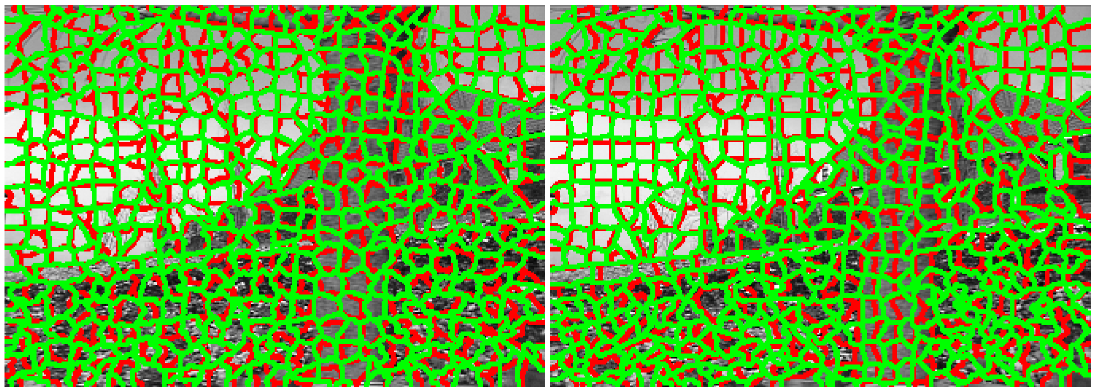


Figure 4.12: Superpixel extraction. We apply two different versions of the TurboPixels algorithm to two temporally adjacent frames (frame 1 and 3) in the flower garden sequence. Superpixel boundaries in frame 1 are marked in red and superpixel boundaries in frame 3 are marked in green. The modified superpixel algorithm (b) results in more accurate and temporally stable superpixels than the original superpixel algorithm (a). This is particularly visible on background regions where the scene motion is relatively small.

stable superpixels that better capture object boundaries[10].

Superpixel Affinity

Once the superpixels are extracted, we form spatial and temporal edges in the superpixel graph. Every edge is assigned an affinity W_{ij} that measures the similarity of the two superpixels (Figure 4.13).

To form spatial connections, we find the immediate spatial neighbors of each superpixel in each frame. Spatial neighbors of superpixel i are defined as superpixels in the same frame that share some boundary with superpixel i. The formation of temporal connections follows the same approach as was used in the superpixel extraction technique. Each superpixel in frame f

[10]See the Superpixel Extraction section at `http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html` for a better visualization of superpixel extraction.

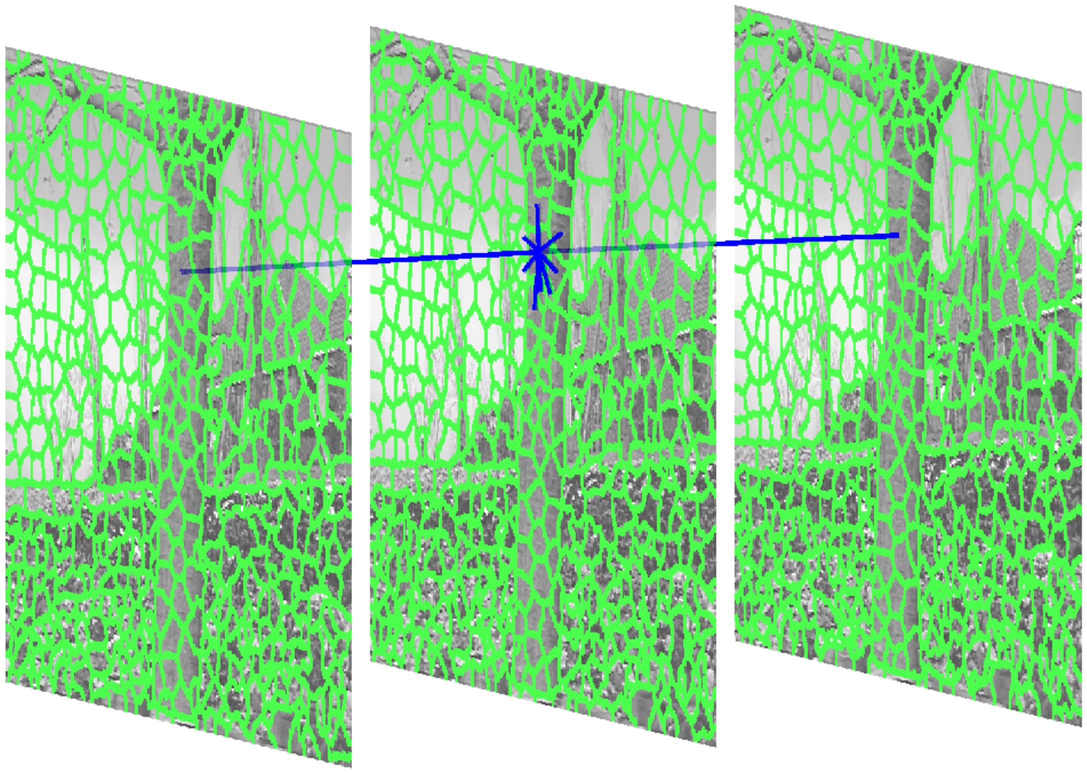


Figure 4.13: Superpixel graph construction, illustrating the typical connectivity for one superpixel. Every superpixel in each frame is connected to its immediate spatial neighbors in that frame. For every two consecutive frames, every superpixel in the first frame is connected to one superpixel in the second frame based on the superpixel flow vectors.

(except the superpixels in the last frame) is connected to one superpixel in frame $f+1$. The correspondence is determined based on the superpixel flow vectors. The center of superpixel i from frame f is projected to frame $f+1$ according to the superpixel flow $\vec{V_i}$. We form an edge between superpixels i and j, where superpixel j is the superpixel in frame $f+1$ that contains the projected center of superpixel i.

Motivated by [43], our superpixel affinity W_{ij} for a spatial edge (i,j) is defined as the combination of appearance (W_{ij}^{a}) and motion (W_{ij}^{m}) affinities. Appearance affinity is obtained by computing the histogram intersection distance of the grayscale (or color, if available) histograms of the two superpixel regions (we use 30 bin histograms for grayscale and $4 \times 4 \times 4$

histograms for RGB). For two N-dimensional normalized (summing to 1) histograms $\vec{H}^1$ and $\vec{H}^2$, their histogram intersection distance is equal to $\sum_{i=1}^{N} \min\left(\vec{H}^1{}_i, \vec{H}^2{}_i\right)$. Motion affinity is computed by comparing the flow vectors of the two superpixels, $\vec{V}_i$ and $\vec{V}_j$, and is equal to $W_{ij}^m = 1 - \frac{\|\vec{V}_i - \vec{V}_j\|}{\max\{\|\vec{V}_i\|,\|\vec{V}_j\|\}}$ capped to the range $(0,1)$. Since our superpixel graph construction incorporates superpixel flow already, we include the motion affinity only for spatial edges. Finally, to give larger superpixels more influence, we augment the affinity by weighting it with the product of areas of the two superpixels (A_i and A_j). Combining that with the goal of not grouping two superpixels if either their appearance or motion is dissimilar results in the following superpixel affinity:

$$W_{ij} = \begin{cases} A_i A_j \min\left(W_{ij}^a, W_{ij}^m\right), & (i,j) \text{ are in the same frame} \\ A_i A_j W_{ij}^a, & (i,j) \text{ are in different frames} \end{cases} \tag{4.4}$$

Optimal Cuts for Each Shot

At this point, we have a superpixel graph and thus can use the parametric maxflow framework to optimize the cost in Eqn. 4.3. However, prior to running the optimization framework, we first detect the shot boundaries in the video with the goal of independently finding closures for each shot.

Temporal superpixel edges across shot boundaries are unreliable. Thus if a video is composed of multiple shots, running the optimization on the whole video results in undesirable solutions. Since this is not the focus of this work, we take a very simplistic approach to shot boundary detection. Similar to the appearance affinity between superpixels, we compute an appearance affinity between consecutive frames by comparing the grayscale histograms of whole frames using the histogram intersection kernel. This results in a $F - 1$ dimensional vector of consecutive frame affinities (where F is the number of frames). The shot boundaries correspond to the detected minima in this vector (Figure 4.14). Given the detected shots, we build a subgraph for every shot by selecting the superpixels and the edges that are contained in the shot. We optimize the cost in Eqn. 4.3 for all the subgraphs and concatenate the results.

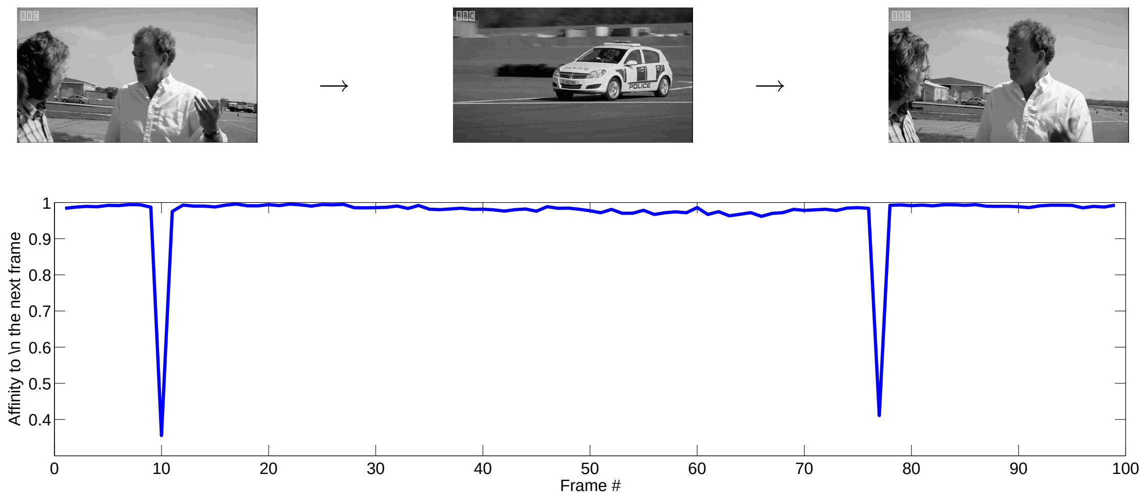


Figure 4.14: Shot detection by finding minima in consecutive frame affinities. The top row shows a video containing 3 shots. The shot changes from people to car at frame 11 and back to people at frame 78. The bottom row shows a corresponding drop in consecutive frame affinity for these frames. These minima are detected in order to find the shot boundaries.

Note that optimizing the cost in Eqn. 4.3 directly results in a trivial solution where all the superpixels are selected. Moreover, just as the edge threshold T_e was used in the 2D case in the computation of gap, we want to give the user the ability to weaken affinities in order to handle the cases of potential bleeding between foreground and background due to appearance or motion similarity. We solve the first problem by introducing large penalties for a subset of superpixels in the graph. Specifically, we run the optimization 6 times for each shot. In the first 4 runs all the superpixels on the left, right, top, and bottom frame boundary respectively are assigned a large penalty. In the 2 additional runs, we assign large penalties first to all top and bottom superpixels, and then to all left and right superpixels. To handle the second issue, we augment the closure affinity in Eqn. 4.4 to :

$$W'_{ij} = \begin{cases} A_i A_j \left(\min\left(W^a_{ij}, W^m_{ij}\right)\right)^{\alpha}, & (i,j) \text{ are in the same frame} \\ A_i A_j \left(W^a_{ij}\right)^{\alpha}, & (i,j) \text{ are in different frames} \end{cases} \tag{4.5}$$

The exponent α controls the contribution of weak affinities. Increasing the exponent effectively lowers all the affinities towards 0, thereby preventing bleeding, but also increases the relative difference between weak and strong affinities. In the results section, we will analyze the effect of changing α on performance.

Post-processing

Running parametric maxflow on the spatiotemporal superpixel graph results in hundreds and sometimes thousands of breakpoints. Some of the solutions differ by a very minor increase in area, while others contain multiple connected components. Furthermore, some desirable solutions are missed. We post-process the results to narrow down the number of solutions to a more manageable number and in the process generate additional good solutions.
Post-processing consists of the following 3 stages:

1. **Filtering solutions and generating new ones by analyzing the area change:** As previously stated, parametric maxflow results in solutions that minimize the cut with increasing area constraints. Some consecutive solutions differ by a very small increase in area. We filter out the solutions where such an increase is insignificant (less then 1% of relative area increase). Conversely, for all other solutions we detect consecutive solution pairs where the relative area increase is above a threshold (more than 5%) and generate a new solution subtracting the one superpixel subset from another.

2. **Selecting connected components and removing small solutions:** Some solutions up to this point contain only a few superpixels or select superpixels in a very small number of frames. We filter out these solutions by keeping only the solutions with at least 2 superpixels, with total area that is at least 1% of the frame area, and that participate in at least 5 frames. We run a connected component analysis for all the remaining solutions. Each solution that contains multiple connected components in space-time is split, generating one solution for each connected component.

3. **Removing duplicate solutions:** The above post-processing steps can result in the generation of duplicate solutions. In this final step we remove duplicate solutions.

For our test videos, this post-processing step reduces the number of solutions of a single run of parametric maxflow from several hundreds to an average of $20 - 80$ solutions.

Results

We perform a qualitative analysis of our approach on several short video sequences. Some sequences (such as the flower garden sequence) are grayscale, while others contain color. In the case of color sequences, we make use of this additional information, comparing color histograms instead of grayscale when computing superpixel affinities. The frame size for each video is on the order of 300×300 pixels, with the length of a video ranging from around 10 frames to 250 frames (hippo sequence). Based on quantitative evaluation (described in latter paragraphs), we set $\alpha = 6$ for our qualitative experiments. We also perform a quantitative evaluation on two datasets [100, 34], comparing different graph constructions and affinity variations, as well as evaluating our approach against standard normalized cuts on the same graphs. Timewise, the bottlenecks of the approach are the preprocessing steps: Pb edge detection, superpixel extraction, and optical flow computation, each taking several seconds per frame. Once a superpixel graph is built, each run of the optimization using parametric maxflow finishes in less than 5 seconds on the whole video, followed by all the postprocessing steps taking approximately 1 second.

Figure 4.15 shows our qualitative results. For each sequence we show 3 frames and visualize several interesting solutions for the chosen frames[11]. Note that in the first sequence (car + people) there are three shots, first of people, then of the car, and then returning back to people. Each of the three columns in the figure displays a single frame from each of these shots. In the car sequence, several objects of interest were successfully recovered, such as the car and the heads of the people. Moreover, a part of the car (windshield) is also recovered in one of the solutions, indicating that our method can be used for part based object recognition in videos or

[11] See the Results section at `http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html` for a video visualization of the results.

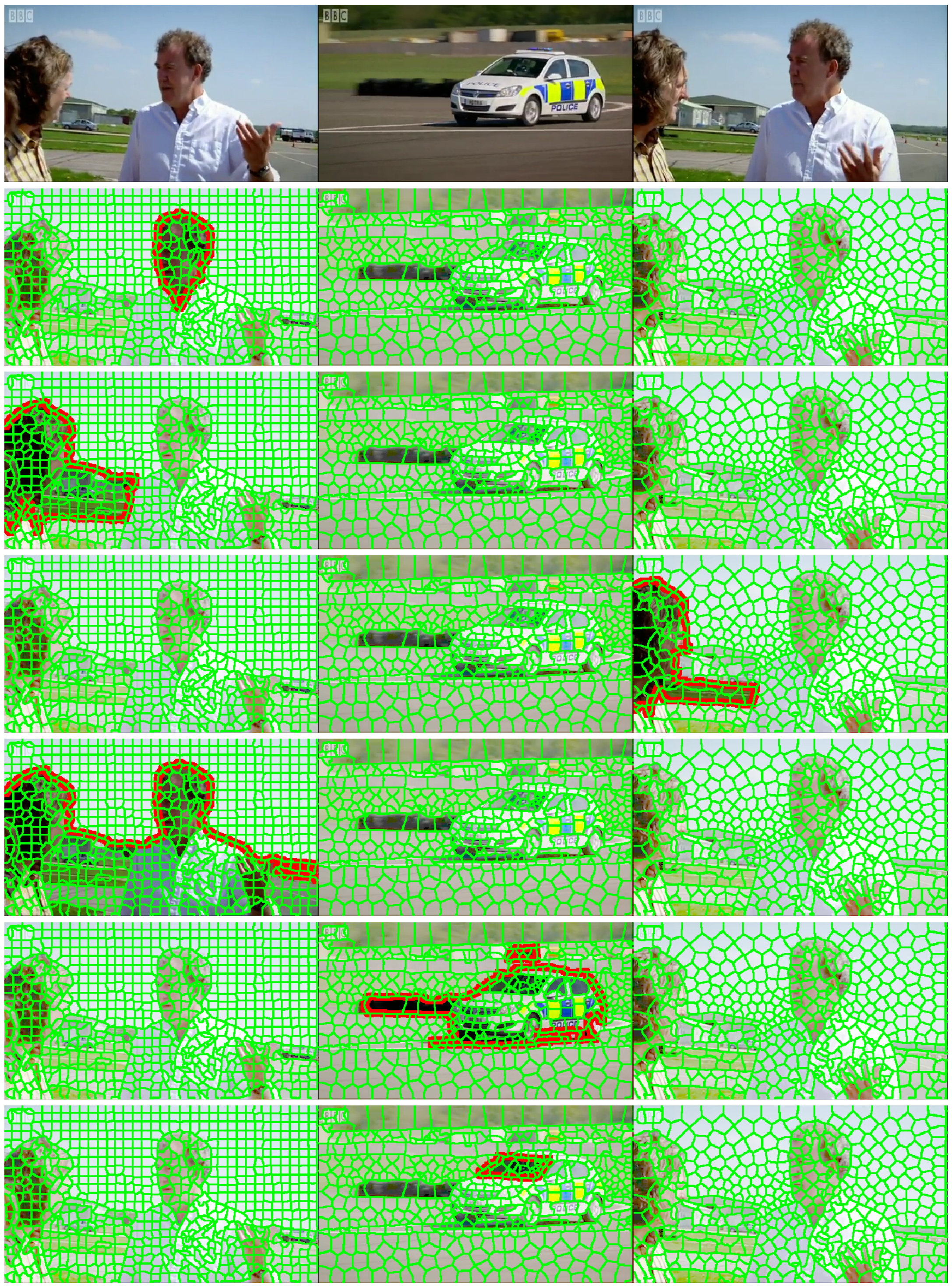


Figure 4.15: Qualitative video figure/ground segmentation results. First row displays 3 frames from a sequence, followed by several rows displaying interesting solutions.

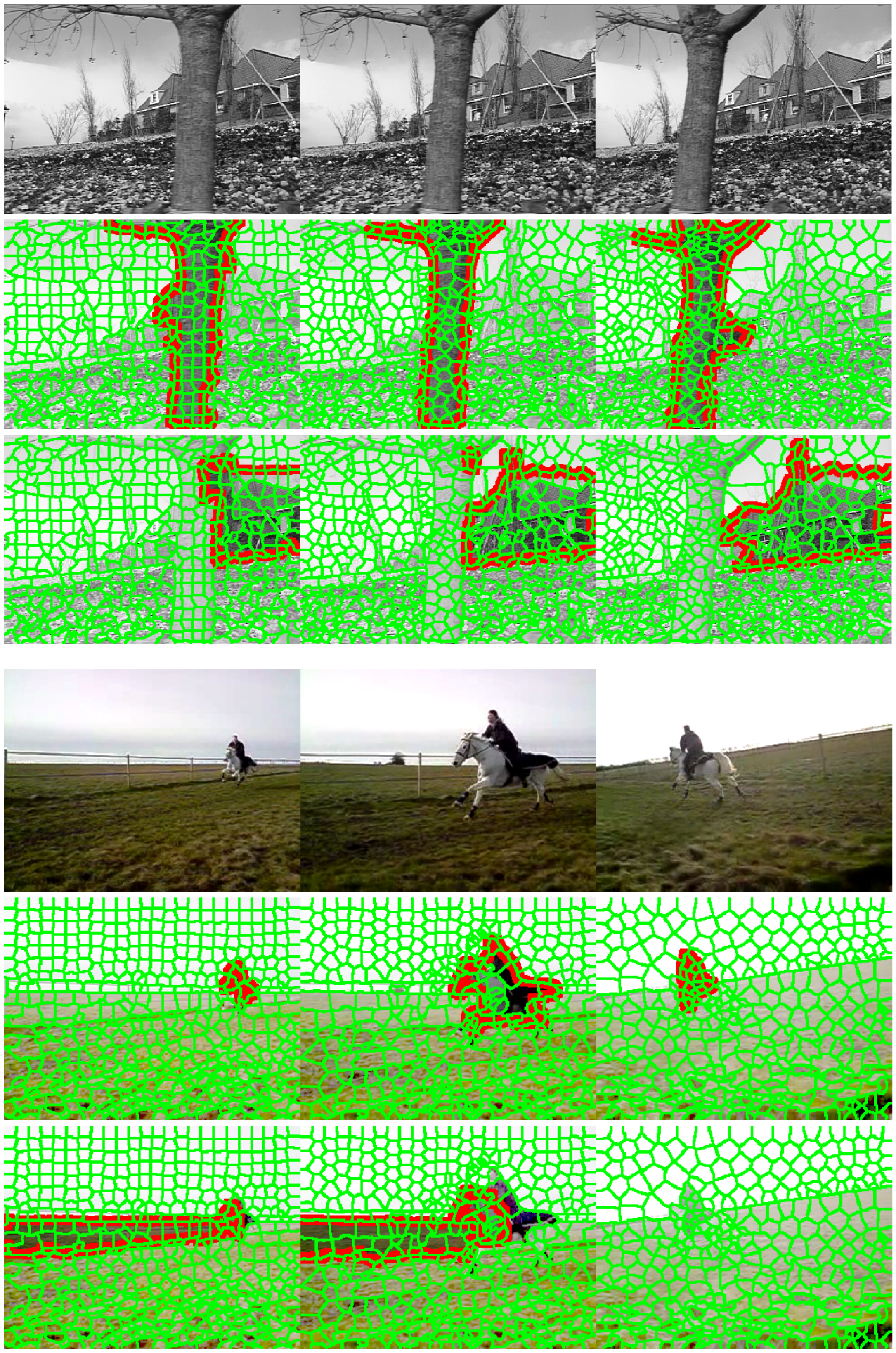


Figure 4.15

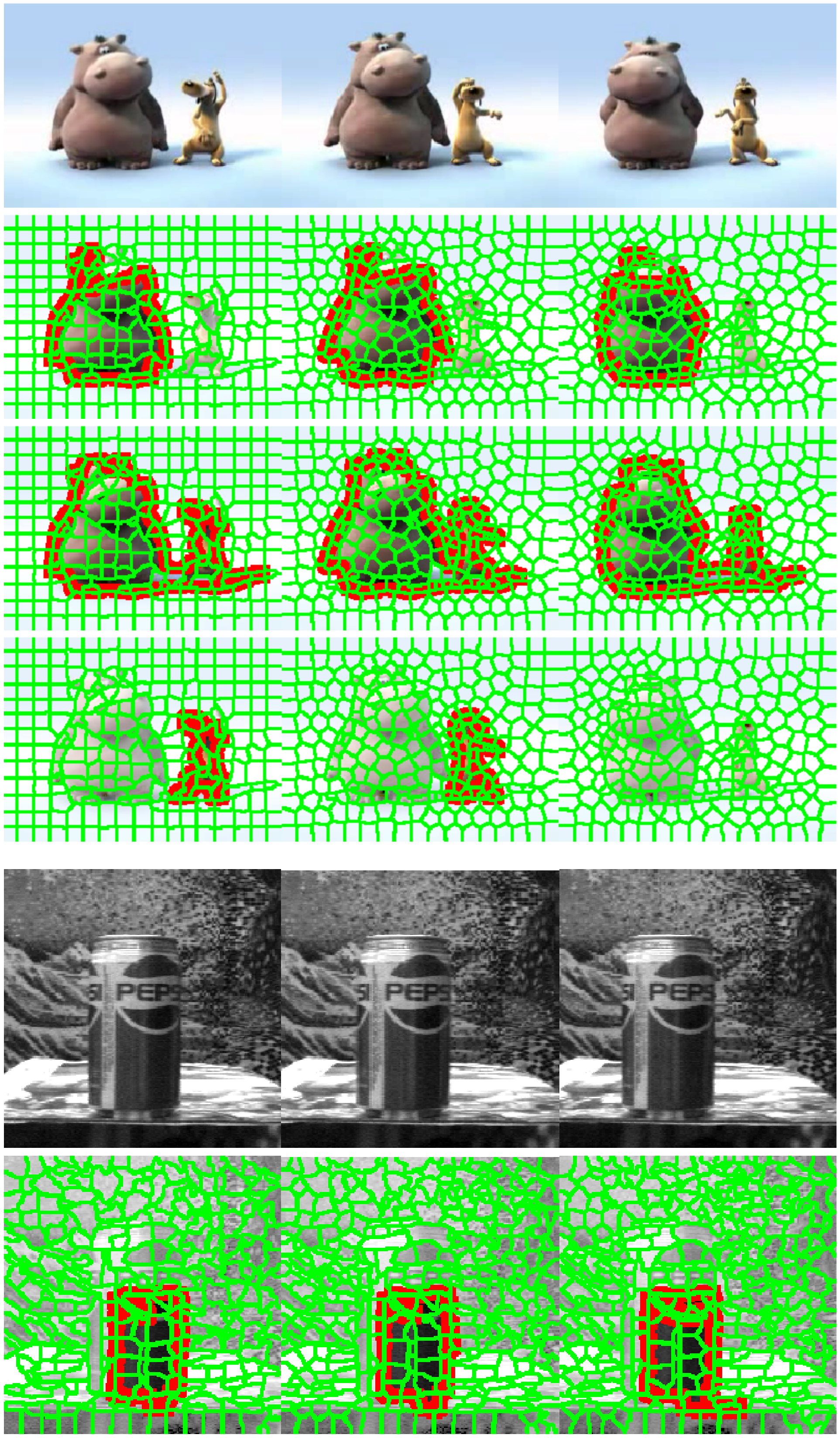


Figure 4.15

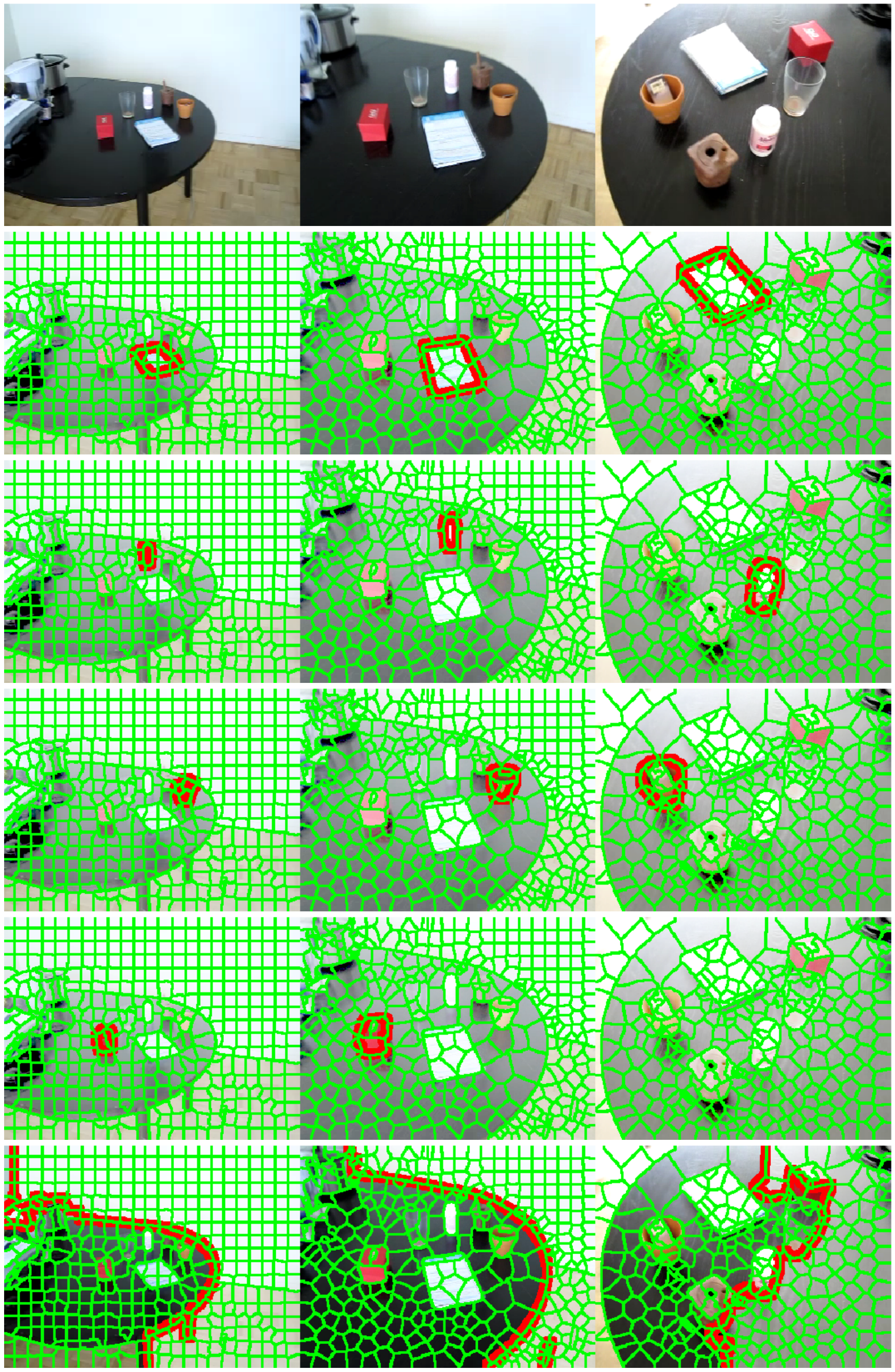


Figure 4.15

for action recognition that require the tracking of parts. In the galloping horse sequence, the horse was recovered well in the middle of the sequence. In the beginning and the end of the sequence only part of the horse is recovered due to poor superpixel boundaries and affinities between the horse and the background, which is also the reason for the incomplete solution in the Pepsi sequence. The horse example also illustrates that our framework works best when given large objects, as small objects usually have higher closure cost and tend to be underseg-mented by superpixels. The table sequence and the flower garden sequence illustrate that our framework can detect most objects in the scene. The results are affected by significant abrupt motion, due to the inability to form good temporal connections in such cases. This can be seen in the hippo sequence. While this sequence illustrates how an additional solution (dog) can be generated by subtracting one solution (hippo) from another (hippo and dog), the dog is only recovered in the first half of the sequence, as it moves abruptly.

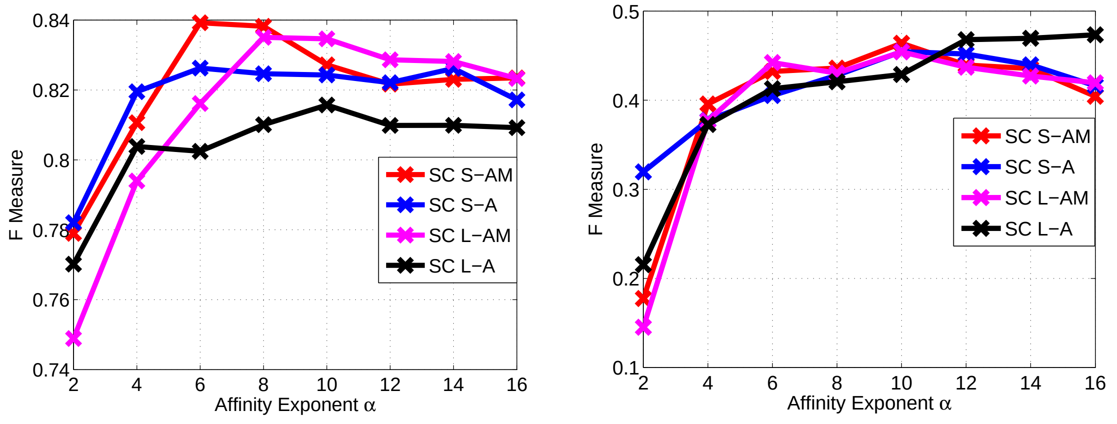
For quantitative evaluation of our method we use 27 sequences from the dataset of Stein et al. [100] (SD) and 20 sequences from the Weizmann action recognition dataset (WD) of Gorelick et al. [34]. Each sequence has a ground truth video segmentation mask, marking one foreground object. Given a set of detected spatiotemporal figures for a sequence, we choose the solution with the maximal F measure relative to the ground truth. For each dataset, we report the average F measure across all sequences.

We compare different variations of our algorithm, as well as replace our parametric maxflow minimization of the unbalanced normalized cuts cost with standard normalized cuts. Unlike our method, normalized cuts requires a user specified number of clusters. Therefore, to compare with our approach we run normalized cuts with 5, 10, 15, 20, and 25 clusters and concatenate all the results. Recall that our previously described graph construction (S-AM) includes only the immediate spatial neighbors and adds the motion affinity W_{ij}^m for spatial edges. We define additional variations over this construction:

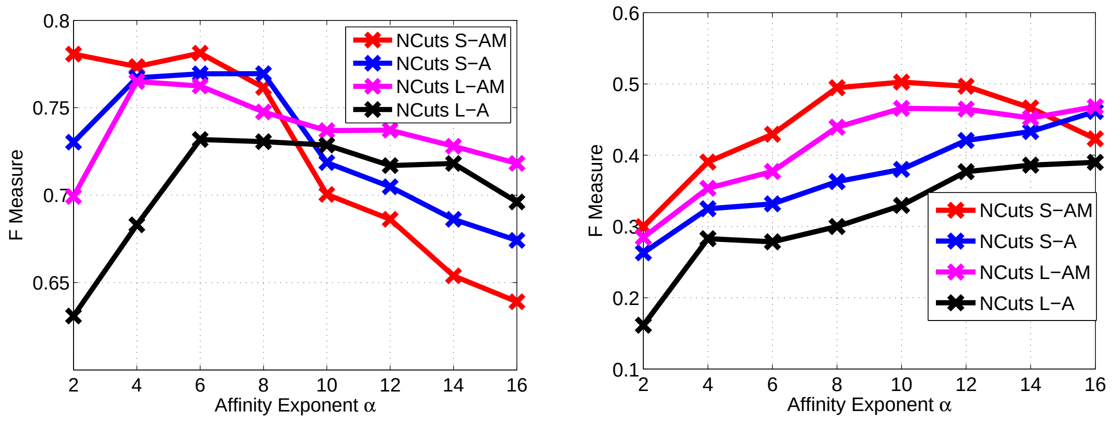
- S-A - Same graph as S-AM, but with affinity only including **appearance**
  $W_{ij} = A_i A_j \left(W_{ij}^a\right)^\alpha$

- L-AM - Same as S-AM but with **larger** spatial connectivity. In addition to the edges in S-AM we add edges between all superpixels in the same frame whose centroids are less than R apart, where R is five times the radius of an average superpixel.

- L-A - Same as L-AM, but with affinity only including appearance $W_{ij} = A_i A_j \left(W_{ij}^a\right)^{\alpha}$



SC



NCuts

Figure 4.16: Quantitative evaluation of spatiotemporal closure detection. Top row shows the results of our method (SC) on SD (left) and WD (right). Bottom row shows the results of running normalized cuts on the same graphs and datasets.

We compare our method (SC) to normalized cuts (NCuts) for all the above graph constructions on both datasets. While we are able to solve the unbalanced normalized cuts problem in a globally optimal fashion, normalized cuts cost is NP-hard to optimize and therefore only an approximation is provided. Despite that, the cut balancing in NCuts further constrains the solutions to be balanced and compact and helps to avoid bleeding, while our closure cost pushes the solutions to contain more superpixels which can result in bleeding. Figure 4.16 illustrates the performance as we vary α. Note that WD provides more challenge for both SC and NCuts as the foreground objects in this dataset are much smaller than in SD. We also observe that our method achieves comparable results using S-AM and L-AM, indicating that our increase of spatial connectivity has only a marginal effect on the results. Finally, we illustrate the tradeoff between the higher bleeding robustness of NCuts vs. the suboptimality of its solution. This is particularly visible in the performance on WD where the objects are small and bleeding is more of an issue, while in SD they are large and it is more important to obtain an accurate result. That is why SC achieves better performance on SD, while NCuts has the advantage on the other dataset.

4.8 Limitations and Future Work

We have presented a closure detection method in 2D images as well as spatiotemporal domains. Relying on compactness allowed us to use closure to group large subsets of superpixels in a purely bottom-up fashion. However, while our closure cost is shown to be effective, it is by no means “correct” in the sense that its minima must correspond to real objects. Since it uses a purely bottom-up closure cue for grouping, it should be thought of as a guideline rather than a rule, and top-down approaches would potentially perform much better. Some limitations of our cost can be seen in performance on non-compact objects, such as a horse or fast-moving objects that are non-compact in time. Moreover, both the image cost and the spatiotemporal cost give preference to objects with larger area. This feature enables us to effectively form very large groups of superpixels bottom-up, but hinders performance for small objects. Finally, our spatiotemporal cost includes only local constraints for temporal coherence. Tracking approaches that maintain a global shape and/or appearance model of the target are expected to do better. That said, both our 2D and spatiotemporal costs could be augmented to include global shape or appearance constraints. It would enable us to handle more global effects, and segment objects in the presence of occlusion, for example. We plan to explore this extension in the future.

Perhaps our most significant contribution is the use of superpixels and redefining closure detection as finding groups of superpixels instead of grouping edgels. This reformulation decreases the complexity of the problem and provides an ideal scope for feature extraction. Unfortunately, errors in superpixel segmentation propagate throughout our framework. Superpixel bleeding prevents the accurate segmentation of figure from ground, but an even larger issue is that it provides incorrect scope for feature extraction, thereby weakening gaps or strengthening affinities between figure and ground. In future work, we plan to explore additional cues for gap and affinity computation to strengthen the robustness of our approach to weak object edges or similar foreground/background motion. For example, we plan to use superpixel junctions to learn an affinity measure between pairs of superpixels that are both inside and adjacent to the boundary. Such an affinity measure can encode a learned measure of continuity and T-junction, and could significantly strengthen our cost function. Ultimately, our goal is to detect closure in cases such as the Kanizsa triangle (Figure 2.2 in Chapter 2) or detect spatiotemporal closures in presence of large occlusions. Currently this is beyond the capabilities of our framework and only objects with a relatively small amount of gap, whether spatial or temporal, can be detected. However, we believe that we can partially approach this goal through the use of more global shape cues in our closure cost and multiscale superpixel segmentations, as coarser superpixels will be more suitable for capturing large gaps.

4.9 Conclusions

Our reformulation of the problem of finding cycles of contours in images as the problem of finding spatially coherent subsets of superpixels, whose collective boundary has strong image edge evidence yields an optimal framework for closure detection that compares favorably with two leading prior approaches. In contrast to competing approaches that focus on the detection of a single, best, closed contour, our optimization framework generates a small number of solutions that can be though of as promising shape hypotheses, better serving high-level recognition tasks. While superpixels provide an ideal scope for learning a gap measure from training data, they offer a number of additional advantages that we are currently exploring. In an extension to the main approach, we show that superpixels also provide a convenient mechanism for incorporating appearance information, if appropriate and if available. For example, if the object was known to be homogeneous in appearance, our modified cost function can easily incorporate such a prior, as discussed in Section 4.7.1. We also provide a crude approach for using multiscale superpixel information, with the plan of pursuing a more elegant coarse-to-fine framework for finding contour closure using multiple superpixel scales. In our last extension, we describe how a slightly modified cost (Eqn. 4.3) can be used for spatiotemporal closure detection. To conclude, our closure detection method efficiently recovers a small number of 2D and spatiotemporal figure/ground hypotheses and opens the way for better solutions of high-level vision problems.

Chapter 5

Multiscale Symmetric Parts Detection and Grouping

5.1 Introduction

The medial axis transform [7] decomposes a closed 2-D shape into a set of skeletal parts and their connections, providing a powerful parts-based decomposition of the shape that's suitable for shape matching [97, 89]. While the medial axis-based research community is both active and diverse, it has not kept pace with the mainstream object recognition (categorization) community that seeks to recognize objects from cluttered scenes. The main reason for this disconnect is the restrictive assumption that the silhouette of an object is available – that the open problem of figure-ground segmentation has somehow been solved. Even if it were possible to segment the figure from the ground, a second source of concern arises around the instability of the resulting skeleton – the skeletal branches often don't map one-to-one to the object's coarse symmetric parts. However, these limitations should in no way deter us from the goal of recovering an object's symmetric part structure from images. We simply need an alternative approach that doesn't assume figure-ground segmentation and doesn't introduce skeletal instability.

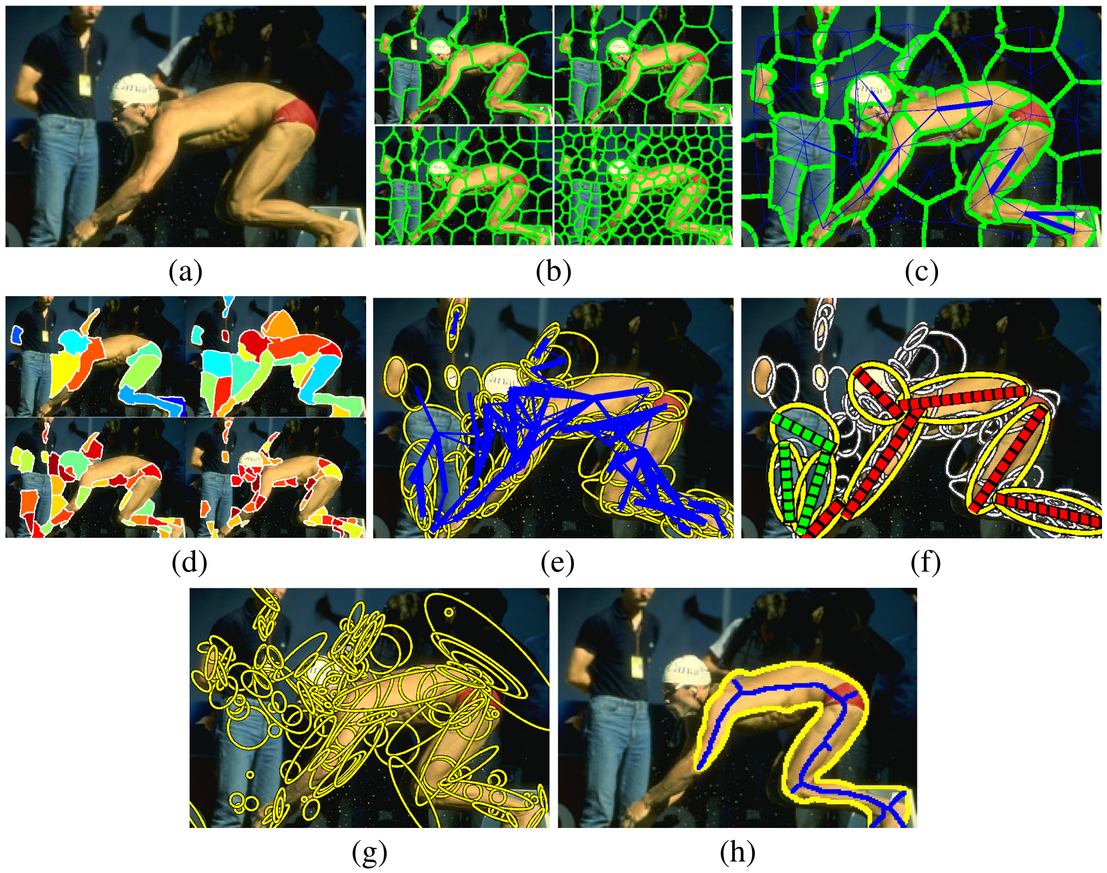


Figure 5.1: Overview of our approach for multiscale symmetric part detection and grouping: (a) original image; (b) set of multiscale superpixel segmentations (different superpixel resolutions); (c) the graph of affinities shown for one scale (superpixel resolution); (d) the set of regularized symmetric parts extracted from all scales through a standard graph-based segmentation algorithm; (e) the graph of affinities between nearby symmetric parts (all scales); (f) the most prominent part clusters extracted from a standard graph-based segmentation algorithm, with abstracted symmetry axes overlaid onto the abstracted parts; (g) in contrast, a Laplacian-based multiscale blob and ridge decomposition, such as that computed by [54], shown, yields many false positive and false negative parts; (h) in contrast, classical skeletonization algorithms require a closed contour which, for real images, must be approximated by a region boundary. In this case, the parameters of the N-cuts algorithm [93] were tuned to give the best region (maximal size without region undersegmentation) for the swimmer. A standard medial axis extraction algorithm applied to the smoothed silhouette produces a skeleton (shown in blue) that contains spurious branches, branch instability, and poor part delineation.

In this chapter, we introduce a novel approach to recovering the symmetric part structure of an object from a cluttered image, as outlined in Fig. 5.1. Drawing on the principle that a skeleton is defined as the locus of *medial points*, i.e., centers of maximally inscribed disks, we first hypothesize a sparse set of medial points at multiple scales by segmenting the image (Fig. 5.1(a)) into compact superpixels at different superpixel resolutions (Fig. 5.1(b)). Superpixels are adequate for this task, balancing a data-driven component that's attracted to shape boundaries while maintaining a high degree of compactness. The superpixels (medial point hypotheses) at each scale are linked into a graph, with edges adjoining adjacent superpixels. Each edge is assigned an affinity that reflects the degree to which two adjacent superpixels represent medial points belonging to the same symmetric part (medial branch) (Fig. 5.1(c)). The affinities are learned from a set of training images whose symmetric parts have been manually identified. A standard graph-based segmentation algorithm applied to each scale yields a set of superpixel clusters which, in turn, yield a set of regularized symmetric parts (Fig. 5.1(d)).

In the second phase of our approach, we address the problem of perceptually grouping symmetric parts arising in the first phase. Like in any grouping problem, our goal is to identify sets of parts that are causally related, i.e., unlikely to co-occur by accident. Again, we adopt a graph-based approach in which the set of symmetric parts across all scales are connected in a graph, with edges adjoining parts in close spatial proximity (Fig. 5.1(e)). Each edge is assigned an affinity, this time reflecting the degree to which two nearby parts are believed to be physically attached. Like in the first phase, the associated, higher granularity affinities are learned from the regularities of attached symmetric parts identified in training data. Consequently, we explore two graph-based methods for grouping the detected parts. The first method is the same greedy approach that was used to cluster superpixels into parts. The second method is motivated by the optimization technique in Chapter 4 and uses parametric maxflow to globally minimize an unbalanced normalized cuts criterion over the part graph. Both methods yield part clusters, each representing a set of regularized symmetric elements and their hypothesized attachments (Fig. 5.1(f)).

Our approach offers clear advantages over competing approaches. For example, classical multiscale blob and ridge detectors, such as [54] (Fig. 5.1(g)), yield many spurious parts, a challenging form of noise for any graph-based indexing or matching strategy. And even if an opportunistic setting of a region segmenter's parameters yields a decent object silhouette (Fig. 5.1(h)), the resulting skeleton may exhibit spurious branches and may fail to clearly delineate the part structure. From a cluttered image, our two-phase approach recovers, abstracts, and groups a set of medial branches into an approximation to an object's skeletal part structure, enabling the application of skeleton-based categorization systems to more realistic imagery.

5.2 Related Work

The use of symmetry as a basis for part extraction has a long history in computer vision, including Blum's medial axis transform (MAT) [7], Binford's generalized cylinders [6], Pentland's superquadric ellipsoids [77], and Biederman's geons [4], to name just a few examples. The literature is vast, and space permits us to highlight only a small subset of approaches that assume a 2-D symmetry-based, part-based shape prior *without* assuming an object prior. Thus, approaches that learn to segment particular categories of objects or scenes, often referred to as *image labeling* or *knowledge-based segmentation*, are excluded for they assume knowledge of object or scene content. Likewise, the rich body of skeletonization literature that assumes that a closed curve is provided is also not reviewed here, for it assumes figure-ground segmentation. We thus review only approaches that attempt to extract and group a set of 2-D symmetric parts from a cluttered image.

The use of symmetry as a basis for multiscale abstract part extraction was proposed by Crowley [24], who detected peaks (rotational symmetries) and ridges (elongated symmetries) as local maxima in a Laplacian pyramid, linked together by spatial overlap to form a tree structure. Object matching was then formulated as comparing paths through two trees. Shokoufandeh et al. [94] proposed a more elaborate matching framework based on Lindeberg's multiscale blob model [54]. This family of approaches can be characterized as imposing a strong part-based symmetry prior, detecting parts at multiple scales, and grouping them based on a simple model of spatial proximity. However, simply detecting parts as local maxima in a set of multiscale filter responses leads to many false positives and false negatives, suggesting that successful part extraction requires paying closer attention to image contours.

Symmetry has long been a foundational non-accidental feature in the perceptual grouping community. Many computational models exist for symmetry-based grouping, including Brady and Asada [10], Cham and Cipolla [17], Saint-Marc et al. [84], Ylä-Jääski and Ade [110] and, more recently, Stahl and Wang [98], among others. Such systems face one or more important limitations: 1) the complexity of pairwise contour grouping to detect symmetry-related contour pairs; 2) the requirements of contour smoothness and precise pointwise correspondence dictated by the geometric emphasis of many such approaches; and 3) that such approaches typically stop short of grouping the detected symmetries (parts) into objects.

Our methodology addresses each of these limitations. On the complexity issue, by adopting a region-based approach, our superpixels (medial point hypotheses) effectively group together nearby contours that enclose a region of homogeneous appearance. Drawing on the concept of extracting blobs at multiple scales, symmetric parts will map to "chains" of medial points sampled at their appropriate scale. Our goal will be to group together the members of such chains, ignoring those superpixels (the vast majority) that don't represent good medial point hypotheses. On the smoothness and precision issue, we will learn from noisy training data the probability that two adjacent superpixels represent medial point approximations that belong to the same symmetric part; this probability forms the basis for our affinity function used to cluster medial points into chains. Finally, on the issue of part grouping, we will also learn from noisy training data the affinity function that will form the basis of part attachment. Addressing these three issues yields a novel framework that aims to narrow the gap between work in the segmentation and medial axis extraction communities.

5.3 Medial Part Detection

The first phase of our algorithm detects medial parts by hypothesizing a sparse set of multiscale medial hypotheses and grouping those that are non-accidentally related. In the following subsections, we detail the two components.

5.3.1 Hypothesizing Medial Points

Medial point hypotheses are generated by compact superpixels which, on one hand, adapt to boundary structure, while on the other hand, enforce a weak compactness shape constraint. In this way, superpixels whose scale is comparable to the width of a part can be seen as deformable maximal disks, "pushing out" toward part boundaries while maintaining compactness. If the superpixels are sampled too finely or too coarsely for a given part, they will not relate together the opposing boundaries of a symmetric part, and represent poor medial point hypotheses. Thus, we generate compact superpixels at a number of resolutions corresponding to the different scales at which we expect parts to occur; as can be seen in Fig. 5.1(b), we segment an image into 25, 50, 100 and 200 superpixels. To generate superpixels at each scale, we employ a modified version [70] of the normalized cuts algorithm [93] since it yields compact superpixels.

Each superpixel segmentation yields a superpixel graph, where nodes represent superpixels and edges represent superpixel adjacencies. If a superpixel represents a good medial point hypothesis, it will extend to (and follow) the opposing boundaries of a symmetric part, effectively coupling the two boundaries through two key forms of perceptual grouping: 1) *continuity*, where the intervening region must be locally homogeneous in appearance; and 2) *symmetry*, in that the notion of maximal disk bitangency translates to two opposing sections of a superpixel's boundary. Fig. 5.2(b) illustrates a symmetry section (blow-up of the subimage in Fig. 5.2(a) containing the athlete's leg) whose medial point hypotheses are too large (undersampled), while in Fig. 5.2(c), the medial point hypotheses are too small (oversampled). When they are correctly sampled, as in Fig. 5.2(d), they can be viewed as a sparse approximation to the locus of medial points making up a skeletal branch, as seen in Fig. 5.2(e).

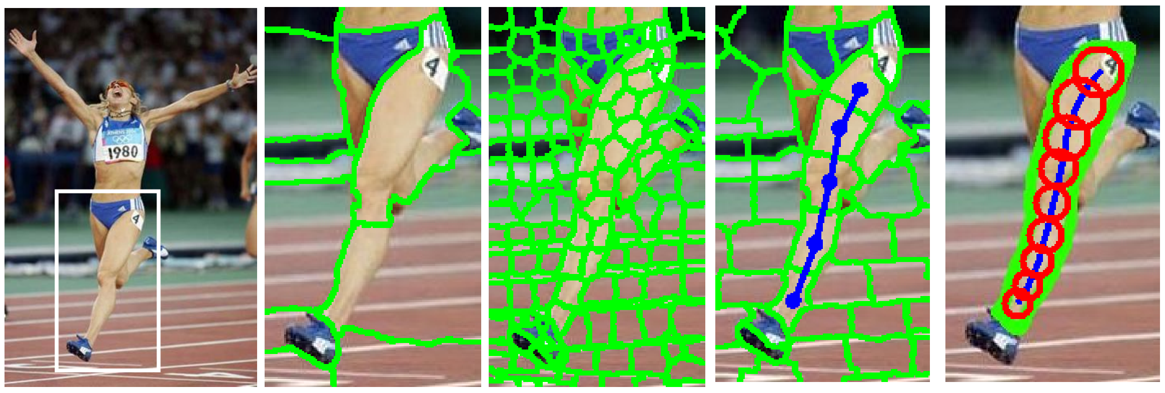


Figure 5.2: Superpixels as medial point samples: (a) a region of interest focusing on the athlete's leg (b) superpixels undersample the scale of the symmetric part; (c) superpixels oversample the scale of the symmetric part; (d) superpixels appropriately sample the scale of the symmetric part, non-accidentally relating, through continuity and symmetry, the two opposing contours of the part; (e) the medial point hypotheses that effectively capture the scale of the part represent a sparse approximation to the locus of medial points that comprise the traditional skeleton.

5.3.2 Clustering Medial Points

If two adjacent superpixels represent two medial points belonging to the same symmetric section, they can be combined to extend the symmetry. This is the basis for defining the edge weights in the superpixel graph corresponding to each resolution. Specifically, the affinity between two adjacent superpixels represents the probability that their corresponding medial point hypotheses not only capture non-accidental relations between the two boundaries, but that they represent medial points that belong to the same skeletal branch. Given these affinities, a standard graph-based clustering algorithm applied independently to each scale yields clusters of medial points, each representing a medial branch at that scale. In Section 5.4, we group nonaccidentally related medial branches by object, yielding an approximation to an object's skeletal part structure.

The affinity $A_s(i,j)$ between two adjacent superpixels R_i and R_j at a given scale has both shape A_{shape} and appearance components $A_{appearance}$. We learn the components and how to combine them from training data. To generate training examples, we segment an image into superpixels at multiple scales, and identify adjacent superpixels that represent medial points that belong to the same medial branch as positive evidence; negative pairs are samples in which one or both medial point hypotheses are incorrect or, if both are valid medial points, belong to different but adjacent parts. The boundary of the union of each superpixel pair defines a hypothesized boundary in the image (which may or may not have local gradient support).

To compute the shape-based affinity, we fit an ellipse to the union of two adjacent superpixels (we find an ellipse with the same second moments as the superpixel union region). We assign an edge strength to each boundary pixel equal to its Pb score [63] in the original image. Each boundary pixel is mapped to a normalized coordinate system by projecting its coordinates onto the major and minor axes of the fitted ellipse, yielding a scale- and orientation-invariant representation of the region boundary. We split our normalized coodinate system into rectangular bins of size $0.3a \times 0.3b$, where a and b are half the length of the major and minor ellipse axes, respectively. Using these bins, we compute a 2-D histogram on the normalized boundary coordinates weighted by the edge strength of each boundary pixel. Focusing on the superpixel pair union and its local neighborhood, we only consider bins in the range $[-1.5a, 1.5a]$ and $[-1.5b, 1.5b]$, resulting in a 10×10 histogram. This yields a shape context-like feature that reflects the distribution of edges along the presumed boundary of adjacent superpixels. Fig. 5.3 illustrates the shape feature computed for the superpixel pair from Fig. 5.1(c), corresponding to the thigh of the swimmer.

We train a classifier on this 100-dimensional feature using our manually labeled superpixel pairs. The margin from the classifier (an SVM with RBF kernel) is fed into a logistic regressor

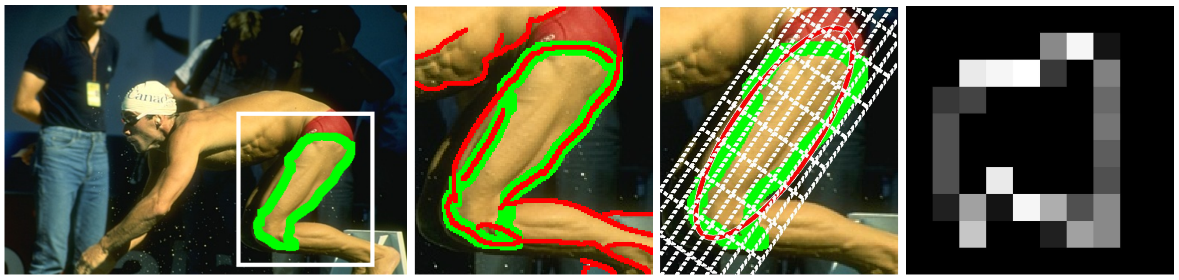


Figure 5.3: Superpixel shape feature: (a) boundary of two adjacent superpixels representing two medial point hypotheses; (b) a blow-up of the two superpixels, in which the boundary of their union (green) defines a section of a hypothesized symmetric part which may or may not have underlying image edge support (red); (c) the normalized scale- and orientation-invariant coordinate system (grid in white) based on the ellipse (red) fitted to the superpixel union; (d) the shape-context-like feature that projects image edgels, weighted by edge strength, into this coordinate system.

in order to obtain the shape affinity $A_{shape}(R_1, R_2)$ whose range is $[0, 1]$. Table 5.1 compares various approaches for computing the shape affinity; the SVM with RBF kernel and the SVM with a histogram intersection kernel yield the highest performance.

For the appearance component of the affinity, we compute the absolute difference in mean RGB color, absolute difference in mean HSV color, RGB and HSV color variances of both regions, and histogram distance in HSV space, yielding a 27-dimensional appearance feature. To improve classification, we compute quadratic kernel features, resulting in a 406-dimensional appearance feature. We train a logistic regressor with L1-regularization to prevent overfitting on our relatively small training dataset while emphasizing the weights of the more important features. This yields an appearance affinity measure between two regions ($A_{appearance}(R_1, R_2)$). Training the appearance affinity is easier than training the shape affinity. For positive examples, we choose pairs of adjacent superpixels that are contained inside a figure in the figure-ground segmentation, whereas for negative examples, we choose pairs of

	SVM-R	SVM-H	CC	HI
$F_{measure}$	0.75	0.75	0.42	0.44
Mean Precision	0.79	0.79	0.29	0.31

Table 5.1: Shape affinity comparison according to two measures: $F_{measure}$ and mean precision evaluated on test pairs of superpixels. We evaluate 4 methods: SVM with RBF kernel (SVM-R), SVM with histogram intersection kernel (SVM-H), as well as cross correlation (CC) and histogram intersection (HI) against a mean histogram of all positive training pairs.

adjacent superpixels that span figure-ground boundaries.

We combine the shape and appearance affinities using a logistic regressor to obtain the final pairwise region affinity $A_s(i, j)$. The resulting graph is used in conjunction with an efficient agglomerative clustering algorithm based on [31] (complexity: $O(|S|)$, where S is a set of all superpixels) to obtain medial parts (medial point clusters). As the algorithm relies on having edge weights that measure the dissimilarity between pairs of elements, we first convert the superpixel affinity to edge weights as $W(i, j) = \frac{1}{A_s(i,j)}$. The clustering algorithm initializes all medial point hypotheses as singletons, and maintains a global priority queue of edges by increasing edge weight (decreasing affinity A_s). At each iteration, the edge with the lowest weight (highest affinity) is removed from the queue, and the two clusters that span the edge are hypothesized as belonging to the same part. If each of the two clusters is a singleton, these are merged if the affinity is sufficiently high (the affinity captures the degree to which the union is symmetric). If one or both clusters contain multiple medial points (superpixels), the global symmetry A_s of the union is verified (in the same manner as a pair is verified, i.e., based on the same shape feature built over the union and the same logistic regressor) before the merge is accepted. Thus, while local affinities define the order in which parts are grown, more global information on part symmetry actually governs their growth. The result is a set of parts from each scale, where each part defines a set of medial points (superpixels). Combining the parts from all scales, we obtain the set $Part_1, Part_2, \ldots, Part_n$. Fig. 5.1(d) shows the parts extracted at four scales.

5.4 Assembling the Medial Parts

Medial part detection yields a set of skeletal branches at different scales. The goal of grouping is to assemble the medial branches that belong to the same object. Drawing on the non-accidental relation of proximity, we define a single graph over the union of elements computed at all scales, with nodes representing medial parts and edges linking pairs in close proximity. Assigned to each edge will be an affinity that reflects the likelihood that the two nearby parts are not only part of the same object, but attached. Two different graph-based clustering techniques are then explored to detect part clusters. However, since some parts may be redundant across scales, a final selection step is applied to yield the final cluster of medial branches, representing an approximation to the object's skeletal part structure. The following two subsections describe the two steps.

5.4.1 Medial Part Clustering

A minimal requirement for clustering two parts is their close proximity. While the projections of two attached parts in 3-D must be adjacent in 2-D (if both are visible), the converse is not necessarily true, i.e., adjacency in 2-D does not necessarily imply attachment in 3-D (e.g., occlusion). Still, the space of possible part attachments can be first pruned to those that *may* be attached in 3-D. Two parts are hypothesized as attached if one overlaps a scale-invariant dilation of the other (the part is dilated by the size of the minor axis of the ellipse fitted to it, in our implementation).

The edges in the graph can be seen as weak local attachment hypotheses. We seek edge affinities that better reflect the probability of real attachments. We learn the affinity function from training data – in this case, a set of ground truth parts and their attachments, labeled in an image training set. For each training image, we detect parts at multiple scales, hypothesize connections (i.e., form the graph), and map detected parts into the image of ground truth parts, retaining those parts that have good overlap with ground truth. Positive training example pairs consist of two adjacent detected parts (joined by an edge in the graph) that map to attached parts in the ground truth. Negative training example pairs consist of two adjacent detected parts that map to non-attached (while still possibly adjacent in 2-D) parts in the ground truth.

As mentioned earlier, our multiscale part detection algorithm may yield redundant parts, detected at different scales, but covering the same object entity. One solution would be to assign low affinities between such parts. However, this would mean that only one part in a redundant set could be added to any given cluster, making the cluster more sensitive to noisy part affinities. The decision as to which part in a redundant set survives in a cluster is an important one that is best made in the context of the entire cluster. Therefore, we assign a high affinity between redundant parts, and deal with the issue in a separate part selection step.

Formally, our part affinity is defined as:

$$A_p(i,j) = P_r(i,j) + (1 - P_r(i,j))A_{p,\neg r}(i,j) \tag{5.1}$$

where $P_r(i,j)$ is the probability that parts i and j are redundant, and $A_{p,\neg r}(i,j)$ is the affinity between the parts given non-redundancy. $P_r(i,j)$ is computed by training a quadratic logistic classifier over a number of features, including overlap (in area) of the two parts (O_{ij}), defined as the overlap area normalized by the area of the smaller part, overlap of the two parts' boundaries (B_{ij}), and appearance similarity (A_{ij}) of the two parts. The features are defined as follows:

$$\begin{aligned} O_{ij} &= \frac{|Part_i \cap Part_j|}{\min\{|Part_i|, |Part_j|\}} \\ B_{ij} &= \frac{|\partial(Part_i \cap Part_j)|}{\min\{|\partial Part_i|, |\partial Part_j|\}} \\ A_{ij} &= A_{appearance}(Part_i, Part_j) \end{aligned} \tag{5.2}$$

where $|\cdot|$ is the region area and $|\partial(\cdot)|$ is the region perimeter.

The affinity $A_{p,\neg r}(i,j)$ between non-redundant parts i and j, like affinities between medial points, includes both shape and appearance components. The components are best analyzed based on how the two parts are attached. Given an elliptical approximation to each part, we first compute the intersection of their major axes. The location is normalized by half of the length of the major axis, to yield a scale-invariant attachment position r for each part. We define three qualitative attachment "regions" to distinguish between four attachment types: inside ($|r| < 0.5$) , endpoint ($0.5 < |r| < 1.5$), or outside ($|r| > 1.5$). Our four apparent attachment categories can be specified as follows:

1. end-to-end ($J_{ij} = 1$) – The intersection lies in the endpoint region of both parts.
2. end-to-side ($J_{ij} = 2$) – The intersection lies in the inside region of one part and in the endpoint region of the other part.
3. crossing ($J_{ij} = 3$) – The intersection lies in the inside region of both parts.
4. non-attached ($J_{ij} = 4$) – The intersection lies in the outside region of one or both parts.

Fig. 5.4 gives examples of these four attachment types.

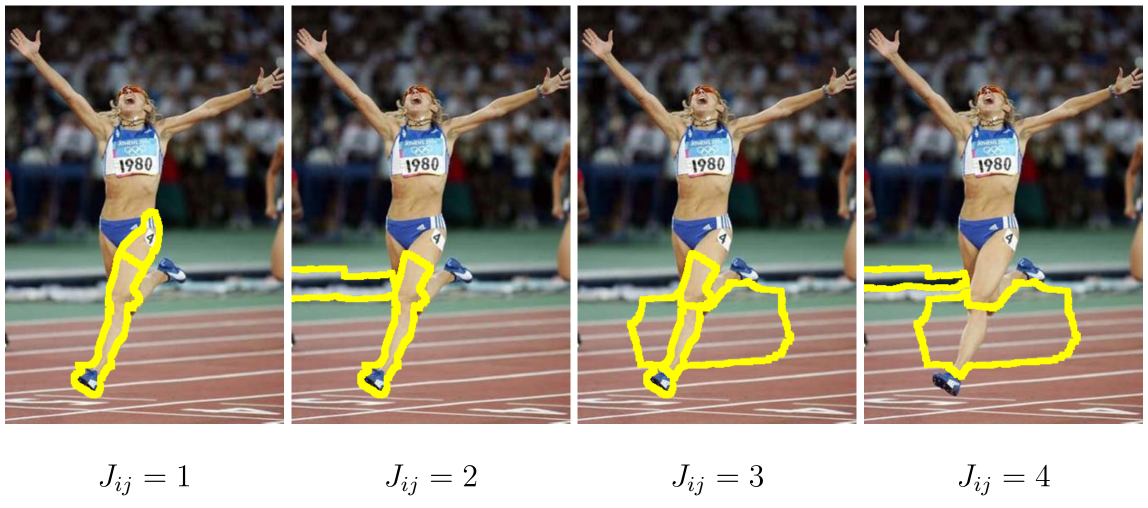


Figure 5.4: Attachment categories. The four different attachment categories of parts (yellow).

The shape component of our affinity is based on the principle that when one part attaches to (interpenetrates) another, it introduces a pair of concave discontinuities (Hoffman and

Richards' principle of transversality [39]), reflected as a pair of L-junctions marking the attachment. In contrast, when one part occludes another, the L-junctions are replaced by T-junctions, reflecting an occlusion boundary. This is a heuristic, for there could be an appearance boundary between two attached parts, misinterpreted as an occlusion boundary.

Since extracting and classifying contour junctions is challenging in the presence of noise, we will instead focus on the evidence of an occlusion boundary between two parts, based on image edges (E_{ij}) along the attachment boundary between parts i and j. Once the attachment boundary is found, evidence is accumulated as the average Pb [63] of the boundary pixels. Finding the attachment boundary is not trivial since the parts may be sufficiently close but not touching, due to segmentation errors.

The attachment boundary is computed similarly for all four attachment categories. For a pair of attached parts, we first select the part P_1 with the smaller $|r|$ and find the intersection of its boundary with the major axis of the other part P_2. The attachment boundary is centered at the intersection and extends along the boundary of P_1 in both directions, to an extent equal to the length of the minor axis (width) of P_2. For end-to-side attachments, this is illustrated in Fig. 5.5.

Given the attachment category J_{ij}, the attachment boundary evidence E_{ij}, and the appearance similarity A_{ij}, we can define the part affinity $A_{p,\neg r}(i,j)$. One logistic classifier is trained for end-to-end junctions ($A_1(i,j)$), whereas another is trained for end-to-side junctions ($A_2(i,j)$). For crossing and non-attached junctions, we set the affinity to 0 because we empirically found that none of the attached part pairs in the training set exhibited such attachment categories. Our affinity for non-redundant parts becomes:

$$\begin{aligned} A_{p,\neg r}(i,j) = [J_{ij} = 1] \cdot A_1(i,j) \\ + [J_{ij} = 2] \cdot A_2(i,j) \end{aligned} \tag{5.3}$$

Having defined all the components of the affinity function $A_p(i,j)$ (Equation 5.1), we use these affinities to cluster parts that are attached.

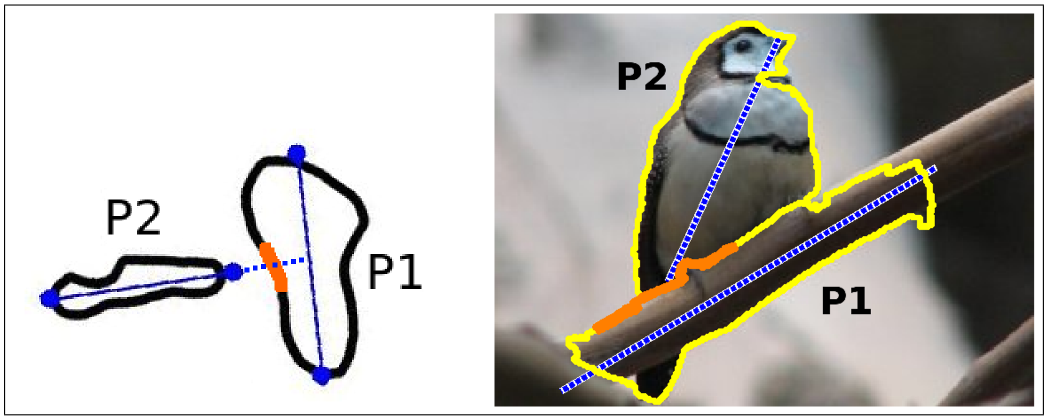


Figure 5.5: Locating the attachment boundary between two parts in the case of an end-to-side attachment. The attachment boundary (orange) between the two parts P_1 and P_2 is centered at the intersection of the major axis of P_2 with the boundary of P_1, and extends along the boundary of P_1 a total distance equal to the length of the minor axis of P_2. Left - illustration of the attachment boundary. Right - attachment boundary between two parts in a real image.

We explore two graph-based approaches for part clustering. Our first approach is the same algorithm [31] used to cluster medial points into parts. Since this technique is greedy in nature, it is susceptible to undersegmentation given noisy part affinities. We therefore explore a second, globally optimal, technique that is based on the work in Chapter 4. Given the goal of finding a well-separated part cluster, we formulate the part clustering problem as finding an optimal unbalanced normalized cut (UNC), which is a measure of cluster dissimilarity to the rest of the graph relative to its internal similarity. Formally, given the part affinities $A_p(i,j)$, $D_i = \sum_j A_p(i,j)$, and a binary indicator vector $\vec{X}$ over parts,

$$UNC(\vec{X}) = \frac{cut(\vec{X})}{volume(\vec{X})} = \frac{\sum_{ij} X_i(1-X_j)A_p(i,j)}{\sum_i X_i D_i}, \tag{5.4}$$

where $cut(\vec{X})$ is the sum of the affinities of all the edges between selected and unselected parts, and $volume(\vec{X})$ is the sum of all the affinities originating from the selected parts. This cost can be globally minimized using parametric maxflow [50], returning multiple solutions with minimal cost under increasing volume constraints. As it stands, however, the cost has a trivial minimizer $\vec{X} = 1$ that selects all the parts. To avoid this trivial solution, we modify the cost and add a small fixed penalty α_p for adding parts to the numerator. Moreover, note that our affinities $A_p(i,j)$ measure part attachment and not similarity in a clustering sense. Two parts in the graph are similar and should be clustered together if they are attached to the same object, meaning that there is a high affinity attachment path between them. To that end, we first compute a shortest path distance $D_p(i,j)$ for all pairs of parts based on their attachment affinities, and convert it into part similarity $W_p(i,j) = e^{-\frac{D_p(i,j)}{\sigma_p}}$. Letting $D'_i = \sum_j W_p(i,j)$, our final unbalanced Ncuts cost becomes:

$$UNC(\vec{X}) = \frac{cut(\vec{X}) + penalty(\vec{X})}{volume(\vec{X})} = \frac{\sum_{ij} X_i(1-X_j)W_p(i,j) + \alpha_p \sum_i X_i}{\sum_i X_i D'_i}. \tag{5.5}$$

In the results section, we will compare the two part clustering approaches, showing that the second method achieves a slightly better performance.

5.4.2 Medial Part Selection

Our affinity-based grouping yields a set of part clusters, each presumed to correspond to a set of attached parts belonging to a single object. However, any given cluster may contain one or more redundant parts. While such parts clearly belong to the same object, we prune redundancies to produce the final approximation to an object's skeletal part structure. Our objective function selects a minimal number of parts from each cluster that cover the largest amount of image, while at the same time minimizing overlap between the parts. The problem is formulated as minimizing a quadratic energy over binary variables. Let $X_i \in \{0,1\}$ be an indicator variable representing the presence of the i^{th} part in a cluster. We seek the subset of parts that minimizes the following energy:

$$E = \sum_i X_i \left(K - |Part_i|\right) + \sum_{i,j} X_i X_j O_{ij} \tag{5.6}$$

where K controls the penalty of adding parts. In our experiments, we found that $K = 0.1 \cdot median\left\{|Part_i|\right\}$ is an effective setting for this parameter. We find the optimal X by solving a relaxed quadratic programming problem, in which real values are rounded to 0 or 1 [78].

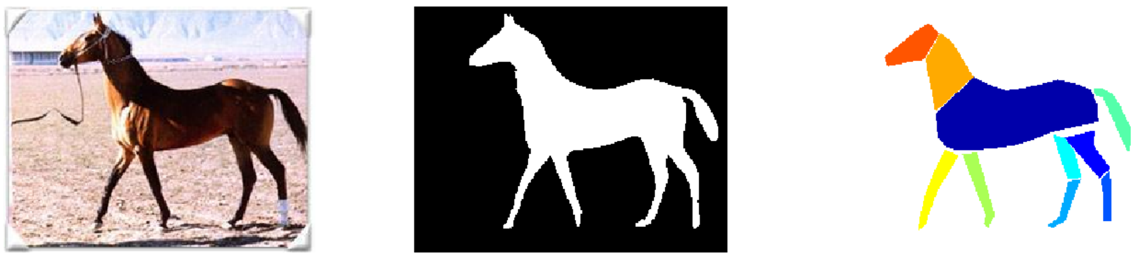


Figure 5.6: Ground truth used for training: sample image (left), figure/ground segmentation (middle), and part segmentation (right).

5.5 Results

To evaluate the method, we train the various components using the Weizmann Horse Database [8], consisting of images of horses together with figure-ground segmentations; in addition, we manually mark the elongated parts of the horses, together with their attachment relations. Fig. 5.6 illustrates an example training image and its ground truth segmentations. Once trained, we first qualitatively evaluate the system on images of objects with well-defined symmetric parts drawn from *different* (i.e., non-horse) image domains, reflecting our assumption that both symmetry and part attachment are highly generic.

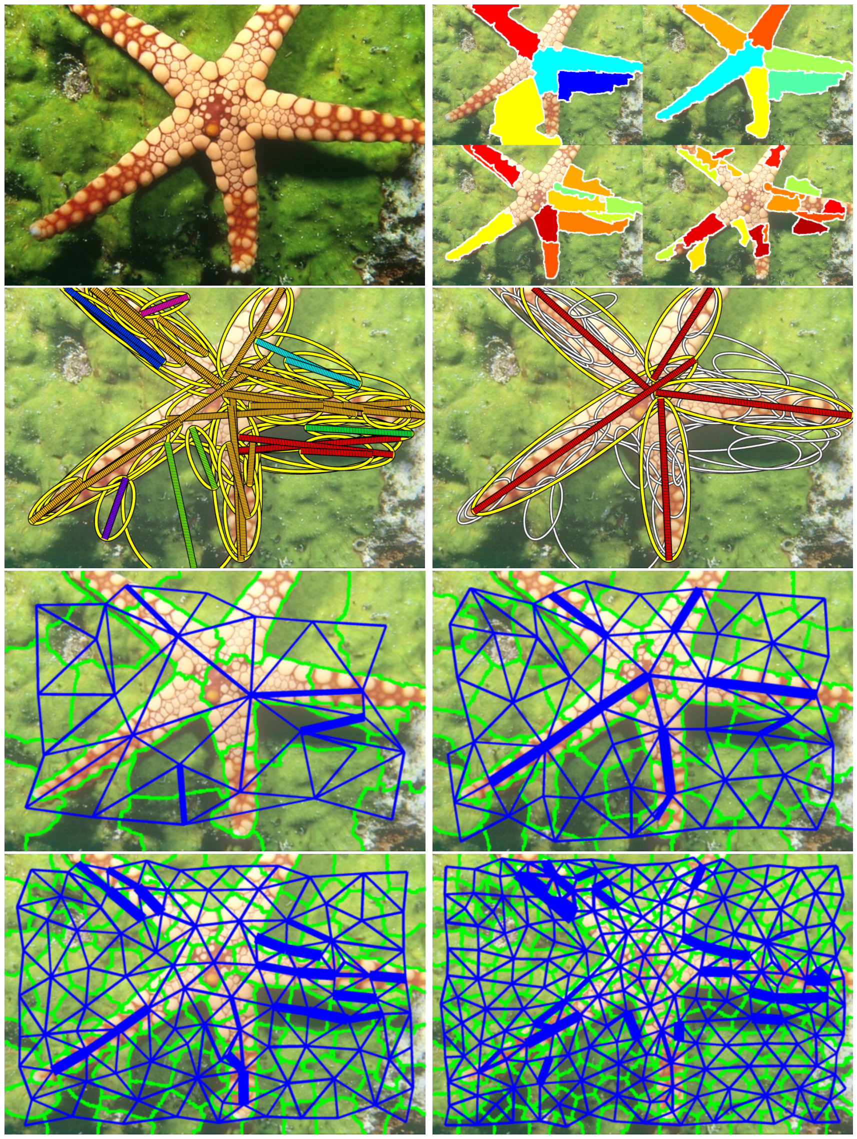
Fig. 5.7 shows qualitative results of our algorithm applied to a number of different image domains[1]. For all the results in this figure, the greedy technique [31] was used for part clustering. In each case (a-m), the figure shows the input image, the most prominent groupings of medial branches, as well as the results of intermediate stages in our system. For part cluster visualization, in order to avoid clutter, the abstractions of the parts in each cluster are shown as ellipses with their major axes (medial branch regularizations) depicted by dotted lines. All other parts are shown with faint (grey) elliptical part abstractions (without axes, for clarity), illustrating the ability of our algorithm to correctly group medial branches.

We organize the results in decreasing order of success, with Figs. 5.7(a-f) corresponding

[1]Supplementary material (`http://www.cs.toronto.edu/~babalex/symmetry_supplementary.tgz`) contains additional examples.

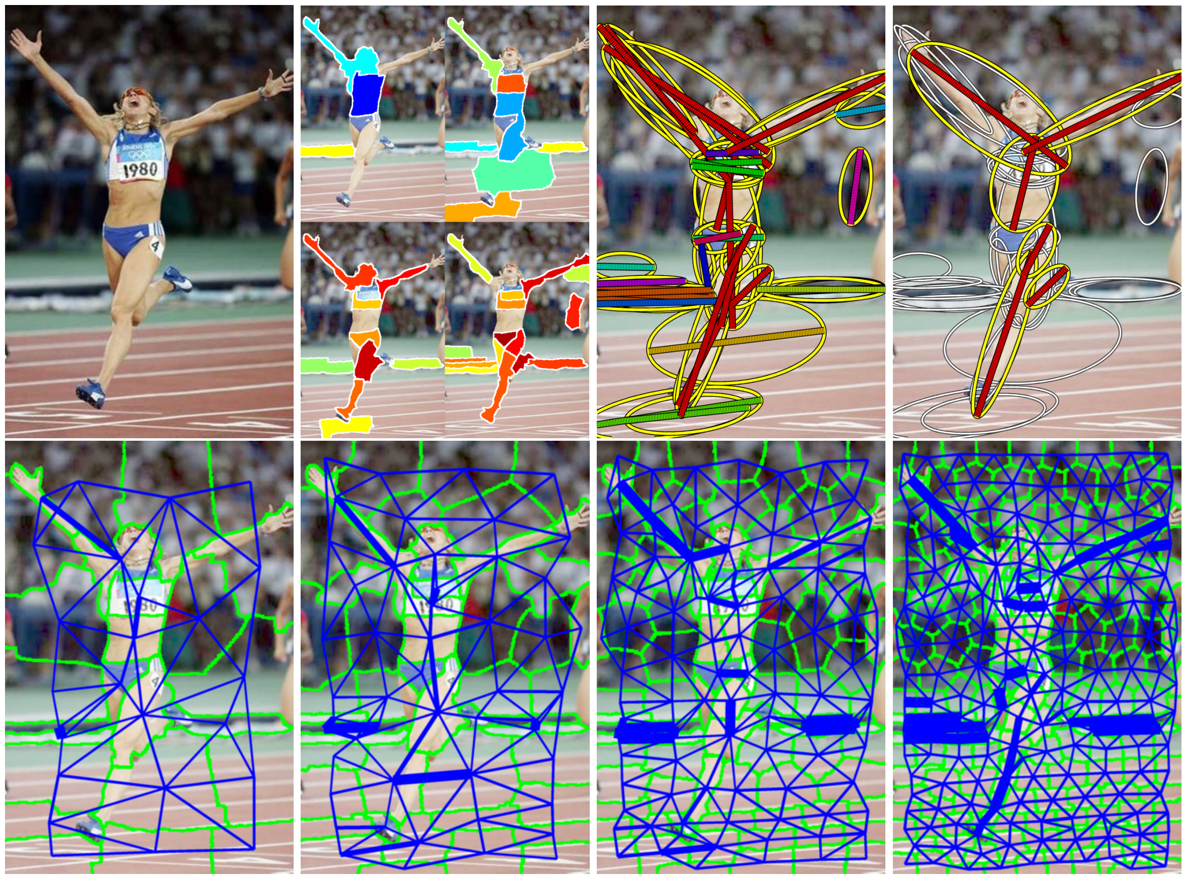
to more successful applications of our system and Figs. 5.7(g-m) illustrating some constraints and failure modes. Examining the results, Fig. 5.7(a) presents an ideal example of our system's operation. All the tentacles of the starfish were successfully recovered and grouped, with the small exception of the center being a part of one of the tentacles. This is a perhaps an easy example since the tentacles are of the same scale, exhibit strong appearance homogeneity and similarity, and are contrasted from the background. Thus, paying closer attention to the superpixels and the affinities, we see that the second scale not only provides perfect medial disc approximations for all the parts, but the affinities between the superpixels of each tentacle are strong. We see that in Fig. 5.7(b), our system has successfully extracted the major parts of the athlete, including the torso, which exhibits heterogeneous appearance, and correctly grouped them together. Fig. 5.7(c) illustrates not only that the parts of the windmill were successfully recovered and clustered, but that the person was also recovered as a separate single-part cluster. The smaller windmills undetected in the background contain parts whose scale was smaller than our finest sampled scale. Figs. 5.7(d-f) show other examples of our system's success, in which the major medial parts of a plane, swan, and statue, respectively, were recovered and grouped to yield an approximation to an object's skeletal part structure.

Figs. 5.7(g-m) illustrate some limitations of our approach. Our framework relies on the assumption that good medial disc approximations could be extracted bottom-up, and our greatest failure mode occurs when this assumption is broken. While Fig. 5.7(g) shows that the bull is correctly detected as a symmetric object, despite irregularities in its contour, the legs of the bull, as well as the arms and legs of the human, are not detected due to insufficient superpixel resolution or low contrast on part boundaries. These are the two main reasons for failing to extract good medial disc approximations and these failures recur in many of the following test cases. For example, in Fig. 5.7(h), one of the swan's wings is not properly detected. Due to insufficient contrast between the wing and the background, the superpixel boundaries fail to capture the part at any scale. Still, the remaining part structure of the swan may provide a sufficiently powerful shape index to select a small set of candidate models, including the swan

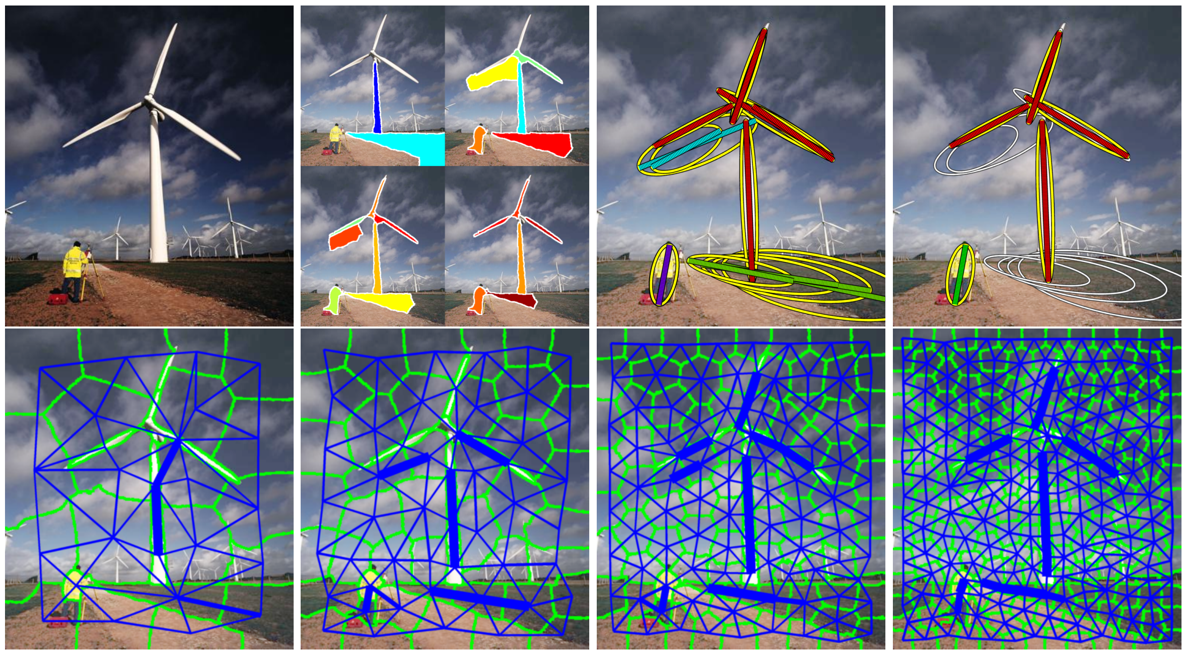


(a)

Figure 5.7: Detected medial parts and their clusters. For each test case, we show the original image in true color, followed by the output of different stages overlaid on brightened images for better contrast. The results (ordered left-to-right and top-to-bottom) illustrate the recovered parts from each superpixel scale, part clusters with axis color indicating cluster membership, and the most prominent part clusters after the part selection stage (yellow ellipses correspond to selected parts, while others correspond to either unselected parts or parts from other clusters). The bottom half of the results shows the extracted superpixels and their affinities at our 4 scales.

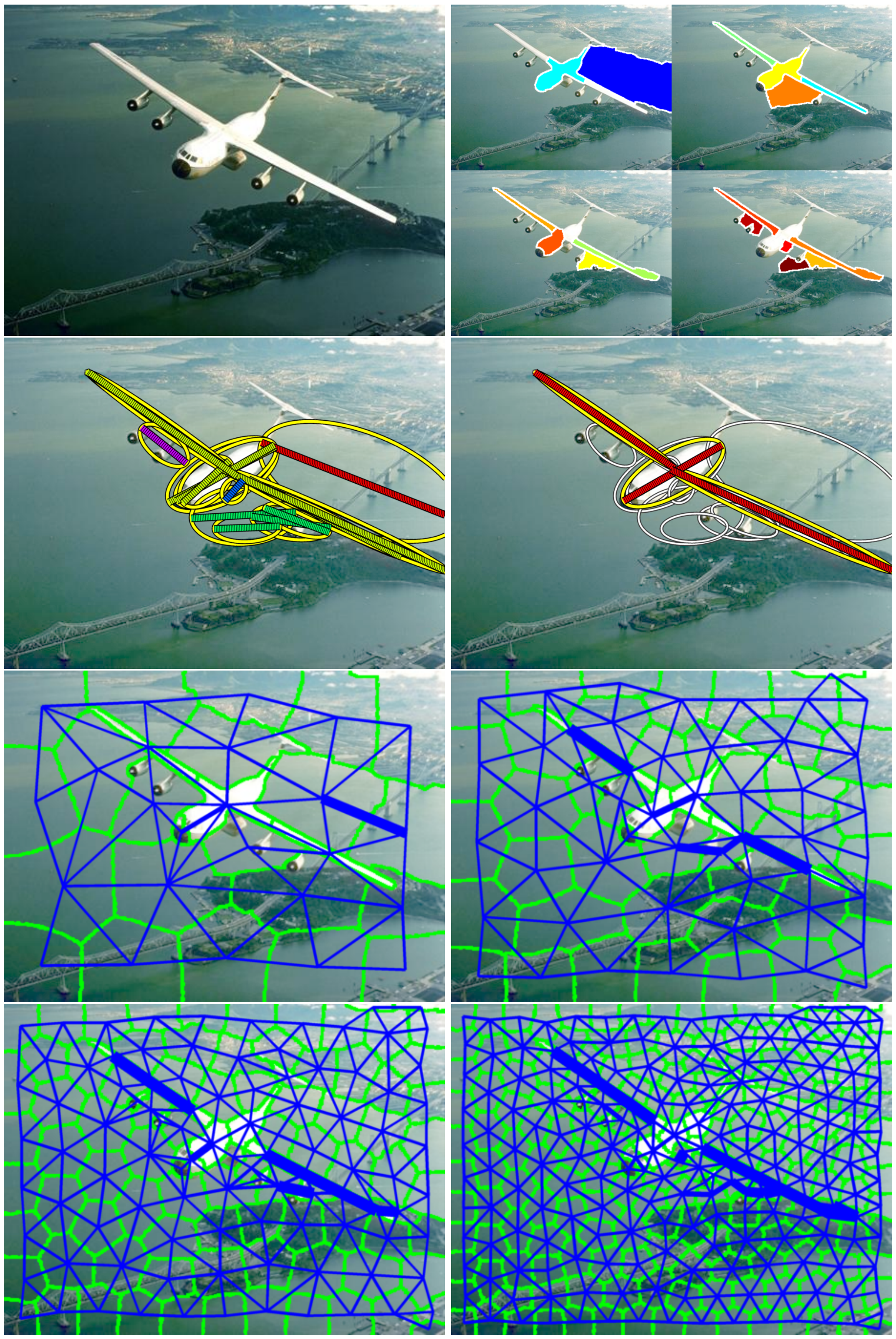


(b)



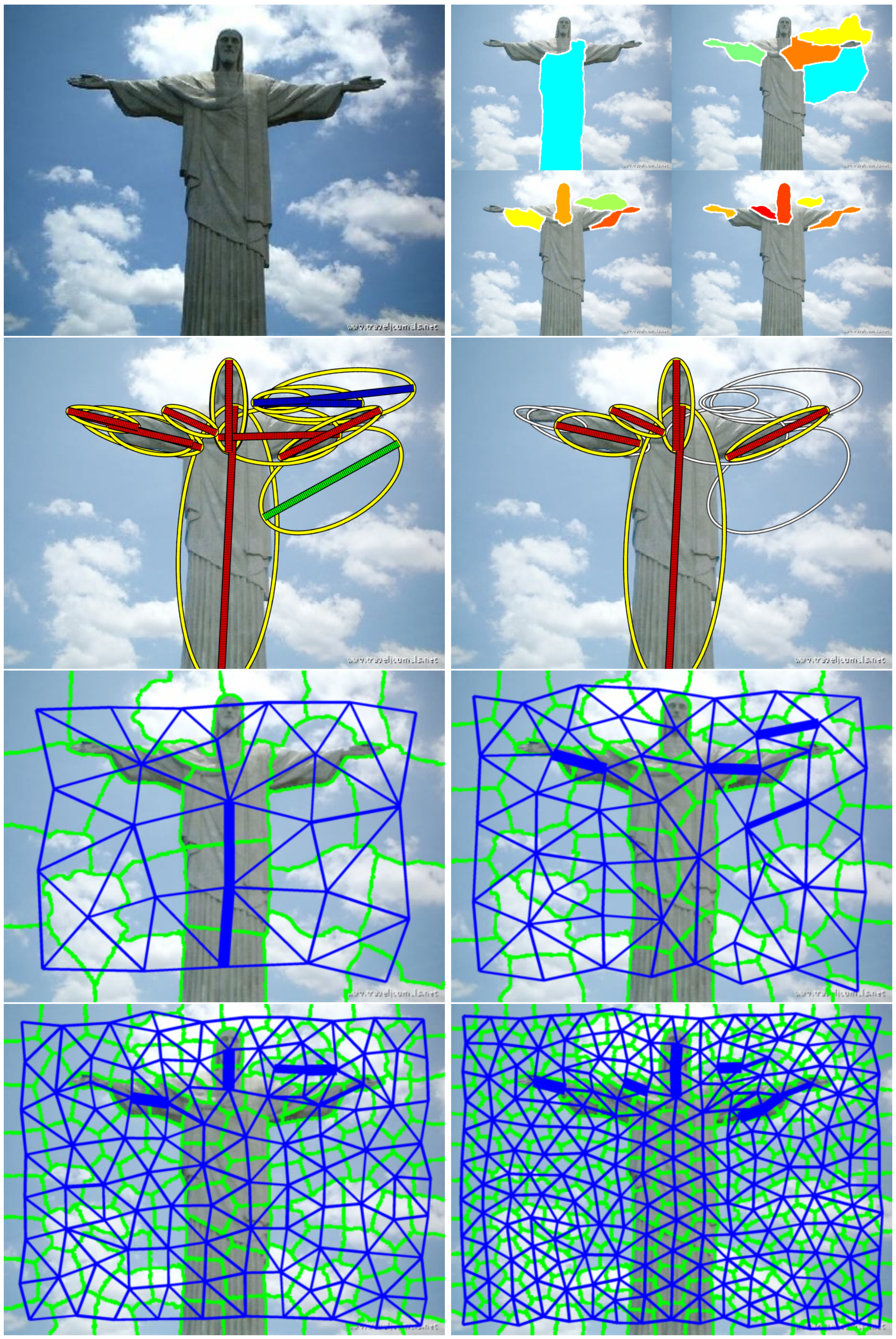
(c)

Figure 5.7



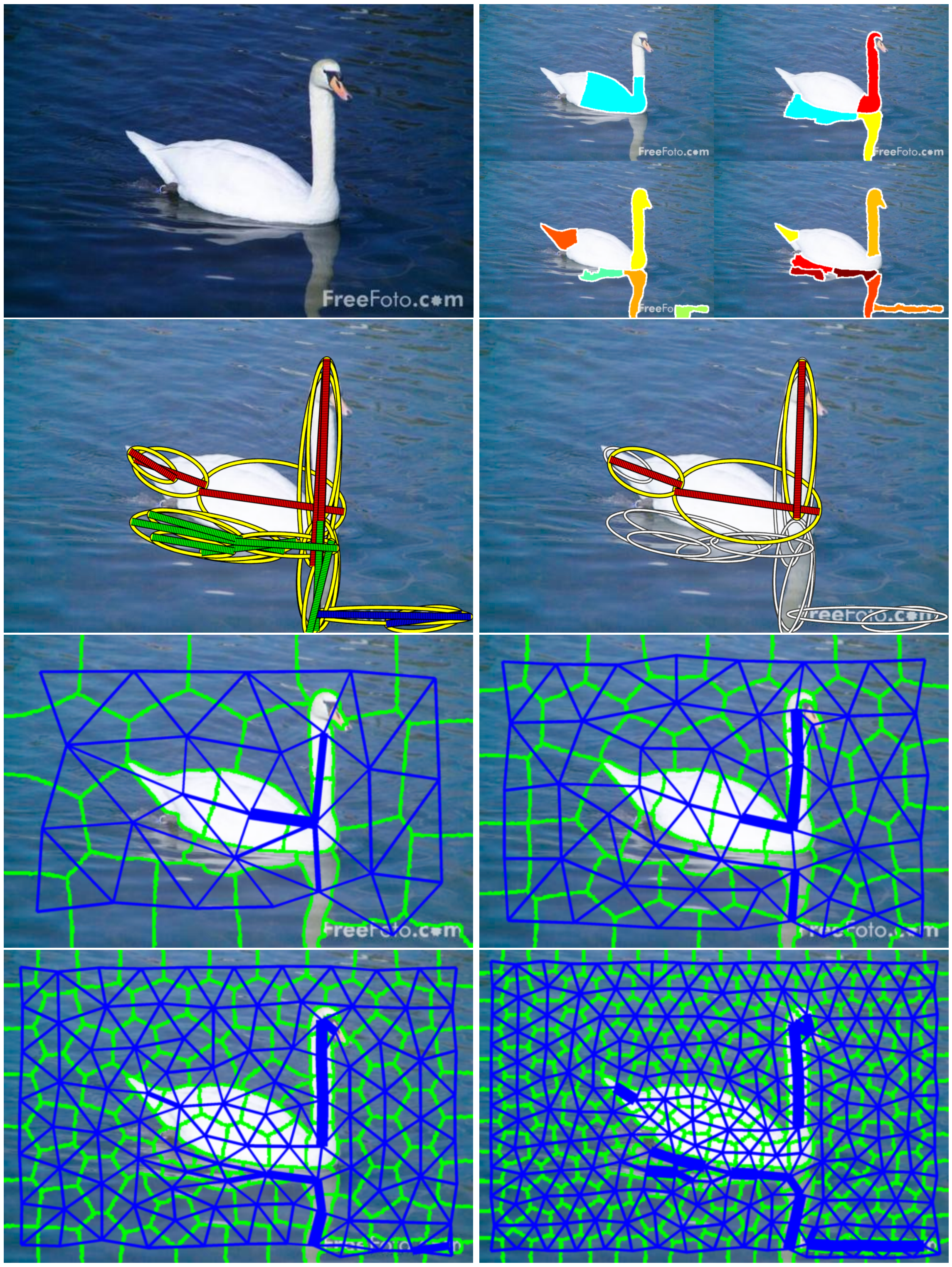
(d)

Figure 5.7



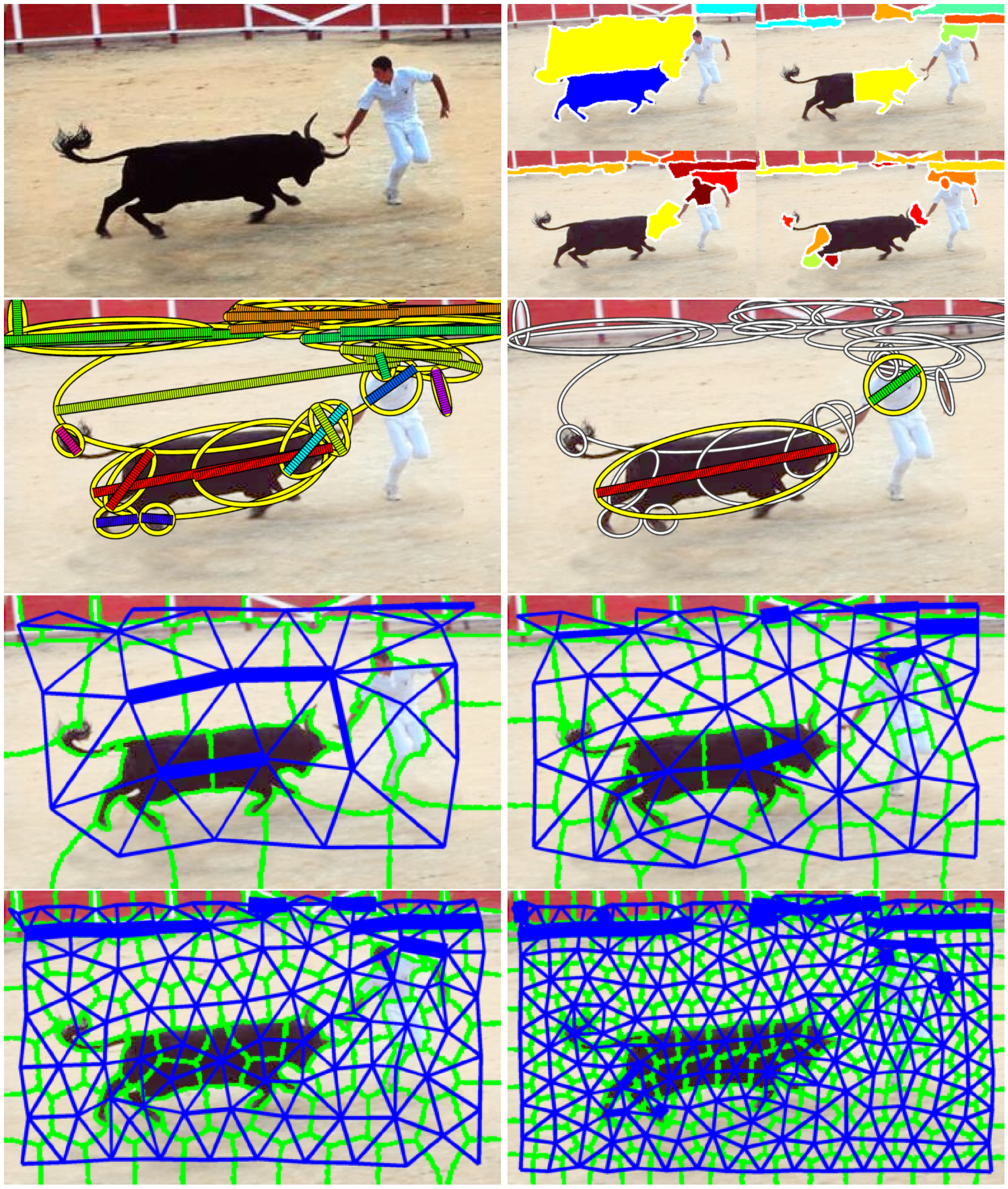
(e)

Figure 5.7



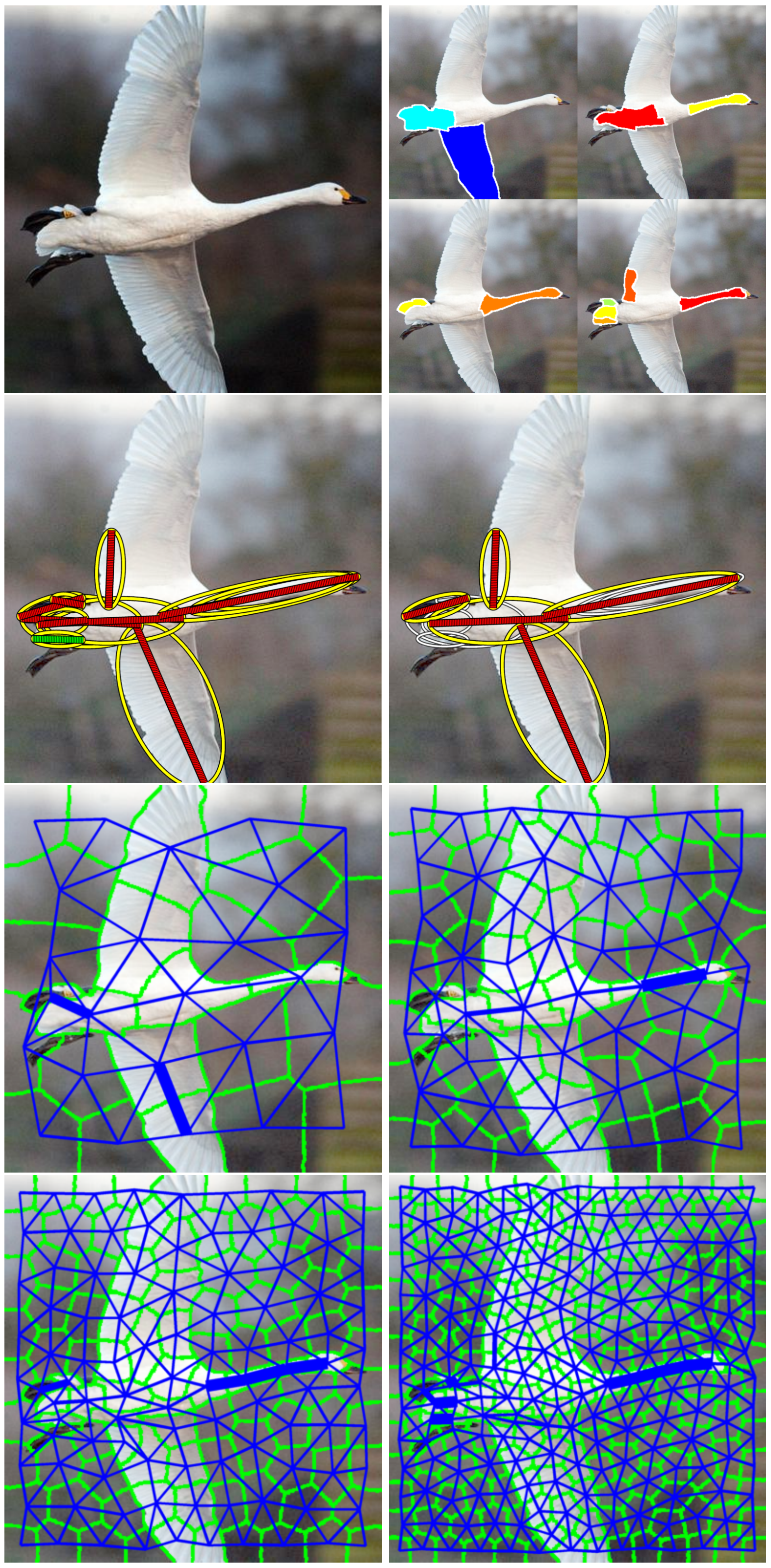
(f)

Figure 5.7



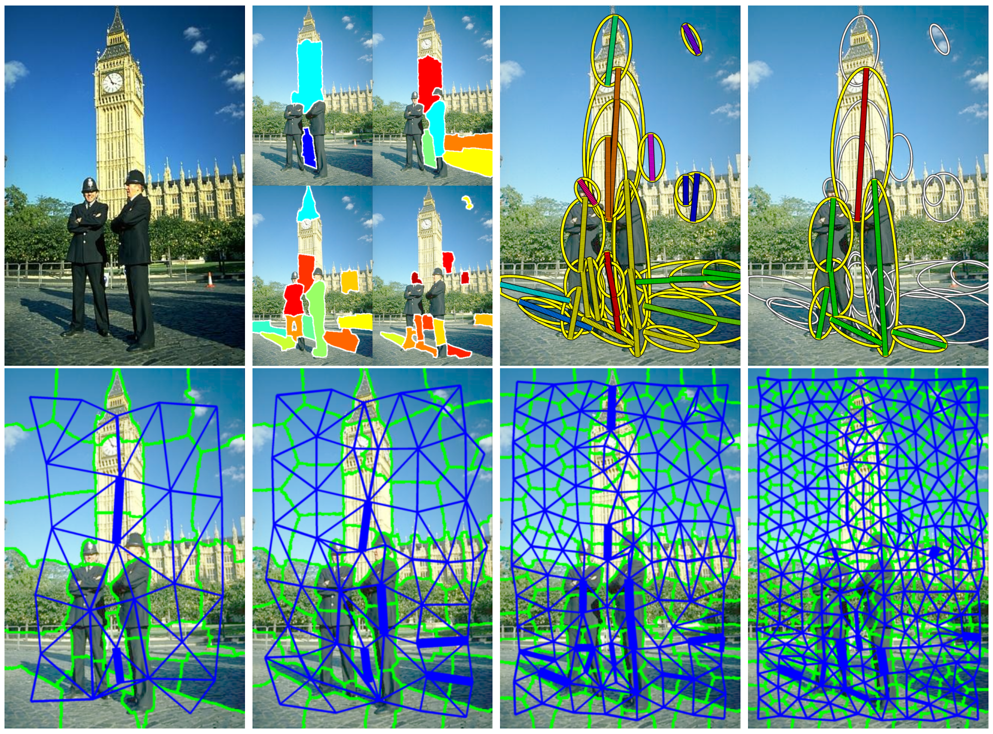
(g)

Figure 5.7



(h)

Figure 5.7


(i)

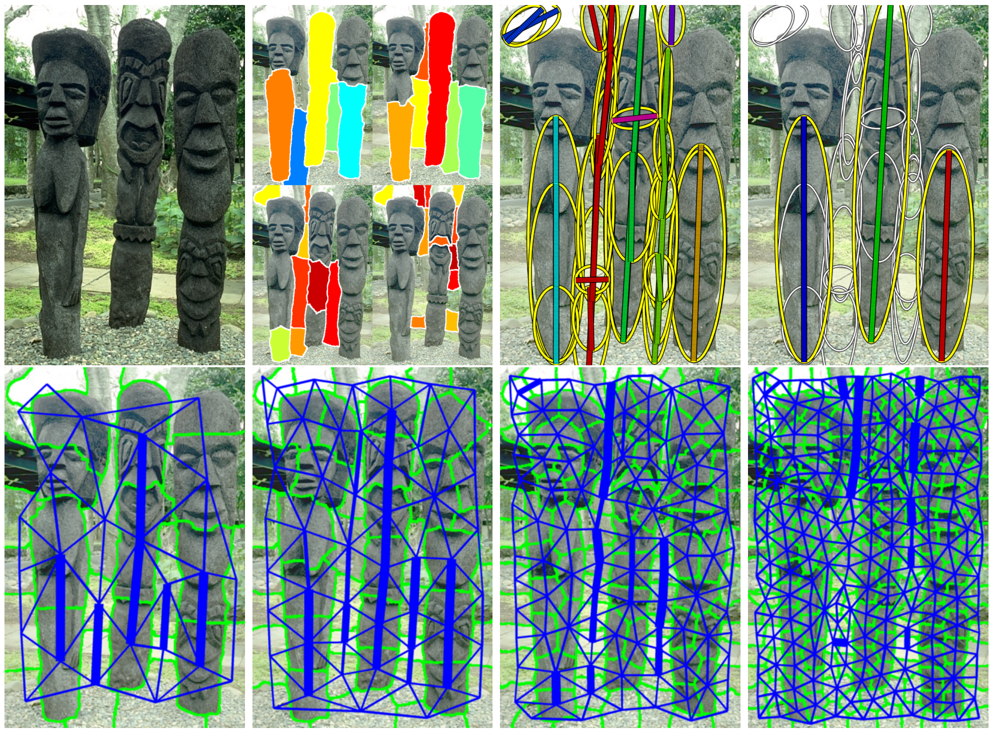
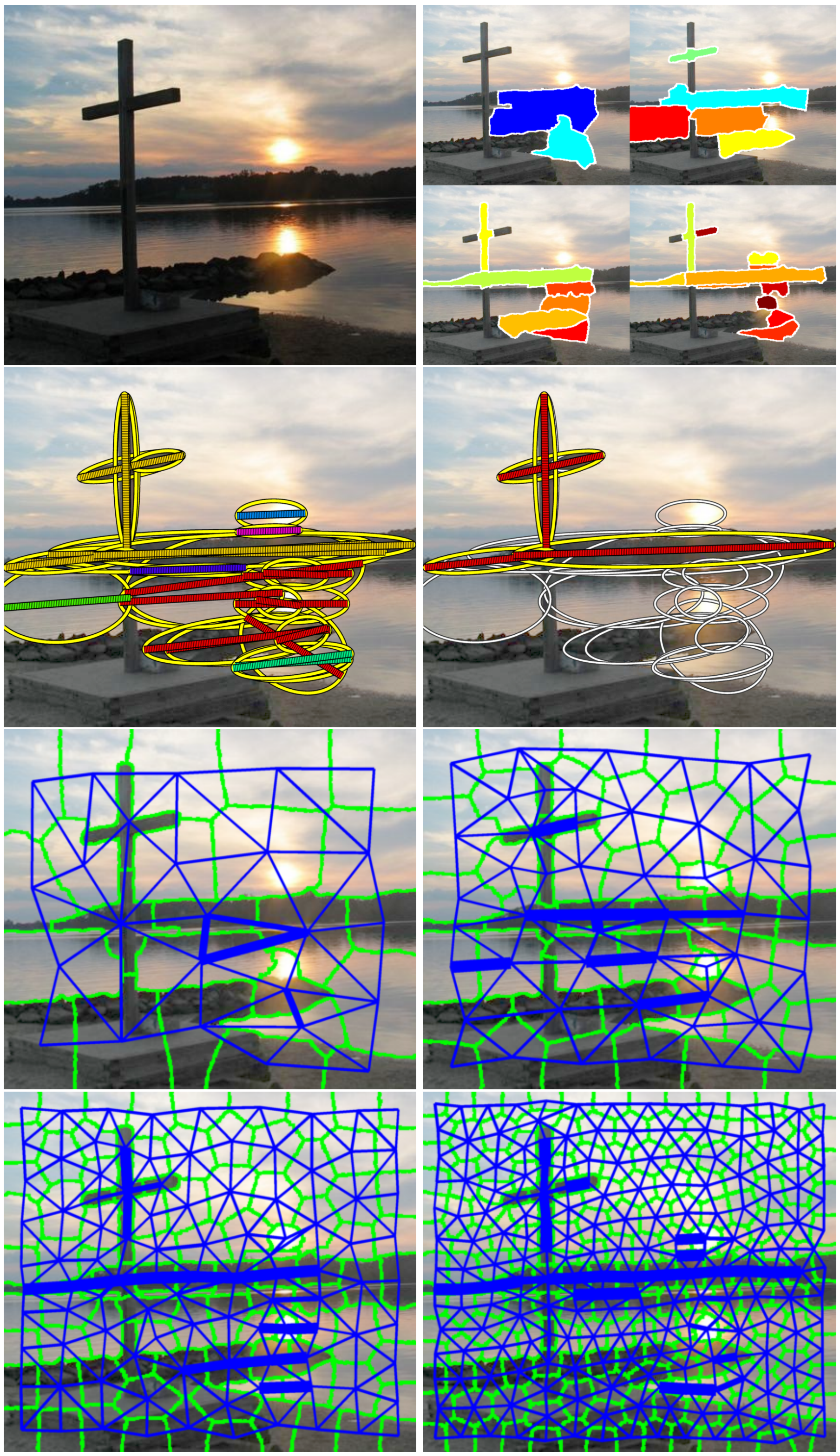
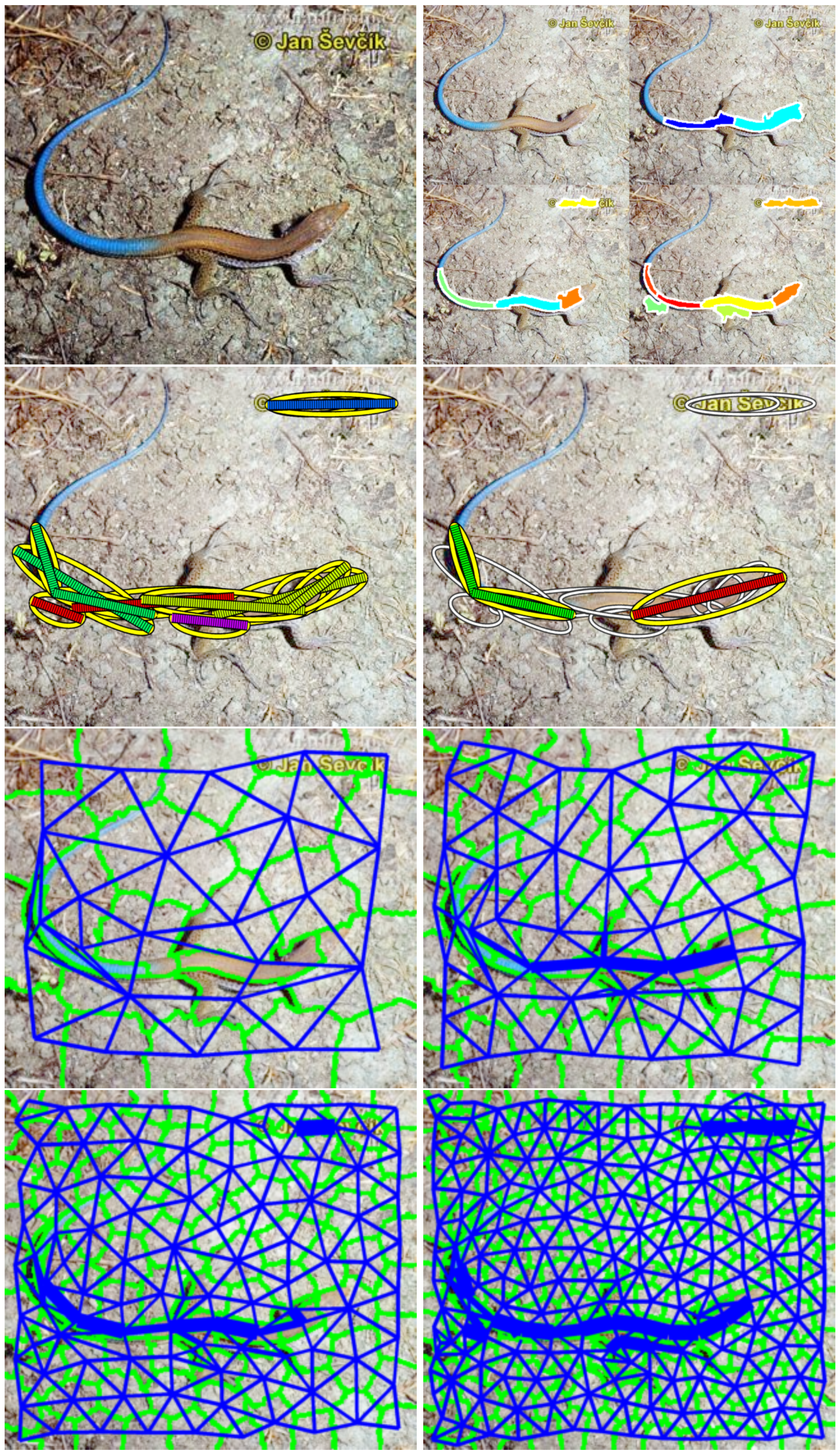

(j)

Figure 5.7



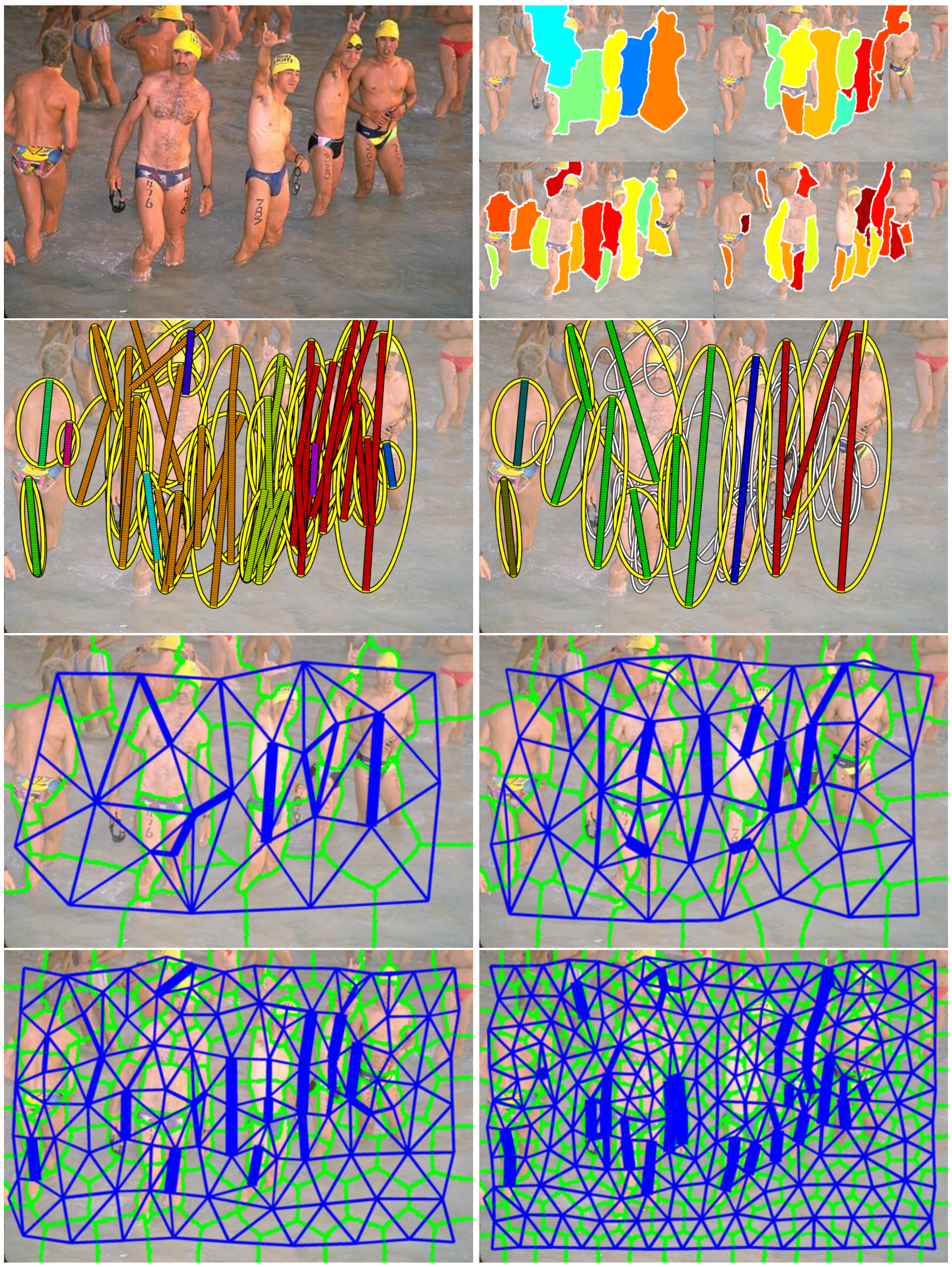
(k)

Figure 5.7



(l)

Figure 5.7



(m)

Figure 5.7

model, which could be used in a top-down manner to overcome such segmentation problems. A similar effect can be seen in Figs. 5.7(j,k), where the top parts of the statues and the center of the cross do not have sufficient contrast with their backgrounds. Figs. 5.7(d,l) illustrate the second cause, where the tail of an airplane and the tail of a lizard are not captured since they are too thin to be well-represented by even our finest superpixel scale.

Additional issues in part detection arise when parts are tapered or have a curved axis. Fig. 5.7(l) shows that although the main parts of the lizard are found, the tail is not composed of a single part since our system assumes parts with straight symmetry axes. Part tapering, and other deformations from roughly parallel part boundaries, also hinder detection. Figs. 5.7(e,f,i) illustrate this effect. Tapering can result in wide sections of a part being captured at a coarse superpixel scale, and thinner sections being captured at a finer scale. Object extremities, such as the hand of the Jesus statue (Fig. 5.7(e)), the tail of the swan (Fig. 5.7(f)), and the tip of the tower (Fig. 5.7(i)), are all better represented at a finer scale than the remainder of these corresponding parts. Even if this effect is overcome and there is a single superpixel scale at which the whole part is well-captured, superpixel affinities are still negatively affected since most parts in our training set had more parallel boundaries.

The part grouping stage of our system also suffers from some limitations. Starting the discussion from our part selection stage, we see that it is not perfect, occasionally resulting in suboptimal part groups. Figs. 5.7(b,e) show examples where more precise arm parts were discarded over inferior arm representations. However, our greatest failure mode is part clustering itself. While the affinities at the superpixel level consist of strong appearance and shape components, our part attachment affinities include weaker constraints and are more susceptible to low contrast between objects and their background. In combination with our greedy part clustering approach, this can often lead to bleeding. In Fig. 5.7(k), too many parts are clustered due to a lack of contrast at their attachment boundaries (symmetric strip of horizon landscape accidentally grouped with vertical mast). Like Fig. 5.7(h), a candidate model may be required to resolve such ambiguous part attachments. Finally, while Fig. 5.7(m) shows that most swimmers and/or their parts were successfully detected, the vast number of resulting parts and their proximity overwhelm our greedy part grouping approach.

To provide a quantitative evaluation of our part detection strategy, we compare its precision and recall to the method of Lindeberg et al. [54], used to generate the symmetric parts shown in Fig. 5.1(g). Both methods are evaluated on 61 test images from the Weizmann Horse Dataset [8]. A ground truth part is considered to be recovered if its normalized overlap (in area) with one of the detected parts is above a threshold (0.4). Our part detection offers a significant improvement in both precision and recall (Fig. 5.8). Moreover, in [54], no effort is made to distinguish part occlusion from part attachment; parts are simply grouped if they overlap. Note that both methods achieve low precision. This is partially due to the fact that there are other symmetric parts in the images, besides the horses' parts, that were not marked in the ground truth. Moreover, due to our multiscale part detection approach, the same ground truth part may be recovered at multiple scales, hindering precision in the absence of some redundancy removal step.

We also provide quantitative evaluation for part grouping. Given a user-specified parameter setting (k in Eqn. 5 in [31]), our first, greedy, part grouping method groups the parts into multiple disjoint clusters. Our second, unbalanced Ncuts-based approach, generates potentially overlapping clusters of parts given parameters α_p (which we fix at 1 for this experiment) and σ_p (that was used to convert part attachment distance into similarity). Leaving for now the part selection step, we compare the two part clustering approaches on the tasks of figure/ground segmentation and part grouping. In the first task, our goal is the same as that in Chapter 4, aiming to select a set of parts that covers the object as closely as possible. The second task is motivated by generic object recognition, where correct part groups are needed to index into a dataset of objects. Here it is less important to achieve a good pixel-wise covering of the object rather than obtain largest possible groups of parts mapping to parts on the object.

Given an image, both methods return multiple clusters of parts corresponding to part group-

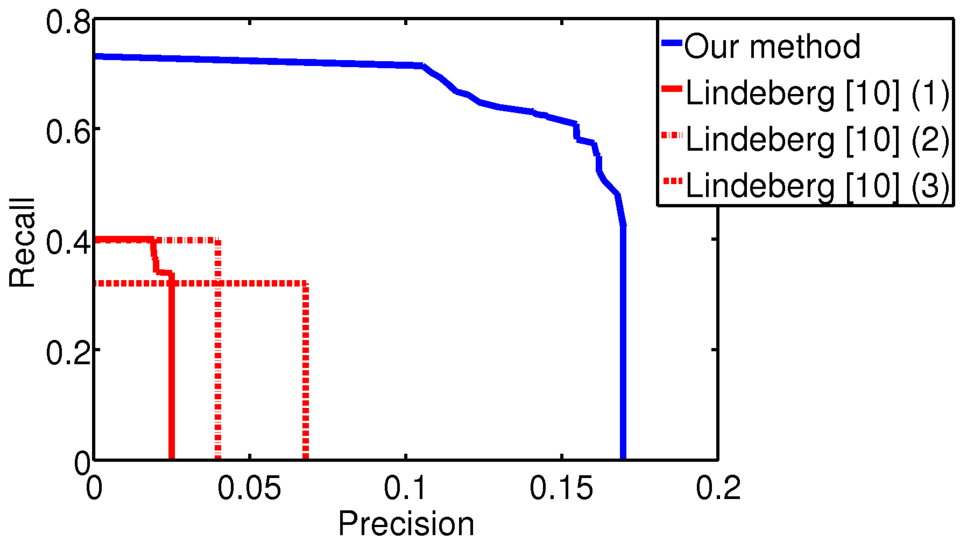
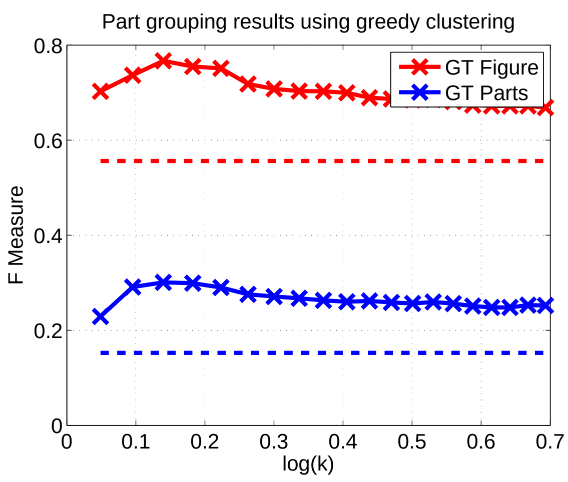


Figure 5.8: Precision vs recall of part detection. Due to the low precision of [54], we prune small parts to increase precision: (1) no pruning, (2) prune parts whose major axis is less than 10 pixels, (3) prune parts whose major axis is less than 20 pixels.

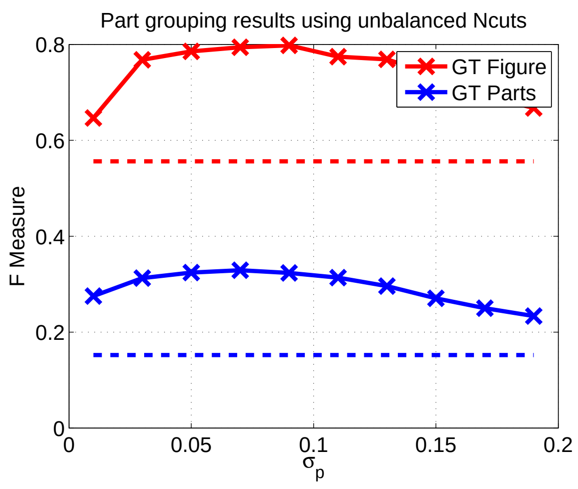
ing hypotheses that would be used for high-level tasks. What is important is that at least one of these clusters corresponds to an object of interest. Thus, for each image we first compute the F measure (either using figure-ground precision/recall or part precision/recall) and chose the solution with the best F measure for each image. We average the best per-image F measures across all images, giving us a mean F measure for each parameter setting for the two methods. Figure 5.9 shows the performance of both methods as a function of their parameters. Observe that the unbalanced Ncuts approach is able to achieve slightly better performance.

5.6 Limitations and Future Work

A number of limitations of the current framework will be addressed in future research. We have shown that successful part recovery strongly depends on our ability to recover good medial disc approximations. To improve the quality of medial point hypotheses, we are exploring a more powerful superpixel extraction framework that allows greater control over compactness,



(a)



(b)

Figure 5.9: Part grouping performance. (a) Part grouping using the greedy method in [31], (b) Part grouping by minimizing unbalanced Ncuts using parametric maxflow. In both cases, performance is measured as a function of method parameters (k in [31] and σ_p in unbalanced NCuts). Dotted lines provide the baselines for the corresponding evaluation tasks, measuring performance when all the detected parts are selected.

along with a multiscale Pb detector. In addition, our current system is restricted to grouping superpixels independently at each scale. Lifting this constraint would increase grouping complexity and complicate affinity computation, but will also make our part model more flexible and is therefore a subject of future work. We also intend to relax our linear axis model to include curved shapes; for example, the ellipse model could easily be replaced by a deformable superellipse.

The chosen clustering approach proves to be another serious limitation. Both stages of our framework, part recovery and part grouping, rely on greedy clustering methods. This allows us to overcome significant computational issues in extracting and grouping parts bottom-up, but can result in over- or undersegmentation in both part recovery and part clustering. The issue is of greater concern in the second stage, where affinities provide only a crude guideline to true part attachments. Exploring global optimization techniques should lead to a better solution of this grouping problem. The use of unbalanced Ncuts, instead of our main greedy approach, has already been shown to be a promising step in this direction. In future work, we plan to further address this issue through better optimization and the incorporation of more global constraints, such as closure.

5.7 Conclusions

We have presented a constructive approach to detecting symmetric parts at different scales by grouping small compact regions that can be interpreted as deformable versions of maximal disks whose centers make up a skeletal branch. In this way, symmetry is *integrated* into the region segmentation process through a compactness constraint, while region *merging* is driven by a symmetry-based affinity learned from training data. Detected parts are assembled into objects by exploiting the regularities of part attachments in supervised training data. The resulting framework can recover a skeletal-like decomposition of an object from real images without requiring any prior knowledge of scene content and without requiring figure-ground segmentation.

Chapter 6

Conclusions

6.1 Review of contributions

The path from image pixels to a full semantic understanding of a scene is long and treacherous. Shortcuts can occasionally be made in case of simplified scenes and object detection scenarios, but in general, some level of bottom-up perceptual grouping is essential before object-specific models can be applied. The required amount of grouping is task dependant, with low-level grouping serving better feature extraction, segmentation, recognition, as well as mid-level perceptual grouping, which is needed for more generic scene analysis. All of the above are important goals in computer vision, suggesting that both low and mid-level grouping are necessary in their own way.

This thesis addressed perceptual grouping at its different levels. With a growing need for fast and accurate superpixel extraction, we start from employing low-level grouping cues to obtain compact superpixels, our first contribution. While superpixels are usually obtained by tuning the parameters of standard segmentation algorithms towards oversegmentation, ours is a specialized superpixel extraction approach. We show that most of the standard segmentation techniques are very fast but do not encode any shape constraints, and therefore suffer from significant undersegmentation. Normalized Cuts, which does encode such constraints, results in much better performance but is very computationally intensive.

Our method takes the best of both worlds, encoding compactness constraints into a local level-set curve evolution framework. For dense superpixel segmentations, it achieves comparable performance to Normalized Cuts, a global approach that is expected to do better in general, while being orders of magnitude faster. The framework is flexible and can accommodate additional shape constraints in the future. However, what is more important is that such compact superpixels can prove to be of great help to higher level vision tasks, which is the subject of our next two contributions.

Bottom-up figure/ground segmentation in the absence of a scene prior has received little recent attention in the computer vision community. Low-level segmentation approaches, building on purely local appearance cues, are unlikely to extract whole objects from realistic images. Recent developments in closure detection showed that globally optimizing a closure cost computed over short fragments of image edges brings us much closer to the ultimate goal of bottom-up figure extraction. However, such approaches have to cope with the prohibitive complexity of grouping a vast number of contour fragments into large and meaningful cycles. It often leads to the use of heuristics and greedy grouping strategies, or imposes additional constraints on the closure cost.

In our second contribution, we show that a standard closure cost can be globally optimized by grouping superpixels instead of finding contour cycles. The boundary of any set of connected superpixels implicitly represents a closed contour. The same is not true for groups of contour fragments, making closure detection simpler when working with superpixels. In addition, superpixels not only provide an ideal scope for the computation of boundary shape features, but give easy access to internal region information, such as area or appearance. This enables us to formulate two closure costs where a learned measure of contour gap is weighted against internal region area or internal homogeneity. Using a polynomial minimum cuts approach, we are able to efficiently recover a small number of figure hypotheses, capturing most of the objects in a scene. A similar approach can be used to recover closures in the spatiotemporal domain, where we efficiently recover a small set coherently moving components from a video.

In our final contribution, compact superpixels serve an even more important role, helping us in the extraction and grouping of symmetric parts. Until now, superpixels were used mainly to reduce complexity. Shape constraints on superpixels were only indirectly important as they reduce undersegmentation. In our final contribution, however, superpixels are not only essential to ease the computational burden, but their compactness is used explicitly.

Starting from the observation that abstract part-based representations are essential for generic recognition, we set out on the quest of recovering such representations bottom-up. Motivated by the skeletonization literature, we opt for working with symmetric parts. However, skeletonization approaches assume a prior figure-ground segmentation, and are sensitive to small shape perturbations. On the other hand, methods that detect symmetry bottom-up suffer from poor precision/recall (filter-based approaches) or prohibitive grouping complexity (contour-based approaches). In contrast to these approaches, we efficiently extract and group symmetric parts with no figure/ground assumptions. Our key contribution is the observation that at an appropriate superpixel resolution, superpixels can serve as good data-driven medial disc approximations, which can be grouped to yield the symmetric parts of an object. Thus, symmetric parts correspond to chains of superpixels at the right superpixel scale. Using multiple superpixel scales enables us to capture most of the parts in an image.

Unlike most symmetry detection approaches that rely on precise geometric relations, we cluster superpixels based on a pairwise superpixel affinity corresponding to the probability of a given superpixel pair to be a section of a symmetric part. Building on both appearance and shape features, our affinity is learned from a set of real training images, making our framework more applicable to real images. A similar approach is taken to learn a part attachment affinity and used to cluster symmetric parts, something which most bottom-up symmetry extraction methods stop short of. Our final results illustrate that while our framework was trained on images of horses, we learn generic grouping rules and are able to extract full skeleton-like representations of objects from a variety of challenging images from different domains.

6.2 Discussion and Future Work

Promising results were shown in several important perceptual grouping tasks. Building on these results opens the path to a great deal of potential future work in perceptual grouping and object recognition. Moreover, while our methods were shown to push the state-of-the-art in their respective perceptual grouping domains, they suffer from a number of drawbacks which should be addressed. We therefore conclude this thesis with a discussion of possible extensions and improvements.

Starting with TurboPixels, we point the reader to the fact that as any local segmentation technique, it too suffers from undersegmentation if superpixel resolution is too coarse. We provide one simple gradient-based pixel affinity and show that our system can work with a more elaborate Pb-based affinity, but it is a far cry from a complete black-box superpixel extraction approach sought by the community. A complete black-box system would call for a better pixel affinity, incorporating both different image cues, as well as information from multiple image scales. Our uniform superpixel seeding is also not applicable in some scenarios. For example, handling narrow or small objects would require us to place more seeds in such areas.

Finally, the level-set formulation in TurboPixels allows for some natural extensions. We have already shown how the original framework can be extended to obtain stable spatiotemporal superpixels (see Section 4.7.3). It is possible to further extend the approach towards 3D domains, such as supervoxel extraction in medical images. In such domains, using superpixels (or supervoxels) is of even greater importance, as an additional dimension results in a prohibitive number of voxels. Level-sets also enabled us to impose compactness and smoothness constraints. In future work, additional, more shape-specific constraints could be explored.

Relying on superpixels, our closure detection approach was able to extract a small number of good closure hypotheses in an efficient manner. However, unlike previous methods that could globally optimize a measure of gap relative to perimeter, we are not able to do so due to restrictions of the parametric maxflow framework. Arguably, minimizing gap over area or a measure of internal homogeneity was shown to be a superior cost, but its minima do not always serve as good figure hypotheses. It would be interesting to explore additional costs, encoding stronger grouping constraints that might result in a better set of hypotheses. For example, we are considering the use of superpixel junctions to add contour smoothness constraints. The challenge is to incorporate these additional cues, while still maintaining all the constraints that enable us to efficiently minimize the cost using parametric maxflow. It would be even more interesting to see if one could train a global closure cost using a set of real images, learning the optimal combination of different perceptual grouping features. Finally, we could start making use of the figure hypotheses acquired by our framework for higher-level tasks, such as object recognition.

In our last contribution, we set our sights even higher, attempting to extract and group symmetric parts. While using superpixels brought us closer to an ultimate, purely bottom-up solution of this difficult problem, we still resorted to the use of greedy clustering techniques for both part detection and grouping stages. As any greedy approach, our method also suffers from undersegmentation in the presence of weak affinities. This is particularly an issue at the part grouping stage, where pairwise affinities are weaker than the affinities used to cluster superpixels into parts. While future efforts should undoubtedly be spent on the design of better affinities at both stages, employing a more global clustering technique should lead to better performance. Moreover, the same complexity that forced us to employ a greedy grouping approach also resulted in breaking the framework into separate part detection and grouping steps. Our ultimate framework would integrate both of these stages, placing them under a unified mathematical formulation. Moving away from potential improvements to the realm of utilizing our system's output, the final challenge lies in using the extracted symmetric representations for generic object recognition.

In conclusion, note that the three works described in this thesis were somewhat independent in their nature. As humans, we make use of multiple grouping cues at different levels to better perceive our world. Therefore, we believe that the greatest potential for enhancement lies in a successful merger of our different frameworks. First, instead of working with compact superpixels to detect parts, we can extend TurboPixels to detect symmetric parts by encoding symmetric shape constraints directly into superpixel evolution itself. Moreover, Chapters 4 and 5 build on Ncuts-based superpixels rather than our own TurboPixels. Ncuts in combination with Pb or globalPb produces better results than TurboPixels, especially at coarse superpixel resolutions which are necessary as part of a multiscale superpixel analysis in Chapter 5. While we do not believe that TurboPixels can be extended to very coarse superpixel resolutions due to the local nature of the algorithm, coarse segmentations can be obtained efficiently by using Ncuts over fine TurboPixels segmentations rather than raw pixels. Second, while our part grouping stage would benefit from a more global clustering approach, it would gain even more if a global closure constraint were imposed on the resulting groups. An attentive reader has probably noticed that work from Chapter 4, that could undoubtedly aid in the extraction and grouping of symmetric parts, was not used in Chapter 5. The reason for this is that chronologically the work in Chapter 5 predates that of Chapter 4. As a result, though we are currently working on extracting skeletons based on the detected shape hypotheses from Chapter 4 or using closure to detect better object parts, this was not done at the time of writing Chapter 5. In the long run, adding more grouping cues, improving on the above techniques, and integrating **all** of them into a single mathematically elegant framework are the keys to success. Taking these next steps would allow for a far better and more generic scene analysis.

Bibliography

[1] Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2007.

[2] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[3] P.J. Besl and R.C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 1988.

[4] I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics and Image Processing*, 32:29–73, 1985.

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

[6] T. O. Binford. Visual perception by computer. In *Proceedings, IEEE Conference on Systems and Control*, Miami, FL, 1971.

[7] H. Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[8] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *European Conference on Computer Vision*, pages 109–124, 2002.

[9] K.L. Boyer and S. Sarkar. Perceptual organization in computer vision: Status, challenges, and potential. *Computer Vision and Image Understanding*, 76(1):1–6, October 1999.

[10] M. Brady and H. Asada. Smoothed local symmetries and their implementation. *International Journal of Robotics Research*, 3(3):36–61, 1984.

[11] R.A. Brooks. Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–149, March 1983.

[12] Richard J. Campbell and Patrick J. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166–210, 2001.

[13] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[14] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[15] Vicent Caselles, Francine Catté, Tomeu Coll, and Françoise Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66(1):1–31, December 1993.

[16] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22:61–79, 1995.

[17] Tat-Jen Cham and R. Cipolla. Geometric saliency of curve correspondences and grouping of symmetric contours. In *European Conference on Computer Vision*, pages 385–398, 1996.

[18] Tat-Jen Cham and Roberto Cipolla. Symmetry detection through local skewed symmetries. *Image Vision Comput.*, 13(5):439–450, 1995.

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

[20] J. H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31(2):159–183, 1987.

[21] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models–their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[22] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1124–1131, Washington, DC, USA, 2005. IEEE Computer Society.

[23] I. J. Cox, S. B. Rao, and Y. Zhong. ”ratio regions”: A technique for image segmentation. In *IEEE International Conference on Pattern Recognition*, page 557, Washington, DC, USA, 1996. IEEE Computer Society.

[24] J. Crowley and A. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):156–169, March 1984.

[25] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13:492–498, 1967.

[26] James Elder and Steven Zucker. A measure of closure. *Vision Research*, 34:3361–3369, 1994.

[27] James H. Elder and Steven W. Zucker. Computing contour closure. In *European Conference on Computer Vision*, pages 399–412, London, UK, 1996. Springer-Verlag.

[28] J.H. Elder and S.W. Zucker. Local scale control for edge detection and blur estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(7):699–716, Jul 1998.

[29] Francisco J. Estrada and Allan D. Jepson. Perceptual grouping for contour extraction. In *IEEE International Conference on Pattern Recognition*, pages 32–35, 2004.

[30] Francisco J. Estrada and Allan D. Jepson. Robust boundary detection with adaptive grouping. In *Computer Vision and Pattern Recognition Workshop*, page 184, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[31] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[32] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, Jan. 2008.

[33] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: combining patch-based and gradient-based approaches. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 54–61, June 2003.

[34] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[35] Peter D. Grünwald, In Jae Myung, and Mark A. Pitt. *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. The MIT Press, 2005.

[36] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 695–702, 2004.

[37] Xuming He, Richard S. Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. In *European Conference on Computer Vision*, pages 338–351, 2006.

[38] H. Helmholtz. *Treatise on Physiological Optics*. New York: Dover, 1962 (first published in 1867).

[39] D. D. Hoffman, Whitman Richards, Alex Pentland, John Rubin, and Joseph Scheuhammer. Parts of recognition. *Cognition*, 18:65–96, 1984.

[40] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Transactions on Graphics*, 24(3):577–584, 2005.

[41] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *IEEE International Conference on Computer Vision*, pages 654–661, Washington, DC, USA, 2005. IEEE Computer Society.

[42] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split and merge procedure. In *IEEE International Conference on Pattern Recognition*, pages 424–433, 1974.

[43] Yuchi Huang, Qingshan Liu, and D. Metaxas. Video object segmentation by hypergraph cut. *IEEE International Conference on Computer Vision and Pattern Recognition*, 0:1738–1745, 2009.

[44] D.W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):23–37, Jan 1996.

[45] Hong Jeong and C. I. Kim. Adaptive determination of filter scales for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(5):579–585, 1992.

[46] I.H. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1075–1088, 2001.

[47] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[48] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Gradient flows and geometric active contour models. In *IEEE International Conference on Computer Vision*, page 810, Washington, DC, USA, 1995. IEEE Computer Society.

[49] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. Toward a computational theory of shape: an overview. In *European Conference on Computer Vision*, pages 402–407, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[50] V. Kolmogorov, Y.Y. Boykov, and C. Rother. Applications of parametric maxflow in computer vision. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[51] Yvan G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102, 1989.

[52] A. Levinshtein, C. Sminchisescu, and S.J. Dickinson. Learning hierarchical shape models from examples. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 251–267, 2005.

[53] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 465–470, Jun 1996.

[54] Tony Lindeberg and Lars Bretzner. Real-time scale selection in hybrid multi-scale representations. In *Scale-Space*, volume 2695 of *Springer LNCS*, pages 148–163, 2003.

[55] T.L. Liu, D. Geiger, and A.L. Yuille. Segmenting by seeking the symmetry axis. In *IEEE International Conference on Pattern Recognition*, volume 2, pages 994–998, Aug 1998.

[56] Liana M. Lorigo, W. Eric L. Grimson, Olivier Faugeras, Renaud Keriven, Ron Kikinis, Arya Nabavi, and Carl-Fredrik Westin. Codimension - two geodesic active contours for the segmentation of tubular structures. *IEEE International Conference on Computer Vision and Pattern Recognition*, 1:1444, 2000.

[57] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, USA, 1985.

[58] D G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artif. Intell.*, 31(3):355–395, 1987.

[59] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[60] Diego Macrini, Kaleem Siddiqi, and Sven Dickinson. From skeletons to bone graphs: Medial abstraction for object recognition. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[61] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[62] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:158–175, 1995.

[63] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:530–549, 2004.

[64] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142, London, UK, 2002. Springer-Verlag.

[65] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, Oct. 2005.

[66] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[67] R. Mohan and R. Nevatia. Perceptual organization for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):616–635, 1992.

[68] Alastair P. Moore, Simon J. D. Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[69] G. Mori, S. Belongie, and J. Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, Nov. 2005.

[70] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 326–333, 2004.

[71] Randal C. Nelson and Andrea Selinger. A cubist approach to object recognition. In *IEEE International Conference on Computer Vision*, pages 614–621, 1998.

[72] Ramakant Nevatia and Thomas O. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8(1):77 – 98, 1977.

[73] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[74] S. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, 1999.

[75] T. Pavlidis. Segmentation of pictures and maps through functional approximation. *Computer Graphics and Image Processing*, 1(4):360–372, December 1972.

[76] M. Pelillo, K. Siddiqi, and S.W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, Nov 1999.

[77] A. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.

[78] Alex P. Pentland. Automatic extraction of deformable part models. *International Journal of Computer Vision*, 4(2):107–126, 1990.

[79] Jean Ponce. On characterizing ribbons and finding skewed symmetries. *Computer Vision, Graphics and Image Processing*, 52(3):328–340, 1990.

[80] X. Ren and J. Malik. Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision*, volume 1, pages 10–17, Oct. 2003.

[81] Xiaofeng Ren, Charless C. Fowlkes, and Jitendra Malik. Cue integration in figure/ground labeling. In *Advances in Neural Information Processing Systems*, 2005.

[82] Xiaofeng Ren, Charless C. Fowlkes, and Jitendra Malik. Scale-invariant contour completion using conditional random fields. In *IEEE International Conference on Computer Vision*, volume 2, pages 1214–1221, 2005.

[83] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[84] P. Saint-Marc, H. Rom, and G. Medioni. B-spline contour representation and symmetry detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1191–1197, 1993.

[85] Pablo Sala and Sven Dickinson. Model-based perceptual grouping and shape abstraction. In *Proceedings, Sixth IEEE Computer Society Workshop on Perceptual Organization in Computer Vision*, 2008.

[86] Sudeep Sarkar and Padmanabhan Soundararajan. Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):504–525, 2000.

[87] Eric Saund. Finding perceptually closed paths in sketches and drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:475–491, 2003.

[88] Stan Sclaroff and Lifeng Liu. Deformable shape detection and description via model-based region grouping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):475–489, 2001.

[89] T.B. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, May 2004.

[90] Thomas B. Sebastian, Hseyin Tek, Joseph J. Crisco, and Benjamin B. Kimia. Segmentation of carpal bones from ct images using skeletally coupled deformable models. *Medical Image Analysis*, 7(1):21 – 45, 2003.

[91] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast multiscale image segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 70–77, 2000.

[92] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *IEEE International Conference on Computer Vision*, pages 321–327, 1988.

[93] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[94] A. Shokoufandeh, L. Bretzner, D. Macrini, M. F. Demirci, C. Jönsson, and S. Dickinson. The representation and matching of categorical shape. *Computer Vision and Image Understanding*, 103(2):139–154, 2006.

[95] K. Siddiqi, Y. B. Lauziere, A. Tannenbaum, and S. W. Zucker. Area and length minimizing flows for shape segmentation. *IEEE Transactions on Image Processing*, 7:433–443, March 1998.

[96] Kaleem Siddiqi, Sylvain Bouix, Allen Tannenbaum, and Steven W. Zucker. Hamilton-jacobi skeletons. *International Journal of Computer Vision*, 48(3):215–231, 2002.

[97] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker Y. Shock graphs and shape matching. *International Journal of Computer Vision*, 35:13–32, 1999.

[98] J. Stahl and S. Wang. Globally optimal grouping for symmetric closed boundaries by combining boundary and region information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):395–411, 2008.

[99] J.S. Stahl and Song Wang. Edge grouping combining boundary and region information. *IEEE Transactions on Image Processing*, 16(10):2590–2606, Oct. 2007.

[100] A.N. Stein, D. Hoiem, and M. Hebert. Learning to find object boundaries using motion cues. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[101] H. Tek and B.B. Kimia. Image segmentation by reaction-diffusion bubbles. *IEEE International Conference on Computer Vision*, 0:156, 1995.

[102] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, Jun 1991.

[103] S. Wang, T. Kubota, J.M. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):546–561, April 2005.

[104] S. Wang and J.M. Siskind. Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):675–690, June 2003.

[105] M. Wertheimer. Laws of organization in perceptual forms. In W. Ellis, editor, *Source Book of Gestalt Psychology*. Harcourt, Brace, New York, NY, 1938.

[106] L. R. Williams and D. W. Jacobs. Stochastic completion fields: a neural model of illusory contour shape and salience. In *IEEE International Conference on Computer Vision*, page 408, 1995.

[107] Lance R. Williams and Allen R. Hanson. Perceptual completion of occluded surfaces. *Computer Vision and Image Understanding*, 64(1):1–20, 1996.

[108] A.P. Witkin and J.M. Tenenbaum. On the role of structure in vision. In *HMV*, pages 481–543, 1983.

[109] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[110] Antti Ylä-Jääski and Frank Ade. Grouping symmetrical structures for object segmentation and description. *Computer Vision and Image Understanding*, 63(3):399–417, 1996.

[111] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *IEEE International Conference on Computer Vision*, page 313, Washington, DC, USA, 2003. IEEE Computer Society.

[112] Qihui Zhu, Gang Song, and Jianbo Shi. Untangling cycles for contour grouping. In *IEEE International Conference on Computer Vision*, Oct. 2007.

[113] Song-Chun Zhu. Embedding gestalt laws in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1170–1187, 1999.

[114] S.W. Zucker. Region growing: Childhood and adolescence. *Computer Graphics and Image Processing*, 5(3):382–399, September 1976.