Class-based Single View Reconstruction

D.Phil Thesis

Robotics Research Group
Department of Engineering Science
University of Oxford

Supervisors:
Dr. Andrew Fitzgibbon
Prof. Andrew Zisserman

Mukta Prasad
Brasenose College

7th July 2009

Mukta Prasad
Brasenose College

Doctor of Philosophy
Trinity Term 2009

Class-based Single View Reconstruction

Abstract

The aim of this thesis is to construct a realistic, freeform 3D model of an object from a single view, given the knowledge of the class it belongs to. "Classes" can be–fruits (oranges, apples etc.), flowers (lilies, hibiscus etc.), faces *etc.*

Single view reconstruction (SVR) is a severely under-constrained problem and relies on cues like shading, texture, occluding contour *etc.* In this thesis, first the projective properties of image silhouettes are exploited to effectively constrain the problem. We show how intuitive user-input and other image-based cues such as multi-local singularities (cusps) and creases can be incorporated to add more definition to the model. The problem is then, to find the smoothest surface, given the set of constraints. All the above constraints are incorporated as linear constraints in the optimization of a quadratic objective. The resultant framework is convex and can be solved easily. The parametric surface representation used here can model objects of any topology: genus $0, 1$ or higher.

An object class is strongly bound by characteristic shape, texture and a family of deformations. Thus, class information is an important cue for SVR, like texture and silhouette. Therefore, the problem of simultaneously reconstructing and learning class-specific shape models from photo collections is addressed next. Only a single view of each object instance is available, so this is an extension of the class-based SVR problem. Object classes which can be represented as wireframes are addressed, *e.g.* lily petals. We show that Non-Rigid Structure from Motion (NRSfM) can be extended to the scenario where each image is of a different object instance. However, this requires a novel method of defining correspondences. Instead of first finding correspondences and then employing existing Non-Rigid Structure from Motion (NRSfM) techniques, we frame this problem as one joint objective which integrates the correspondence finding problem with the other variables of NRSfM. This is solved jointly and analytically with effective bundle adjustment. A specialization of this method to stereo is shown to be an improvement over existing stereo approaches and methods in fitting 3D active shape models.

Work of this nature depends on learning from training data and often requires user-driven annotation. Minimal, intuitive annotation is used during the course of this work. However, we also show how class-based information can also be used for the automatic detection and segmentation of class instances. Local image-based information is used to learn a classifier. This helps to differentiate between image edges lying on object class boundaries, from others. This learning is used to improve detection and segmentation techniques such as Chamfer matching and ObjCut. As a result an end-to-end class-based SVR system is built, which automatically detects, segments and subsequently reconstructions an object class instance with the above-mentioned techniques.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Mukta Prasad, Brasenose College

Acknowledgements

This thesis would not have been possible without the magnanimous support of the Felix Scholarship and its donor, which enabled me to embark on this D.Phil and enjoy the unique Oxford experience. This research has also been supported by Microsoft Research.

I am lucky to have been able to learn from my advisors: Dr. Andrew Fitzgibbon (AWF) and Prof. Andrew Zisserman (AZ), who have nurtured and influenced this work and my research ethic with their inimitable styles. With his relentless zest for perfection of theory and implementation, AWF has always set high standards, both inspiring and impossible to match. AZ has infused optimism and taught me to value clarity of thought, presentation and always remember the bigger research picture.

Many thanks to my examiners Prof. Jean Ponce and Prof. David Murray for going through this thesis in painstaking detail. They have provided much encouragement and food for thought during our discussions.

I was motivated to do this D.Phil thanks to an inspiring introduction to this field by my undergraduate advisors Prof. C. V. Jawahar and Prof. P. J. Narayanan. Special thanks to M. Pawan Kumar who has been a most dependable friend and an inspiration all through. I am indebted to Tom Minka for being a great mentor and providing insightful critique in this work. Frederik Schaffalitzky, Aeron Buchanan, Ondréj Chum and Lyndsey Pickup have helped me find my feet during the initial stages of the D.Phil.

I extend heartfelt gratitude to my friends during this journey—Ramesh Chaturvedi, Karteek Alahari, Maria-Elena Nilsback, Oliver Woodford, Florian Schroff, Carl Ek, Pushmeet Kohli, Nicholas Apostoloff, Vittorio Ferrari, Patrick Buehler, Varun Gulshan, James Philbin, Srinika Ranasinghe, Chenoa Marquis, Sandeep Kakani, Ashok Ganapam, Siddharth Reddy, Madhuri Kottamraju, Padmapriya Ayyagari, Tanushree Mital and Shalav Gupta—during fun and serious times, work discussions and completely inconsequential deliberations.

Above all, I am extremely grateful to my parents and family for their unconditional support.

Contents

Chapter 1

Introduction

1.1 Goal

The aim of this thesis is to construct a plausible 3D model of an object, given only one view of the object and the fact that it belongs to a certain class. The term *class* here refers to the nature of the object, which is indicative of its properties like topology, shape, colour, texture *etc.* Classes can be generic such as fish, building, trees *etc.*, or more specific—clown fish, dolphins, lilies, ivy leaves *etc.* Class-based information about the object should encapsulate the answers to questions such as:

1. What does the typical object class instance look like?
2. How much and what kind of variation can be expected in the class? Which of a set of plausible instances are actually valid?

There are two aspects to the problem of class-based reconstruction:

1. Learn a robust, flexible class model from examples. An object class model (or shape model) is the gist of class-based information, that can be used to generate individual 3D models.
2. Use this class model to generate valid hypotheses for the 3D model given an image.

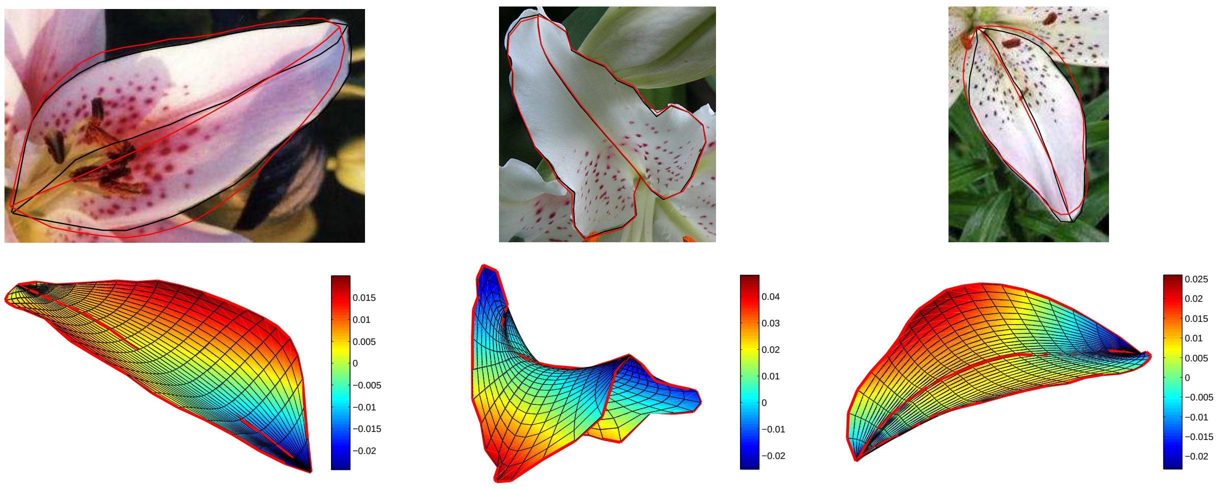


Figure 1.1: Learning class models: A collection of photos of lilies downloaded from the web is used to automatically learn a deformable shape model for the lily petal class. The wireframe reconstructions of lily petals and interpolated surfaces are seen in the bottom row. The projection of the wireframe upon the image is seen for visual evaluation of error in reprojection. The petal surfaces are coloured according to their depth values (z coordinates) in the petal coordinate frame for better shape perception.

Both aspects are relevant for research in class-based reconstruction. In figure 1.1 we show an example of how an object class (lily petals) is automatically learnt from an unordered photo collection. In figures 1.2 and 1.1 we show how such class models can be used to automatically detect, segment and reconstruct object class instances.

Single view reconstruction depends on the use of available cues about the nature of the object's shape such as shading, texture *etc.* This thesis mainly focuses on how *'object class'* is one such cue that can provide prior knowledge on the object shape and appearance. The object silhouette is another important image cue that constrains the family of possible 3D reconstructions to a plausible set. As a part of our work, we also show how an object's silhouette in a single image can be used to effectively model it in 3D. Figure 1.4 shows how objects of varying topology and complexity can be modelled by exploiting their silhouette information in novel ways.

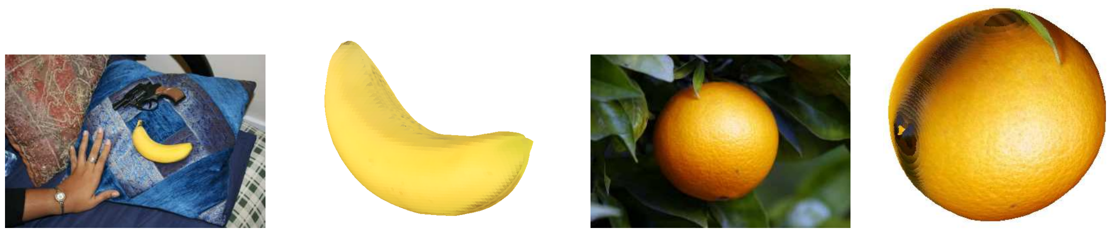


Figure 1.2: Class-based Single View Reconstruction I: Class models for classes such as bananas and oranges are learnt for a wide variety of pose, illumination, object variation and occlusion. This class model is used to automatically detect and segment the object instances. Given this segmentation, simple silhouette-based reconstruction can be used to reconstruct the individual instances *automatically.* Reconstructions for some example images are seen here. Information about the class 'orange' is used to automatically locate, segment and subsequently reconstruct it.

1.2 Challenges

In the field of Computer Vision and Graphics, the reconstruction of three-dimensional models of objects from multiple images has been an extensively researched problem. Methods like triangulation and space carving give us an estimate of the depth of a scene, in the presence of multiple images and camera information. For one image however, triangulation (or space carving) cannot be performed. An infinite number of 3D models can be found which project to the object in the image.

In the absence of multiple images, other image cues such as shading, geometry, texture, occluding contour *etc.* can still help in the retrieval of shape information. Again, a large amount of work has been done in these areas. A drawback of many of these methods is the need for elaborate, specific information, inevitably substituted by assumptions–such as number and nature of light sources, reflectance models *etc.* in the case of shading, and, nature and distribution of texture for texture-based cues.

Nevertheless such cues are crucial and information about the object class forms one such potent source of information. Objects belonging to a certain class display characteristic visual cues. Therefore, this field of work offers many potential interesting problems to work on—how to learn object class appearance and shape characteristics from commonly available examples in the world? How to use this knowledge to then detect, segment and

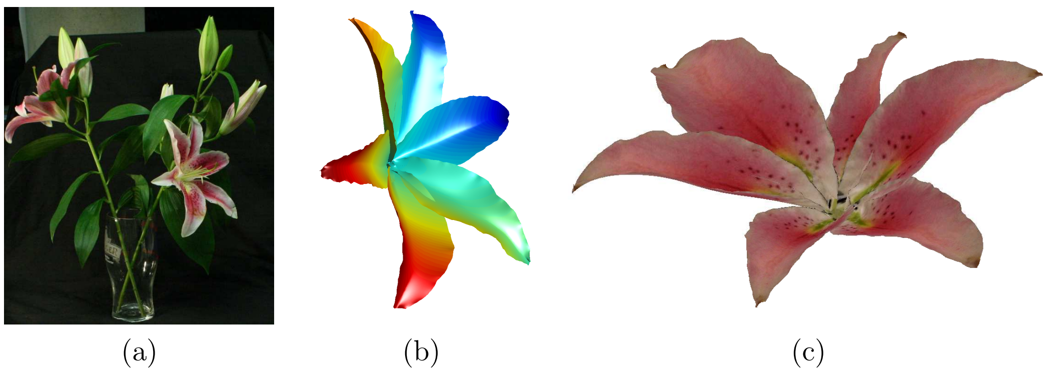


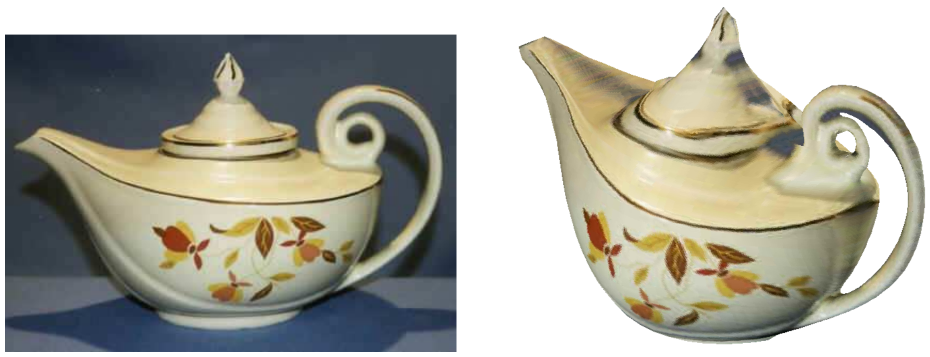
Figure 1.3: Class-based Single View Reconstruction II: Given and image (a) and a class model for lily petals (PCA bases derived from a set of 3D exemplars learnt from stereo, similar to [15]), we can fit a global flower model to best fit the given image. The mesh (b) and surface (c) corresponding to the reconstruction can be seen above in different views. First a global flower model with rigid petals is fit so that the pose and a rough estimate of the petals is assembled to form a valid flower projecting to the image. Subsequently, each petal is independently deformed to create the best reconstruction for the given image.

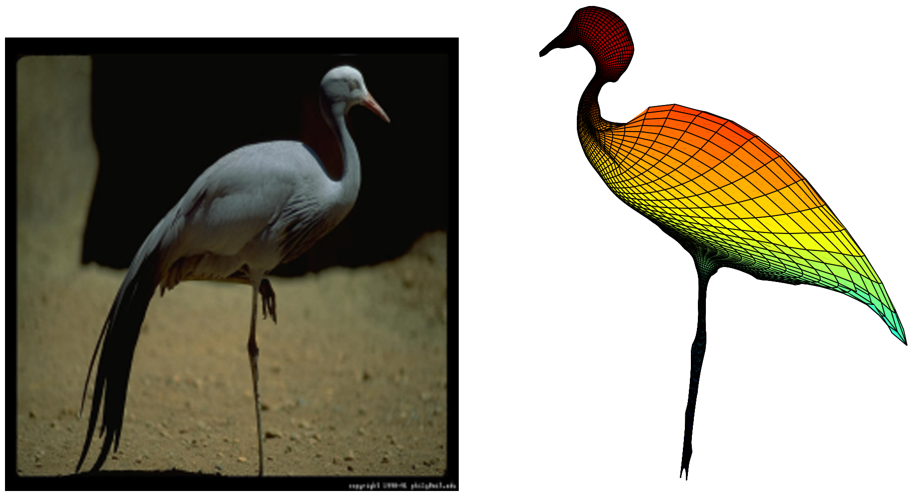
make intelligent guesses about object shapes? This field of work has gained momentum recently owing to renewed approaches based on geometry and machine learning.

1.3 Motivation

The general philosophy about reconstruction is: the more (images), the merrier (reconstruction). A second image enables better reconstruction of a part occluded in one image. Ambiguity and noise can be averaged out effectively too.

However, the importance of being able to guess accurately with minimal information— the crux of single view reconstruction— cannot be overstated. Most 3D models are acquired either thorough painstaking modelling on part of a user of a modelling system such as CAD, or through measurement devices such laser range systems. Such software, hardware and human effort is expensive. In many situations, acquiring multiple images is not possible, or practical. Forensic data, virtual paintings (Van Gogh self-portrait), archive photographs of people or towns form a few such examples. For many sketch-based modelling methods the user wants to create rapid 3D prototypes with minimal input. Multiple


(a) Teapot (genus 2)


(b) Ostrich (with concavities)

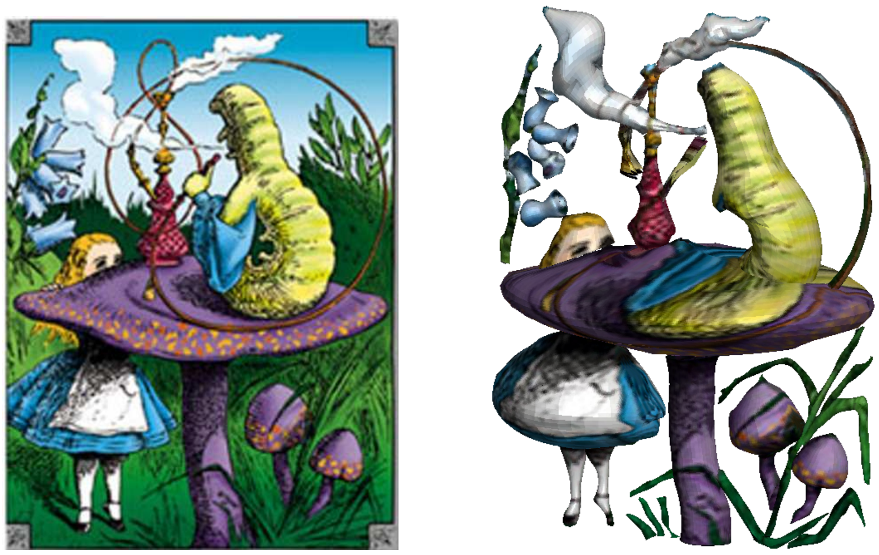

(c) An ensemble of models for 'Alice in Wonderland'

Figure 1.4: Silhouette-based single view reconstruction: Silhouette-based information can be used to create objects with holes (genus 2) such as the teapot (a) and those with concave silhouettes such as the ostrich (b). A collection of objects modelled very simply (foxgloves, hookah, caterpillar) can be used to create quick illustrations of storybook figures such as 'Alice in Wonderland' (c). (See fig 3.12 for more examples).

illustrations involve a lot of input, making the modelling framework less worthwhile. This is why the problem of single view reconstruction is relevant.

The goal of learning and using class-specific information is difficult to implement in practice without targetting a very specific problem or approach. Humans are naturally good at gleaning such information from the ample examples in the world around us. *E.g.* we find it easy to predict human pose and shape from an image silhouette, predict a golf ball's shape from its shading and texture *etc.* Thus, the use of class-based information is useful to create algorithms for guessing specific families of shapes well.

Human perception is prone to biases and faulty assumptions too, as is illustrated by numerous examples of popular visual illusions. Therefore, in addition to being inspired by the human ability at using class-specific information for this task, the real goal is also to outperform them eventually. Then the methods could be used for testing the veracity of human depth perception too.

1.4 Road Map

Chapter 2 reviews popular methods in the fields of single view reconstruction, surface representations, deformable object modelling and object detection and segmentation in our literature survey.

In Chapter 3, a method for single view reconstruction of objects from silhouette-based information is presented. The method can model a variety of topologies, self-occlusion and make use of image-specific information such as multi-local singularities, creases *etc.* as shown in figure 1.4. Our surface is a solution to a convex optimization problem where all the above information can be expressed as linear constraints.

Class-based single view reconstruction involves the learning of class-based information and then using it to predict the best 3D model for a given image. We are particularly interested in the first part of the problem and show how class-specific deformable shape models can be learnt from a collection of images in Chapter 4, as illustrated in figure 1.1.

When each object instance is a different class example, reliable point correspondences are hard to find. However, the correspondence between higher-level components such as curves (for a deformable lily petal class) can be used to learn deformation. The problem can be framed as a combination of regularized reprojection error where all variables—cameras, 3D shape model and correspondences—can be jointly learnt.

Automation is one of the goals of this work. In Chapter 5, user annotation is eliminated with the help of class-specific object detection and segmentation (see figure 1.2). Local image-based information is used to learn a classifier that differentiates between edges on an object class boundary from other less relevant ones. This learning can be used in state of the art methods for improved object detection and segmentation. Given a segmented object, methods from Chapters 3 and 4 can be used for automatic reconstruction.

Appendix A describes the user interface used to extract contours and annotate images.

1.5 Publications

Work from chapter 3 has appeared in Eurographics 05 [153] and also CVPR 06 [154]. The work in Chapter 4 is still under review. The work in Chapter 5 has appeared in ICVGIP 06 [155].

Chapter 2

Literature Survey

This thesis deals with the problem of object reconstruction from single views based on class-specific knowledge. The process of reconstruction involves: (i) applying class-specific knowledge, (ii) input processing (*e.g.* object detection and segmentation), and, (iii) finding a plausible 3D model for the given image. In this chapter we will take a quick tour of existing methods for different aspects of the problem.

Overview: We discuss the various subjects pertinent to this thesis in their order of relevance to the various chapters. First, the existing methods of single view reconstruction of objects in the presence of specific cues (contour, geometry, shading *etc.*) are introduced in § 2.1. This, along with the discussion of surface representations in § 2.2 is useful for a better understanding of chapter 3 where we use image-based and user-driven information to perform single view reconstruction from image silhouettes.

The goal of class-based single view reconstruction is associated with two problems: (i) Defining and learning a class model, which is discussed in § 2.3.1, and (ii) Deploying class models to reconstruct an object from a single view (discussed in § 2.3.2). In chapter 4, we introduce our method for learning deformable class-based models from image collections. To gain insight into this, methods for learning class-based models from multiple views are studied in § 2.4. We also use class information to automatically detect and segment class instances, to eliminate the need for user annotation and help in creating an end-to-end system for automatic single view reconstruction. This is described in chapter 5 and the various existing avenues of methods in this area are discussed in § 2.5.

2.1 Image-based and user-driven cues for single view object modelling

Even in the presence of multiple views, reconstruction is subject to ambiguity depending on the quality of available calibration, correspondences *etc.* Naturally, single view reconstruction is prone to much more ambiguity. In the absence of other information, an infinite number of models can lay claim to the creation of any one given image. Single view reconstruction methods using local cues such as shading, texture, geometry, silhouette *etc.* are prone to problems and therefore have to resort to various domain-specific assumptions, user-input, topology information and global constraints for disambiguation. Class-based learning can be used to provide such disambiguation cues in reconstruction. Object characteristics such as topology can be constrained to create more plausible 3D models. In this context we will now discuss the range of computer vision cues for deriving shape such as occluding contours, geometry, shading, texture, and sketch-based modelling, followed by a discussion of more global models such as those used in class-based reconstruction methods.

2.1.1 Shape from Contour

Koenderink [98] pointed out that the apparent curvature of the occluding contour is related to local surface curvature by the following relation: for any vantage point and without any restrictions on the shape of the rim[1], a convexity of the contour corresponds to a convex

[1]Rim: closed curve on the surface where the rays from the camera centre are tangent to the surface

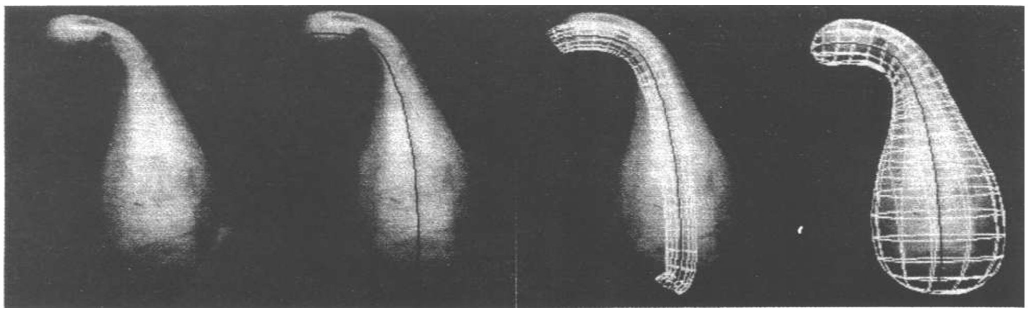


Figure 2.1: Figure from Terzopoulos *et al.* [191]: Given a squash image, user initializes spine. A deformable generalized cylinder is deformed iteratively to generate a symmetric smooth squash that correctly projects to the image.

patch on the surface and a concavity to a saddle shaped patch. Inflections of the contour correspond to the flexional curves (also called parabolic curves) of the surface. This important result cleared existing misconceptions at the time about the relation of image silhouettes to occluding contours. The initial proof by Koenderink was for smooth surfaces in Euclidean spaces. This was extended to perspective projection by Lazebnik and Ponce [108]. Koenderink's work was revisited by Barrow and Tenenbaum [7], Giblin and Weiss [72], Cipolla and Blake [30], among many others. [30] extended the relationship between the apparent contour and the Contour Generator for dynamic scenes with perspective projection, to provide useful qualitative and quantitative constraints on surface shape.

In as early as 1987, inspired by active contour models [96], Terzopoulos [191] (see figure 2.1) proposed the idea of retrieving deformable parametric surface models for objects of cylindrical topology from single monocular images. An approximate contour is initialized around the object. Image-based forces latch this contour to the object silhouette and simultaneously drive the spatial deformation of the corresponding 3D model, according to variational principles [69]. The model undergoes this deformation in an iterative manner and is forced to match its outline in the image as it develops. Symmetry is encouraged with the use of appropriate terms instead of forcing parametric shape family. In their optimization, a complex combination of non-linear functionals are used to model the image-based, axis aligning and model inflating forces. Optimizing the combination of these is difficult and involves using iterative numerical integration schemes. Being non-linear this optimization is sensitive to the initialization. A similar idea was used to find surfaces in three-dimensional images by Cohen and Cohen [31].

2.1.2 Shape from Geometric Information

As early as the 70s, there was interest in using geometric information for scene understanding. Waltz [202] studied the use of pre-annotated information in the form of line drawings in images, for scene understanding and shape recovery. Formalisms encoding all possible configurations of lines, junctions and surface intensities of adjacent regions were used in conjunction with the properties of inter-surface support and gravity to enumerate all possible scene configurations and eliminate implausible ones. Combined with human input and the use of shading and viewing position constraints, these rules were used to recover scene structure. In the 1990s such methods were augmented by the use of more geometric properties. Liebowitz *et al.* [115, 116, 118, 117] showed how simple properties of objects in single images, such as perpendicularity and parallelism of lines, can be used to derive the camera and subsequently metric reconstructions (up to scale) of piecewise planar objects in the scene. Sturm and Maybank [185] used similar geometric constraints to construct an interactive system capable of simultaneously calibrating and reconstructing a piecewise planar object from an image. This algorithm is framed as a least squares fitting problem between points and planes. Their method consists of an alternative iteration between fitting planes in 3D and back-projecting image-based information to intersect the model.

Criminisi *et al.* [43] (see figure 2.2) utilize similar geometric rules to reconstruct perspectively distorted images. However, they additionally use dimensions of known objects to derive dimensions of other objects using relative measurements thus creating Euclidean reconstructions of scenes and even hypothesize convincing models for renaissance paintings.

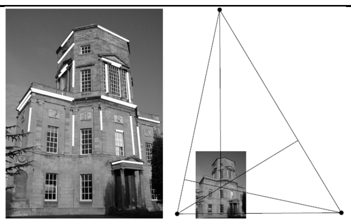
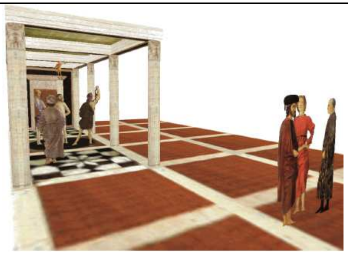
	
(a) from Liebowitz *et al.* [116]	(b) from Criminisi *et al.* [43]
	
(c) Image from [116]	(d) Reconstruction of (c) from [116]

Figure 2.2: (a,c,d) Geometric properties such as parallel and perpendicular lines can be used to compute vanishing points and then construct metric reconstructions of buildings. In the presence of objects of known dimensions, knowledge of relative sizes can be used to retrieve a near accurate euclidean reconstruction as seen in (b).

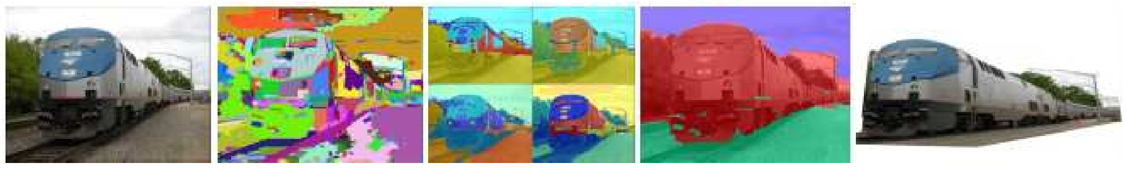


Figure 2.3: Image → superpixels → constellations → labelled based on models → 3D scene; from Hoiem *et al.* [87]

The above methods need user annotation and input. Hoiem *et al.* [87] (see figure 2.3) devise a method to remove this dependence and automatically produce texture-mapped pop-up models of piecewise planar scenes. The algorithm uses colour and texture statistics to group image pixels into superpixels and constellations. In addition to colour and texture, their method uses location, shape and geometry to infer geometric labels (like 'ground', 'sky' *etc.*) for constellations using models learnt from training images. These labels are then used to 'cut and fold' the image into a pop-up model. The automatic nature of this algorithm naturally means that good reconstruction is produced only for very simple images. However, the information used by this approach is largely based on constellation based methods ignoring the more geometric approaches preceding it.

The popular methods of producing piecewise planar reconstructions exploit the constraints provided by the join of different planes and constraints for creating pop-up models but fail to incorporate other cues in the form of occluding contours and more global class-based priors. Geometric information can be used to construct surfaces more generic than

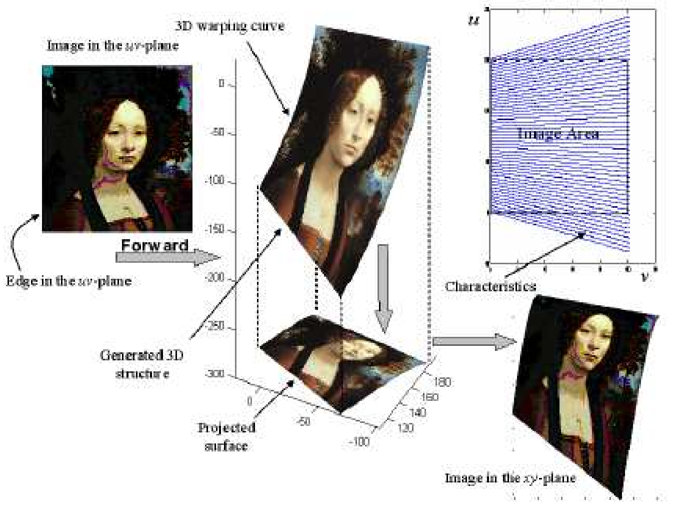


Figure 2.4: Given a plane sheet of paper, and a smooth 3-D open curve in Cartesian XYZ space, the paper is bent so that one edge conforms to the specified curve; from Gumerov *et al.* [83]:

just piecewise planar ones. Gumerov *et al.* [83] (see Fig. 2.4) show how the boundary of applicable surfaces[2] can be exploited to fully recover geometric structure from a single image. Given additionally some 3D correspondences of points in the image, or the 3D mapping for part of the boundary of the surface, a set of nonlinear higher order partial differential equations can be solved to compute a surface. Salzmann *et al.* [166] show similar results for inelastic surfaces. In the presence of much more information such as camera calibration and correspondences, they show that this can be solved in closed-form.

[2]Applicable surface: isometric surfaces with vanishing Gaussian curvature. These surfaces satisfy certain families of partial differential equations. Paper is an example of an applicable surface.

These theoretical and practical results enable us to easily reconstruct a variety of primitives–paper, stiff leaves *etc.* assuming the image can be appropriately parsed, paving the way for object specific reconstruction.

2.1.3 Sketch-based Modelling

Completely automatic modelling is ambitious and prone to error. Image-based cues can be replaced by user-drawn forms of input. Sketch-based modelling methods, use intuitive and approximate user input for rapid prototyping of objects and are important in graphics and animation.

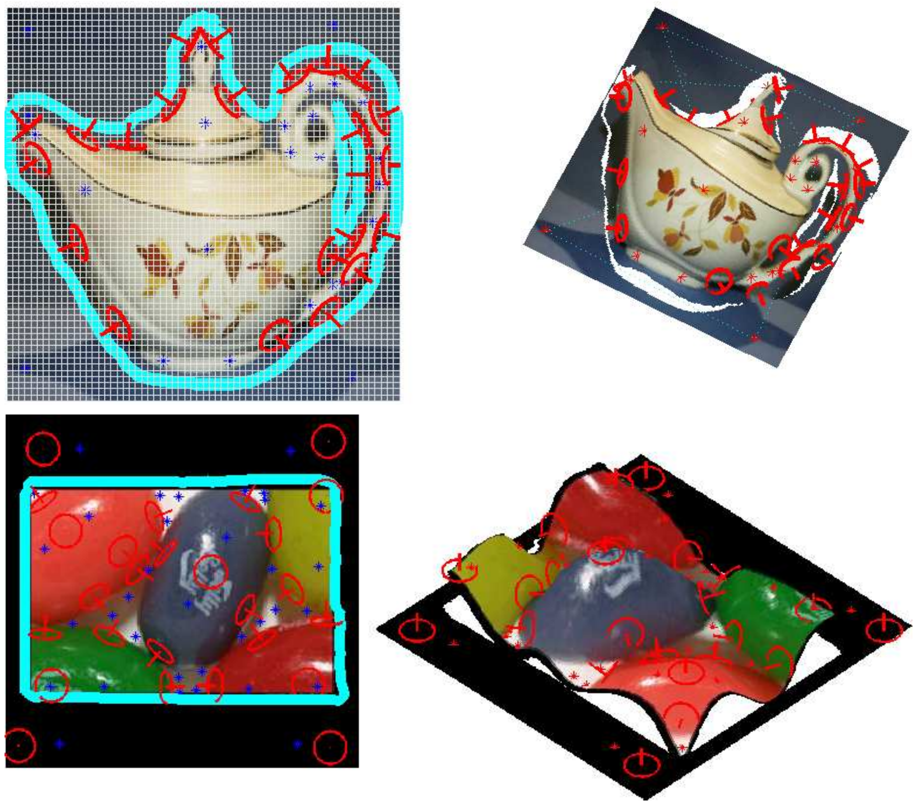


Figure 2.5: My implementation of [211] is used to reconstruct a teapot and jelly beans (compare with our improvement in chapter 3, fig 3.12). Insertion of constraints by user is shown on the left. The constraints are used to create a monge-patch representation of the teapot in 3D on the right. Modelling even simple jelly beans requires considerable effort.

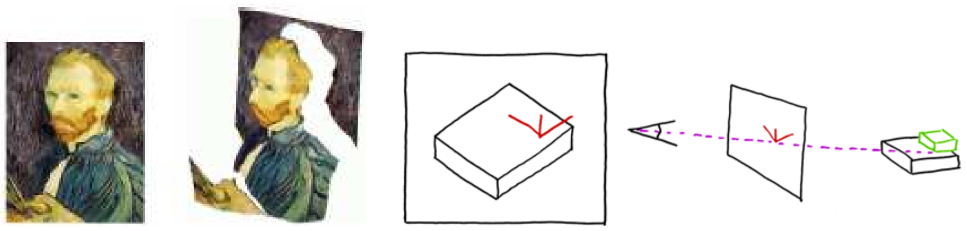

(a) (b)

Figure 2.6: (a) Zhang *et al.* [211]: 3D modelling a Van Gogh painting using Zhang's technique. (b) SKETCH [210]: A series of strokes is drawn in the film plane in red (left). The salient vertex is projected into the scene thus defining the placement of new geometry (green).

Zhang *et al.* [211] (see Fig. 2.6) proposed an approach for reconstructing high quality, free-form, texture-mapped, $2\frac{1}{2}$D scene models (Monge patches [79]) from a single image with arbitrary reflectance properties. It uses a set of user-specified inputs in the form of positions and normals (seen in figure 2.5) to create a smooth 3D surface (by minimizing a convex objective function) which satisfies these constraints. The use of hierarchical transformation with adaptive resolution as prescribed by Szeliski [186] improves the computational speed. The technique is interactive and allows fast reconstruction. The speciality of this system is the use of linear constraints and a convex objective function, which inspires us greatly. The downside is that they construct a $2\frac{1}{2}$D representation of the scene.

SKETCH [210] was an important milestone. This gestural environment is used for rapidly conceptualizing and editing approximate 3D scenes with minimal effort. The system allows the user to create and transform objects (these are mostly made up of axis-aligned line drawings) in a constrained way, thus eliminating the need for complex constrained optimization for generation of models. The system stores the semantics of gestures and relationships between objects in the world; such as positional relationship: The painting is *on* a wall, therefore it must always be translated along the wall. With such relationships, it is easy to build and modify a scene. Though simplistic in its approach and object classes, this work was an inspiration for a range of subsequent work.

Following SKETCH, Igarashi *et al.* [89] (see figure 2.7) came up with the sketching

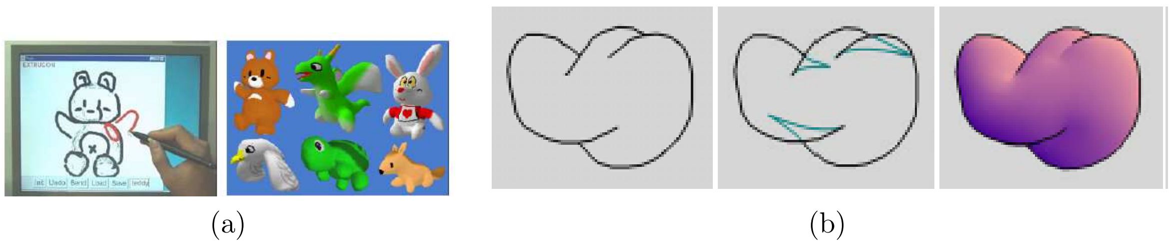


Figure 2.7: (a) Igarashi *et al.* [89]: The user drawing and 3D models produced by *Teddy*. (b) SmoothSketch [95]: extends the the idea. Visible contours of a shape are used to infer the hidden contours, including hidden cusps, and then create a fairly smooth 3D shape matching those contours.

interface *Teddy*, for quick and easy design of freeform "rotund" models represented by polygonal meshes. User-drawn 2D freeform strokes are used to construct a 3D polygonal surface: it inflates the silhouette using silhouette-medial axis distances and constructs an object of spherical topology. This is a good system for rapid prototyping. It has gestural operations allowing for complex editing of an object. With simple strokes the user can extend or scoop out parts of the object. The ad-hoc inflation process is designed for simple silhouettes and complex topologies cannot be constructed.

Karpenko *et al.* [94] modified *Teddy*'s polygonal surface representation to variational implicit surfaces and implement several new user-interaction elements to simplify the modelling of interesting hierarchies. This representation of smoother surfaces has the side-effect of rounding off sharp corners. Karpenko *et al.* [95] overcame the topological limitations of *Teddy* by proposing SmoothSketch, a method that can handle complex surfaces (see figure 2.7). Importantly, it allows the user to express complex notions such as cusps and T-junctions and infers the correct parts of the hidden contour from incomplete user-drawn silhouettes.

Sketch-based methods eliminate certain issues from the reconstruction process such as reliability in feature detection. Also, there is scope to construct effective user-input environments to ensure maximal output for minimal effort. In the absence of accurate automatic systems, effective sketch-based input mechanisms are important for the purpose of single view reconstruction.

2.1.4 Shape from Shading (SFS)

Recovering shape from shading involves the use of varying shading on an object's surface to solve for its shape. By studying how an image is formed, we learn how to trace our way back to the shapes which gave rise to them. Most methods of SFS assume Lambertian reflection models *i.e.* gray level at a pixel depends on the light direction and the surface normal without being affected by inter-reflections. Similarly the light is assumed to be a point source. Despite simplistic assumptions, the problem of reconstruction is difficult because for a given gray value at each pixel, surface orientation must be determined. The solution is under-constrained and ambiguous. Horn *et al.* [88] discuss how to ascertain the plausibility of the shading in an image, before trying to recover shape from it. The SFS problem can be expressed as an energy minimization problem with the following constraints: (1) Brightness constraint: the reconstructed shape should produce the same brightness as the input image at each surface point, and (2) Smoothness constraint: ensures the recovered surface is smooth and overcome problems due to noise. The shape at the occluding boundary is given for initialization. Brooks and Horn [26] minimized this energy function, in terms of the surface normal using Variational Calculus. Frankot and Chellappa [63] enforced integrability to this work, in order to recover integrable surfaces. Surface slope estimates from the iterative scheme were expressed in terms of a linear combination of a finite set of Fourier basis functions. Their results showed improvements in both accuracy and efficiency over Brooks' algorithm. Szeliski sped it up using a hierarchical basis pre-conditioned conjugate gradient descent algorithm in [187]. Alternative to the smoothness constraint, Zheng and Chellappa [213] introduced the concept of intensity gradient constraint, which specifies that the intensity gradients of the reconstruction and input image should be similar. This is easier to implement, with no special requirements and converges more quickly.

In [148], Pentland used the linear approximation of the reflectance function in terms of the surface gradient, and applied a Fourier transform to the linear function to get a closed form solution for the depth at each point. The linear approximation can cause trouble, when there are large non-linear terms.

It is useful to use more than one cue at a time, so that the one can help disambiguate the results from the other. Shimshoni and Ponce [177] take on the problem of deciding whether a given 2D line drawing comes from a valid polyhedron. Given an accurate reflectance model for the surface (such as Lambertian), they fuse the problem of SFS with the constraints of projected polyhedra to verify their silhouettes. Using the cues together is a good idea, but the presence of local minima necessitates good initialization.

Despite many intelligent attempts, the problem is that the large number of assumptions made by SFS techniques do not hold: *e.g.* surfaces are never really Lambertian, there are multiple light sources and the methods are very sensitive to noise. Texture, despite also being a local cue, has been a comparatively more effective cue to exploit for surface recovery.

2.1.5 Shape from Texture (SFT)

Gibson [73, 74] was one of the first to propose psychological theories about how the human perception relies on gradients of features, stressing on the importance of gradient of texture on receding planes. Given the frontal view of a texture element, its projected view from the surface of an object, can be used to estimate surface shape. Witkin [205] pointed out that the distorting effects of projection must be distinguished from texture properties themselves and proposed a method for reconstruction of planar and curved surfaces. He assumes orthographic projection and uniformity of texture in all directions (isotropy). Kanatani [93] added an assumption of homogeneity of texture, *i.e.* constant number of texels per unit area on the surface. However he showed how planar and curved surfaces can be recovered under perspective projection using geometric properties of the surface.

Later, Forsyth [61] constructed a maximum a posteriori estimate of surface coefficients

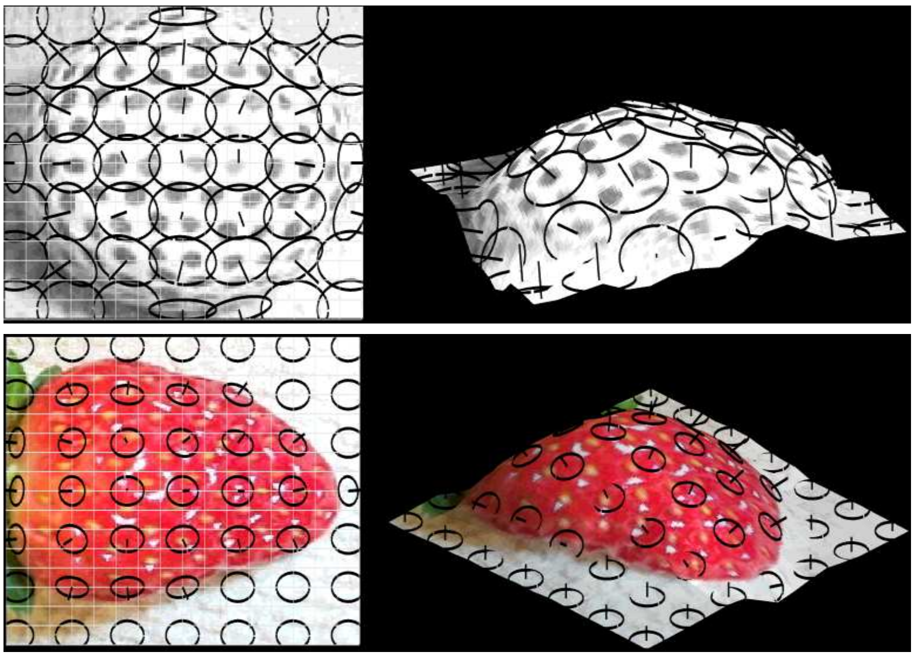


Figure 2.8: My implementation of [120] is used to reconstruct a golf ball and a strawberry using normals derived from texture.

using the deformation of individual texture elements. This method does not need any information about the boundary of the observed surface or the distribution of elements. An inhomogeneous, marked Poisson point process is used to model the surface texture. For generic orthographic view and texture, each texture element yields the surface gradient unique up to a two-fold ambiguity. Each image texel can be expressed as an affine transform (with a degree of freedom = 3) of the model texel. By assuming one of the texels as a model texel, the transforms for the rest can be found. In order to recover the surface, the texture imaging transformation is found at each of a set of scattered points up to a sign ambiguity. Using priors over the surface surface with an Expectation-Maximization like formulation, the ambiguity of each texture element's transformation is removed, to yield a smooth surface. Loh *et al.* [120] use a similar method to extend this for perspective projection. Without any assumptions about the texture, they show how the frontal texel can be estimated. A search is conducted over all possible frontal texels with imposed surface smoothness, to estimate a unique, consistent estimate of frontal texel and hence a consistent surface shape (see figure 2.8). White and Forsyth [203] show that ambiguities in estimation of texture normals can be broken using shading cues.

SFT methods have been more effective than SFS methods in general. There is however the need for global models and other cues such as occluding contour. In general, research in SFS/SFT methods have taken a backseat in recent times in favour of more class-specific methods with global models and probabilistic treatments.

2.2 Surface representation

How a 3D object surface should be represented is an important question for the purposes of computing and displaying the 3D object. There are several ways of grouping the forms of surface representation: analytic *vs.* numerical, explicit *vs.* implicit, and so on. Depending on the nature of the task and the limitations different surface representations may be desirable.

Piecewise planar surfaces: Polygon meshes are one of the most popular methods of surface representation. A smooth object surface is approximated by a discrete mesh of polygons (usually triangles). A list of ordered vertices and faces represent the object. Objects of arbitrary topology can be represented. The resolution of the mesh is often decided by bounding the error in representation of the actual surface. The resolution may be varied depending on surface complexity enabling more polygons in high curvature areas. Alternatively, uniform resolution across the surface may be desired. Geometry images [82] represent an irregular surface mesh by reparametrizing its characteristics such as the x, y, z coordinates, normals *etc.* as the the r, g, b values of an equivalent regular grid-based image representation. By finding an ideal cut and a parametrization, image-based compression techniques can be used to store a model more efficiently without losing important surface detail (an example is seen in figure 2.9). Availability of computing resources have made it

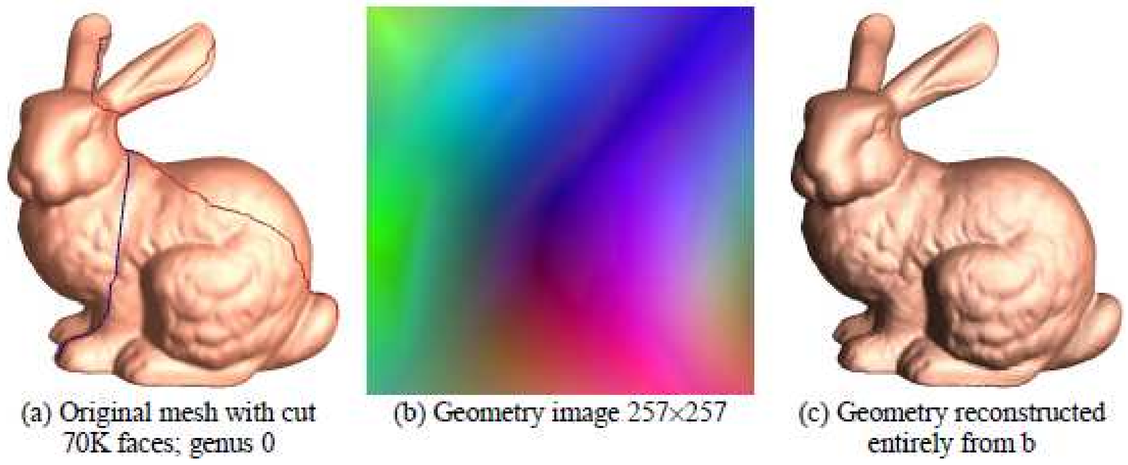

(a) Original mesh with cut 70K faces; genus 0
(b) Geometry image 257×257
(c) Geometry reconstructed entirely from b

Figure 2.9: Geometry images [82]: The high resolution 3D bunny (with 70K faces) can be reparametrized by a mere 127×127 geometry image. The resulting 3D model approximates the actual 3D bunny with reasonable accuracy.

possible to store arbitrarily high resolution models hierarchically that can be rendered and edited at adaptive resolutions [47, 109].

Measures such as normals, second derivatives and curvatures are important for various tasks, such as representation, shading *etc.*, and can only be approximately computed for such methods. Effective methods for creating, editing and fusing mesh-based models make it popular with modelling methods such as *Teddy* [89] (see figure 2.7). Alternatively, many modelling methods use other surface representations: parametric, implicit and point clouds, for actual construction of an object. For final representation and rendering, all other representations can be converted to meshes and *vice versa* ([58, 102]).

Monge patches: The surface may be treated as a depth map which depends on the x, y values of the surface [79]. Such an explicit surface has the functional form $z = f(x, y)$. Such $2\frac{1}{2}$D surfaces are not fully freeform. *E.g.* a Monge patch cannot intersect a ray perpendicular to the x, y plane more than once, nor can it be tangential, as seen in figure 2.10. This representation is simple to use and has been the basis of work such as Zhang *et al.* [211] (see figures 2.5,2.6) and various work on retrieving shape from shading.

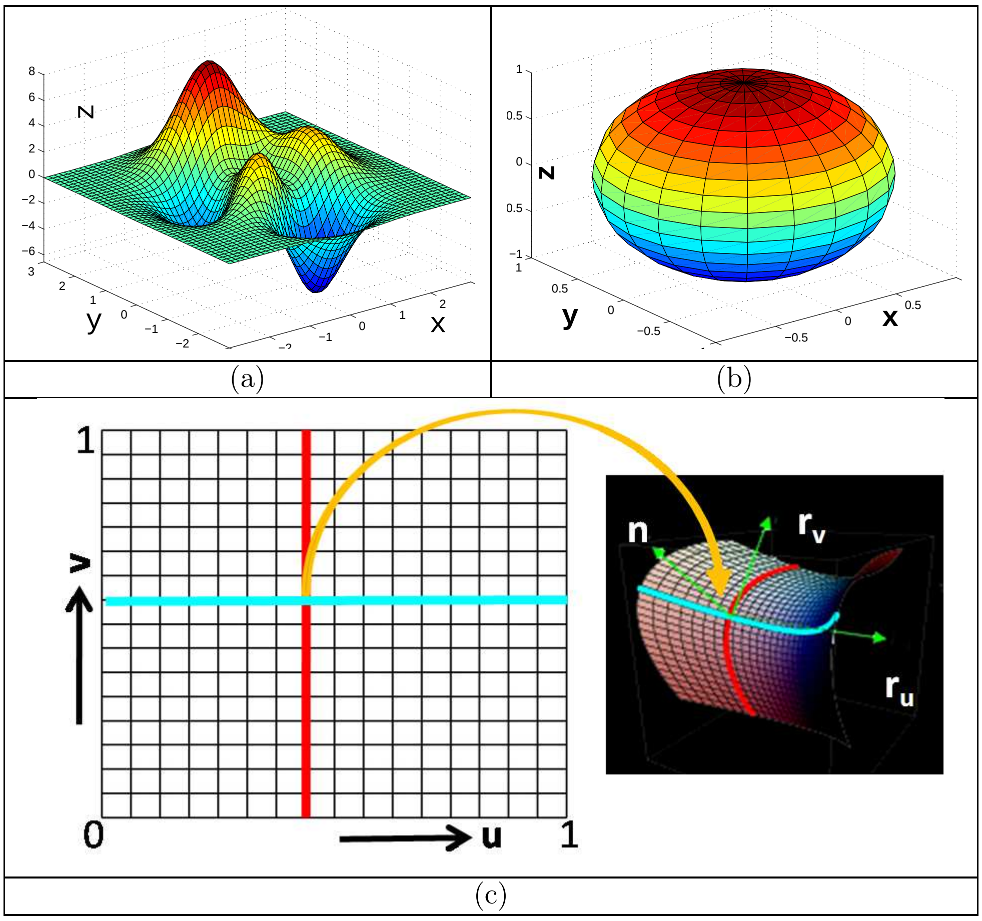


Figure 2.10: (a) can be represented as a monge patch but (b) cannot. (b,c) can however, be expressed as parametric surfaces. Each point on the parameter space (u,v) is associated with a position on the surface $\mathbf{r}(u, v)$. The normal at $\mathbf{r}$ is given by $\mathbf{n}$ and the derivatives along u and v are given by $\mathbf{r}_u, \mathbf{r}_v$

Geometric primitives For objects whose topological nature is known geometric primitives may be used for effective representation. *E.g.* topologically cylindrical objects can be represented by generalized cylinders. They are defined by the medial axis, and discs along the axis length and normal to it, defining the surface around it. These have been popular for defining axially symmetric objects. Terzopoulos *et al.* [191] (figure 2.1) use deformable generalized cylinders to model simple objects from monocular views.

Volumetric primitives such as superquadrics [6, 147] have been used for representing relatively simple, closed shapes. A 3D function with 6 degrees of freedom, it can be expressed in implicit or parametric form. Deformations such as tapering, twisting and bending can be added to represent more complex objects. Despite this flexibility, the modelling power of methods such as generalized cylinders and superquadrics is limited. Terzopoulos and Metaxas [190] add a local deformation field simulating forces and user-interaction to enable representation of a wider range of models. Their surface was effectively parametric, with a superquadric regularizer. In this scheme, complex topology would require representation as a set of multiple superquadrics, somewhat like an articulated model.

An interesting surface representation is that of ribbons [161]. They are plane shapes obtained by sweeping a geometric figure (generator) along a plane curve (axis, or spine). Blum, Brady and Brooks are three popular types of ribbons. Blum ribbons are obtained by sweeping a disc; Brooks ribbons are obtained by sweeping a line segment making a constant angle with the spine, while Brady ribbons are obtained by sweeping a line segment whose extremities form a local symmetry (see figure 2.11). Each ribbon can only represent a limited family of shapes, *e.g.* the Blum ribbon is equivalent to a generalized cylinder. However, each type of ribbon (see [161]) has important properties that are important for its generation and recovery from images. Two points on the boundary of a Blum or a Brady ribbon which correspond to the same axis point form a local symmetry. Such ribbon pairs can be found by testing all possible pairs of contour points for local symmetry, and these ribbons pairs can in turn be grouped into ribbons. Blum and Brady ribbons can be segmented using local symmetries (see Brady and Asada [21]). Ponce [151] proved several properties of ribbons and showed that for the specific skew-symmetric class of Brooks ribbons, contour curvature could be used as a local signature to characterize point pairs. This is used to find (using methods from [140]), segment and even recover 3D structure for this class of ribbons.

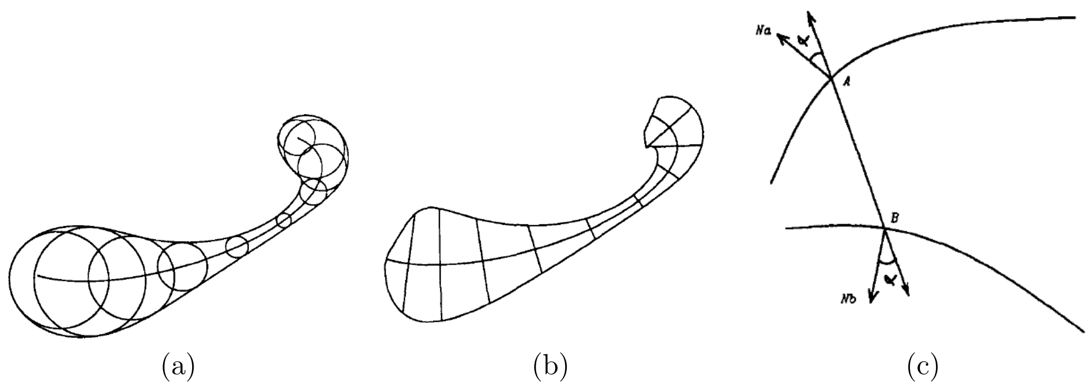


Figure 2.11: Ribbons: Examples from Ponce [151]. (a) Blum ribbon: generated by sweeping a disc along a plane curve. (b) A Brooks ribbon is generated by sweeping a line segment along a plane curve. The angle between this line segment and the tangent to the curve is constant. (c) A Brady ribbon is defined by local symmetries

Parametric surface: A surface can be represented as a function of a set of parameters which adequately represent the surface dimensionality. For a 2D manifold, surfaces can be made truly freeform by the following parametric representation $[x, y, z]_{u,v} = \mathbf{f}(u, v)$. Each point on the parametric grid represents a unique point on the surface and *vice versa.* One of the advantages of parametric surfaces is the ease of evaluating measures such as position, tangency and curvature at points along the surface, and ease of traversal along the manifold. This representation has been used in a variety of work such as [191, 154]. Parametric surfaces are not capable of handling arbitrary surfaces and topologies without the help of other constraints and modifications, but work such as [154] shows how this can be done.

Non-uniform rational B-spline (NURB) surfaces and Bezier surfaces are special cases of parametric representation where a surface is represented by a set of piecewise smooth polynomial patches [59, 51]. Compared to polygonal meshes, each patch can be curved and is a weighted combination of basis functions of a specific order (*e.g.* cubic splines) and therefore a surface can be represented with higher accuracy and reduced storage. Measures such as normals, curvature *etc.*, can be more accurately computed making further processing such as shading easier.

Implicit surfaces: The relationship between the different coordinates of the object surface can be expressed as a function $f(x, y, z) = c$. In implicit surfaces, it is not possible to extract one coordinate as a closed-form function of the others. Therefore, the function defines a potential field and the surface is the locus of a value in the potential field (given by *e.g.* $c = 0$). For a closed surface all points on the inside of the surface may have $c < 0$ while points on the outside have $c > 0$. This easy representation of an object's interior is one of the advantages of this representation. This representation has been used by work such as [181, 94] and is capable of representing variable topology easily. For example, varying c can result in smooth topological variation. Surface editing is also made easy, *e.g.* fusing two objects is equivalent to summing their potentials. Similarly, collision detection is easier. The implicit function itself may be defined either by the analytic function shown here, or by discrete samples and other procedural methods. For the purpose of visualization the potential field is used to find a polygonal output (using *e.g.* the marching cubes algorithm [121], or subdivision based algorithms [131]). Rendering can also be performed without intermediate surface representation with the use of methods such as ray tracing. Bloomenthal [16] provides a comprehensive survey of methods in implicit modelling.

2.3 Class-based Modelling

So far we have surveyed the range of single view reconstruction strategies depending on local cues. To tackle ambiguities and noise, they sometimes make use of a global smoothness prior [120, 203, 191]. In some examples, even scene constraints, such as support between interacting objects and with the ground plane, have been used to effectively constrain reconstruction [43, 87]. Despite such measures most methods we have discussed so far are largely local and the obvious need for a higher level of class-based understanding has been highlighted many times. In this context, we will discuss the role of class-based models for shape recovery.

2.3.1 Shape models

The simplest model of an object is a perfect sample denoted by a constant surface, *e.g.* the mean or mode. This would be the compared with all new instances and their similarity could be measured according to some distance measure. However, a model like this only allows limited variation from the given sample—equal penalty for any dimension mismatch– and therefore is often an incorrect model for variation. Therefore, more sophisticated models representing plausible object variations are needed. A broad survey of existing methods can be found in [33].

Hand-crafted models: Flexible models can be built from simple components–circles, lines, arcs *etc*. The parts can move around relatively and undergo variations in scale, orientation *etc*. Yuille *et al*. [209] model facial parts using parametrized circles and arc. An approximate fit of the model to an image is refined by changing parts of the model. Black and Yacoob [14] use a similar model for the purpose of tracking and recognizing human expression. Such models capture detail for well known and expected shapes, but lack generality and require a completely new manually-specified model for every application. More recently, there has been work on modelling specific classes of objects like trees [142] and flowers [90] (see Fig. 2.12). These methods use the intrinsic properties of flowers and trees, to make the process of user-input and modelling easier.

Articulated models: are an effective way of representing a deformable model by sliding or rotating joints connecting piecewise rigid components. Beinglass and Wolfson [10] showed how articulated objects based on automatically detected interest points could be used for object recognition. With the use of the Generalized Hough Transform, they show how projective transformation can be handled. Connected sub-parts vote for each joint they're connected at, and the method attempts to handle occlusion too. Grimson and Lozano-Pérez [81] also worked on similar lines by examining hypotheses on the basis of local measurements (using his "interpretation tree" approach) while the articulated models

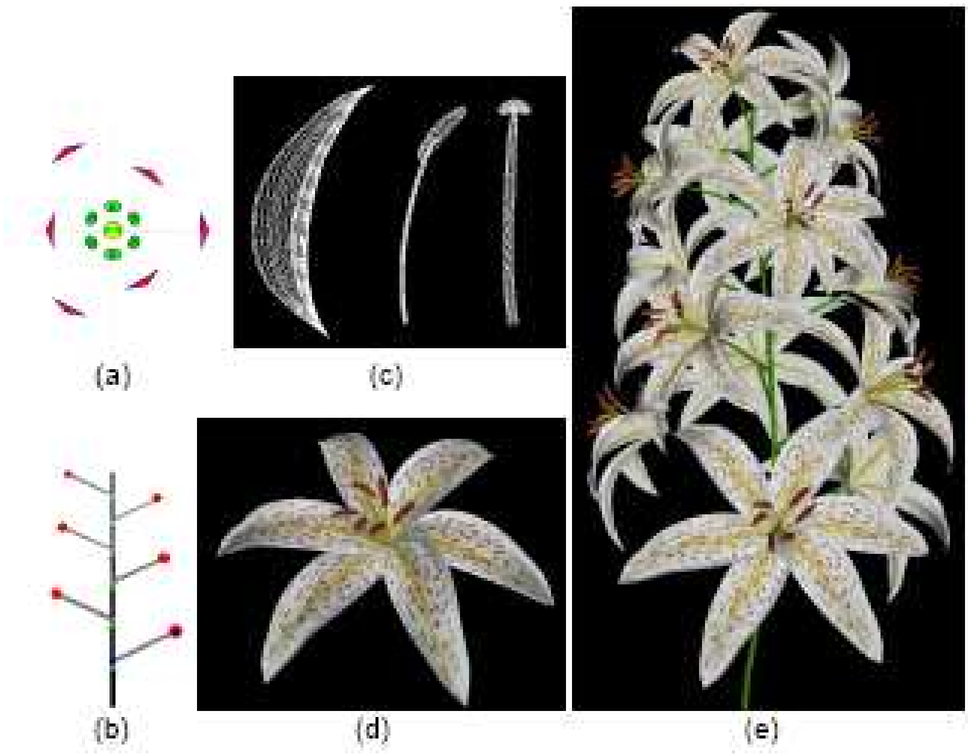


Figure 2.12: Ijiri *et al.* [90]: The structural information of a lily is used to design an easy system for user-based sketching and editing.

themselves are in the form of polyhedra (therefore restricted in shape). In modern times, articulated models have been used in Pictorial Structures [52, 56] for object detection and recognition.

Fourier shape models: Shapes are often represented over some low-dimensional subspace of bases. These bases can be the trigonometric functions that form the backbone for Fourier expansion. By varying the parameters and the number of terms used, different shapes can be generated and the complexity of the model controlled. Staib and Duncan [184] (also see Scott [174]), recognize the need for global shape models as opposed to the existing local techniques in use. They replaced traditional parametric models of limited power such as cylinders, polynomials, superquadrics etc. with parametric Fourier shape models. There are some knobs—*e.g.* curve complexity can be controlled by coefficient choices. Though such models can be proved equivalent to other statistical models, it is harder to encode shape variations to reflect in the parameters of the trigonometric expansion terms. Open boundaries pose one such problem. Another classic is the difficulty of approximating sharp features with a limited number of terms.

Finite Element models: Finite element methods provide another method for modelling objects as physical entities with internal stiffness and elasticity. Similar to Fourier transforms, the shapes are represented as low-order frequency displacement eigenvectors corresponding to vibration modes of the object (see Pentland and Sclaroff [146, 145], Terzopoulos *et al.* [190]).

Local models of shape: Classes such as cloth are not stiff enough to be solved with method such as [166, 83] and yet are not simple enough to be modelled by linear PCA based models. In different attempt, Salzmann *et al.* [167] learn deformation models for local patches instead of attempting to learn global models: the space of deformations of smaller parts is smaller, making it easier to learn. The idea is to combine this prior over the collection of patches forming a surface to retrieve a valid shape hypothesis. They are forced to approximate their optimization at various levels, leading to a less than ideal implementation of an ambitious formulation.

Statistical models of shape: Point Distribution Models (PDMs) [37] propose that any class deformation can be expressed by a mean and modes of variation. Shapes are typically represented by a set of representative 'landmark' points. Different shapes are aligned with each other using algorithms such as Procrustes alignment [76], or Iterative Closest Point algorithm [212]. A statistical method (such as PCA) is then used to extract a mean and modes of variation representative of this class of objects from given samples (see [198, 139]). PCA techniques are equivalent to fitting a Gaussian to the exemplar data. A limitation of linear PDMs is that non-linear variations must be approximated by combining linear variations, which sometimes results in a non-optimal model. Also linear variations of the bases may produce implausible object shapes or be unable to produce valid ones because of the inaccuracy of linear models. Bregler and Omohundro [24] attempt to improve the specificity of PDMs by deriving multiple subspaces from different clusters of training data and constraining potential shapes to lie within the intersection of the potential subspaces. Heap and Hogg [86] alternatively suggest using a two-level hierarchical implementation in shape space to improve specificity, where the lower level accommodates sub-parts and their individual PDMs.

Heap and Hogg [85] offer a clever workaround by transforming the space in which the shapes are represented to account for the non-linearity. By transforming the Cartesian coordinates for non-linear components of hand models to lie in a polar space, they are able to model the nature of deformations more accurately. A polynomial regression generalization of PDMs by Sozou *et al.* [183] allows landmark points follow polynomial paths (as opposed to linear) with variation in shape parameters and thus capture more non-linearity. Subsequently they [182] also use a multi-layer perceptron to perform non-linear PCA which is more effective in capturing non-linear variability. Inspired by Mika *et al.* [132], Romdhani *et al.* [159] use a Kernel PCA approach to handle non-linearity in the presence of pose constraints for reconstruction with multiple views. The optimization between a candidate model and the target is now conducted in the feature space by using the kernel trick.

Cootes and Taylor [40] approximate the density estimate of the point samples (each point is a shape sample) by a mixture of Gaussians. Though the model continues to be linear, they are able to model complex variations in shape. Additionally they achieve higher specificity by being able to more effectively specify invalid regions between valid regions. Cootes *et al.* [39] extends this to include linear parametric models of shape variation which now define a full diffeomorphic deformation field (similar to Bookstein [17]). Landmark finding can be integrated into the process of constructing the warp field in each image thus eliminating the need for explicitly specifying landmark points.

2.3.2 Fitting models from image cues

Given a shape model, it must then be fitted to data by performing parameter estimation. The problem can involve fitting a test shape in 3D, or, fitting an object in 3D from its silhouette in 2D.

Fitting using contours (shapes) The development of active contour models helped pave the way for more principled class-based approaches to the problem of contour-based shape fitting.Active shape models [37] are associated with techniques of fitting parameters of a learnt shape distribution to an image to retrieve a plausible shape reconstruction. This can be done by minimizing the distance of a 3D candidate to either a test shape, or its projection in the form of a test image silhouette. There are a large number of unknowns: the correspondences, camera and pose parameters, and, the shape (and if available, appearance) parameters. Because of the large number of unknowns involved an iterative method of fitting is adopted. Given a PDM, Cootes and Taylor [37, 34] devise an iterative method of (i) fitting the best 3D model given correspondences on a test image contour, and, (ii) refining the correspondences by searching in the vicinity of the current projected 2D model. Instead of treating each projected landmark point equally, local statistical models are built along the normals to each landmark point in the projected 2D models. This helps to improve the search for the optimal model during fitting. Cootes and Taylor [38] also proposed the use of a multi-resolution technique for locating image structure to make the search less susceptible to local minima.

There has been research beyond the field of active contour models. Effects of 3D variation and deformation have been approximated by a wide variety of 2D methods in object recognition. Gavrila [67] proposed the use of hierarchical template trees to handle class-specific shape variation, though the generalization in shape attained by such methods is limited to the template bank. Ferrari *et al.* [55] use contour segments which can be relatively displaced to handle class-specific shape variation. Shotton *et al.* [178] recognize the importance of contour parts in the recognition of object parts in order to identify instances of deformable objects. Fergus *et al.* [54] introduce the 'constellation' of parts idea in recognition where the multi-part shape and appearance are learnt simultaneously while additionally accounting for relative scales of parts. The distribution of the parts is jointly Gaussian. Fischler and Elschlager [56]and subsequently Felzenswalb and Huttenlocher [52] propose a similar but exactly optimizable framework for representing, sampling from, and learning multi-part objects. Their parts now form a tree structured graph. This was extended to work on k-fan graphs by Crandall *et al.* [42].

Most of the above methods deal with fitting 2D structures (or projections of known 3D structures) on 2D image information, but it is much harder to derive the 3D structure corresponding to 2D information. However, the principle of ASMs is easily extensible to the 3D reconstruction problem, as shown in the reconstruction of faces from occluding contours by Keller *et al.* [97]. When the correspondences between the landmark points on the 3D points are known for the 2D image, fitting the shape model is easy. The level of difficulty increases progressively when this is not known, or when the object is occluded and the scene is cluttered. Also if frontal views are not assumed, then unknown camera parameters complicate the optimization further.

Fitting using appearance Active appearance models by Cootes and Taylor [35] propose the use of appearance based cues in addition to shape cues for the purpose of deformable model representation and fitting. Unlike shape, appearance is more susceptible to effects such as illumination. However there is access to more information, helping tackle occlusion and clutter better. As a direct consequence of capturing appearance, AAMs facilitate the sampling of entire images instead of being limited to evaluate likelihoods of contours. During training the effect of varying parameters to the error between hypothesized and actual appearance is learnt. For effective fitting, residuals should be used accurately to update parameters. AAM fitting is susceptible to local minima because of the complexity in the problem. Cootes and Taylor [36] furthered their original work to allow constraints (such as those inserted by a user) to be inserted for more effective fitting. Lanitis *et al.* [106, 107] proposed a compact parametrized model of facial appearance allowing parameters to be recovered more easily.

Many of the above ideas were originally proposed for 2D deformable models. Cootes *et al.* discuss problem of 3D modelling by treating shape reconstruction solely in terms of novel view synthesis, bypassing the actual 3D reconstruction. A 2D AAM is maintained for each training view. At test time, the nearest AAM (in terms of pose) is chosen to fit the given image.

In the context of this thesis, 3D deformable model fitting to 2D images is relevant. Many of the ideas proposed above can be directly extended for building 3D models. An extension of the basic AAM method was demonstrated by Blanz, Vetter and Romdhani [15, 160] in their work on single view face reconstruction. They fit a 3D statistical model of shape and texture to best fit the appearance and shape in a 2D image and power this by more effective parameter fitting methods. Solem and Kahl [181] extend the ASM/AAM techniques for reconstruction using level-set techniques. They fit multi-stage shape and appearance models like ASMs but they target richer information in faces such as eyebrow contours *etc.* This naturally involves more annotation but results in smaller surface representations, computation times and robustness to illumination effects such as specularities. The surface is fit to inferred 3D features and constrained fitting of surface models results in regularized surfaces.

There has been a vast variety of methods in class-based model shape and appearance fitting. There are many challenges such as representing and fitting models of varying number of variables such as a 3D shape which has variable number of vertices, or combined optimization of all parameters (such as simultaneously determining correspondences and reconstruction), coming up with better model representations, *etc.*

2.4 Using multiple views for class-based reconstruction

Rigid structure from point matching The first solution to this problem assumes known correspondences across an image collection and a simplistic orthographic projection model (see Tomasi and Kanade [193]). By exploiting the rank constraints on the generative process behind the images, the solution can be found by a simple matrix factorization of the tracking matrix. Any factorization of the form $product = factor1 * factor2$ is vulnerable to ambiguities such that, $product = factor1 * \lambda * \lambda^{-1} * factor2$. In the case of orthographic projection this ambiguity is resolved relatively simply. Several variations and flavours of this original method have been proposed. Iterative methods have been put together to extend this result to para-perspective [150] and fully perspective [188] cameras to obtain reasonably good local solutions if initialized well by affine systems. Even an online version, with every image updating the factorization in a relatively quick step instead of an expensive global factorization, was proposed by Morita and Kanade [135]. In order to accommodate several rigid objects, Costeira and Kanade [41] relax the rigidity constraint and use a permutation based method to group the tracking matrix into submatrices, each corresponding to an object. Beardsley *et al.* [9] note that most methods for projective structure recovery operate in a batch mode, thus making computations expensive. They propose a sequential method instead that recovers structure as each image is captured. For path planning operations, such as those required in robot navigation, this is useful.

Ideas for efficient calibration from unordered image sets put forth by Schaffalitzky and Zisserman [170] were put together in systems such as Photosynth [180]. Photosynth shows how several views of the same rigid structure (e.g. “Pantheon”, “Half-dome”) can be interrelated via the common coordinate system of a 3D reconstruction. Advanced methods for tracking objects in scenes [4] have further enabled us to perform rigid structure from motion (SFM) in video sequences effectively. In order to recover metric reconstruction, some absolute measurements from the real world must be used.

Rigid structure from curve matching In the absence of surface correspondences, then, how may the problem be approached? One solution is to move from zero-dimensional point matching to one-dimensional curve or line matching. Lazebnik and Ponce [108] point out that more complex geometric entities such as curve correspondences provide much richer information about 3D shape than point correspondences. The difficulty with curve correspondences, however, is a variant of the aperture problem: although the curve is in correspondence as a whole, individual points on the curve are not naturally in correspondence. Measures such as curvature provide poor matching constraints as curvature can change drastically under projection (consider a smooth helical segment which is imaged with a cusp). Projective concepts such as bi-tangency do provide extra correspondences, but again these are few. Furukawa *et al.* [66] address the problem of estimating rigid structure and motion from the apparent contours across successive image frames. A RANSAC-based voting approach is used to identify a set of 'frontier points' or correspondences across images, and simultaneously estimate camera configurations.

Existing work on curve matching uses the constraints associated with rigid-body assumption to constrain the matching. Schmid and Zisserman [171] showed how the use of 2 and 3 view matching tensors allows correspondence transfer: a point on one curve which is not parallel to an epipolar line induces a point match on the corresponding curve in a second view, and constrains the local matching homography allowing surface texture adjacent to the curve match to resolve ambiguities. Rigid curve matching in multiple (> 3) views is addressed by Berthilsson *et al.* [12] who introduce a bundle adjustment strategy to allow curve correspondences to vary along the image curves. Kaminski and Shashua [92] derive constraints on algebraic curves from multiple views, while Martinsson *et al.* [126] combine curve fitting and reconstruction for planar curves.

Non-rigid structure from point matching The introduction of non-linearity in the SFM problem introduces extra variables (*i.e.* extra factors in the factorization approach) such as a shape model and fitting parameters. The 3D shape in each view is modeled as a linear combination of *basis shapes*, denoted $\mathcal{B}_{1..K}$. The shape in each input view is given by linear combination coefficients α, possibly with associated transformation parameters– $s, \mathtt{R}, \mathbf{T}$. Most methods in NRSfM need a set of point correspondences across images (or video). A set of P point correspondences over N input images is represented as $2P \times N$ *measurement matrix* $\mathtt{W}$, which concatenates the 2D points $\mathbf{w}_{pn}$. Either factorization or bundle adjustment upon the least squares error function can be performed to derive the solutions. To derive the $\mathtt{W}$, most methods exploit the temporal smoothness to constrain correspondences and camera motion between successive frames [23, 22, 45, 194]. Naturally, this is susceptible to problems typical of tracking such as lost tracks, unreliable correspondences and occlusions and clutter. Torresani *et al.* [195] show how rank bounds can be used to constrain low level motion in images (as shown by Irani [91]) thus augmenting available point tracks. In a problem parallel to non-rigid structure from motion (NRSfM), Bascle and Blake [8] use factorization to separate pose and expression from images for animation. They assume known bases which is equivalent to solving only a subset of the actual problem. However, we are usually interested in finding all the unknowns. The recovery of the unknown model parameters (α) was initially cast as a matrix factorization problem (for scaled orthographic projection this is a three stage factorization as shown by Bregler *et al.* [23]). However, more recent work has cast it as a maximum *a-posteriori* (MAP) estimation of the parameters [45] or maximum likelihood distribution fitting [194]. The optimization for the latter case can be performed by either coordinate descent based methods [194], or by bundle adjustment based approaches [189, 197]. For the weak perspective camera model and linear bases Xiao *et al.* [207] improve upon the coordinate descent method of [22, 195, 194] by giving a closed-form solution for this case. They point out that to tackle ambiguity both rotation constraints as well basis constraints. An interesting aspect of work in this area is the progression in the use of priors. The completely model-free work by Bregler *et al.* [23] was appended with simple quadratic smoothness priors on bases in [195]. Torresani *et al.* [194] use a PPCA model on latent parameters and bases for a more compact formulation that can be approximately solved using Expectation-Maximization[13].

2.5 Object detection and segmentation

Segmentation is often a precursor to other image-processing and vision tasks such as image matting, scene augmentation and semantic interpretation, and is considered intertwined with the process of object detection and recognition.

2.5.1 Low-level algorithms (bottom-up)

Feature-based segmentation: Assuming some image representation (RGB, HSV *etc.*), one of the most basic methods involves deciding whether a pixel belongs to the foreground or background, based on its colour or texture with little acknowledgement of spatial relationships. Assuming that an object (or its parts) has consistent texture and colour, simple thresholding or the clustering based methods such as K-means [122] or mean-shift (see Comaniciu and Meer [32]) can be applied to segment the image. Many clustering methods have been investigated in the past (ISODATA, competitive learning [199], fuzzy kmeans, tree-based, multi-scale, adaptive kmeans) but the local nature of the observations limits the performance of these methods. The input can be made a bit more sophisticated by using local filter responses and using morphological or histogram based methods such as the watershed algorithm [175] for segmentation.

Incorporating spatial consistency: In addition to proximity in feature space, we also want segmented parts of an object to be spatially close on the image. Various methods based on region growing and split-and-merge techniques have been considered. In addition to taking local features into account such methods encourage spatial consistency thereby improving performance in the presence of noise, occlusion *etc.*

Smoothness and continuity of segmentation can also be imposed effectively in graph based segmentation approaches. Right from the 1970s graph theoretic approaches such as MRFs[3] and variational formulations (Mumford and Shah [138]) have been popular for solving the problem of segmentation. In graph-based approaches, the image is modelled as a graph where the pixels (or features) form the nodes and the linking edges denote weights expressing some similarity function. The challenges involve formulating the problem correctly and then finding the right algorithms to solve it. Segmentation involves creating partitions by breaking edges, the total weight of which is called the cut. Wu and Leahy [206] originally devised a clustering method based on optimizing this cut for undirected graphs. Shi and Malik [176] retrieve more balanced partitions by considering a normalized objective which aims to minimize the similarity across different partitions of graph and maximize it within the same partition. The resulting unbiased unsupervised hierarchical partitioning (clustering) problem is approximately solved as a generalized eigenvalue problem.

In the modern day, Graph Cuts form the backbone of most popular segmentation methods. For sub-modular energies exact global optima can be found, thereby allowing objective evaluation in contrast to other existing algorithms. Boykov and Jolly [19] addressed the problem of segmenting monochrome images by taking a set of hard and soft constraints to solve for a globally optimal segmentation. The hard constraints can be background and foreground seed constraints, and also to build foreground and background intensity distributions. This can be used to exactly optimize the energy function (Gibbs energy) to quickly segment images with a minimal amount of user input or correction. The energy consists of (1) an evaluation of the fit of opacity distribution alpha to the data given the model, and (2) a smoothness term for consistent labelling in regions of similar gray-level. GrabCut [163] adds an iterative procedure that alternates between estimation and parameter learning. First a “hard” segmentation is obtained using iterative graph cuts. This is followed by border matting to allow mixed pixels near the hard segmentation

[3]A Markov Random Field (MRF [70]) is a stochastic process in which the conditional probability for the labelling of a particular pixel is only a function of the neighboring pixels of its clique and not the entire image. For a reasonably broad class of sub-modular functions MAP (maximum a-posteriori) estimates of graph labelling can be found by various methods (min-cut [60, 20], push-relabel [75] , Liu and Yang [119]).

boundary. Segmentation can be subsequently improved in a real-time fashion with a small number of edits and re-use of existing computations (as shown by Dynamic Graph Cuts [101]).

The basic model remains a very simple combination of unary likelihoods and a Potts-model based pairwise prior. Therefore the ability to model complex objects is limited to user-input seeds and the sophistication of foreground and background models built from them. Complex topologies or images could take a large number of edits without providing the exact desired segmentation.

2.5.2 Higher-level algorithms

In many real life applications there is a need to segment complex objects in the presence of noise, clutter and occlusion. Though pixel level information can be replaced by more robust shape and texture cues from the neighbourhood, there is a need to incorporate higher-level class-specific information and long range relationships for representation and segmentation of complex objects. The use of approaches like CRFs (Conditional Random fields [104]) allows for inclusion of more complex energy terms (such as data-dependent pairwise terms which is neither a prior nor a likelihood) and has been used in a variety of work such as [103, 173].

Borenstein and Ullman [18] discussed the concept of class-specific segmentation as opposed to existing bottom-up segmentation methods. They use known object shape characteristics and handle deformation using a fragment based representation. The fragment dictionary is matched to a test image to maximize a combination of matching score and inter-fragment consistency score, so the final segmentation has the object shape characteristics. Agarwal and Roth [5] advance this by actually learning the spatial layout from data. Although local consistency of fragments is checked, there is no global framework to check for object consistency thereby being unable to guarantee valid class detections.

Shape based segmentation Leibe and Schiele [110] improve upon [18] by introducing a global shape consistency. Instead of directly training a classifier on appearance code-books as [5], they use a probabilistic voting scheme to generate an object hypotheses and category-specific segmentation. So far, basic bottom-up information is ignored, making the segmentation unfaithful to important image detail.

Object detection has used a combination of shapes and texture for object representation ([67, 196, 192]) in addition to incorporation of spatial relationships ([56, 53, 54, 143, 178]). Though these primarily provide detections, the localization can be incorporated into the CRF formulations for segmentation, *e.g.* foreground-background constraints for an automated GrabCut style algorithm [163]. ObjCut by Kumar *et al.* [103] tries to do this by combining the bottom-up information of traditional CRF approaches with a top-down Pictorial Structure model and a latent variable model for object pose. In addition to effectively segmenting complex articulated objects, the method is able to provide reliable detections alongside.

Shotton *et al.* [179] proposed Textonboost which uses a boosted combination of texton features (jointly modelling shape and texture) in a CRF framework. It learns from positive and negative examples and is able to learn a discriminative model. Training and inference can be performed quickly, allowing the use of large datasets. Although their method produced good segmentation and recognition results, the rough shape and texture model caused it to fail at object boundaries. The problem of extracting accurate boundaries of objects is considerably more challenging. The joint modelling of shape, texture and edges allow simultaneous recognition and segmentation, but the lack of explicit object articulation or semantic context information leads to inaccurate segmentation.

Winn and Shotton [204] propose a new method for recognition and segmentation especially in the presence of occlusion. A dense part labelling allowing for asymmetric spatial constraints is used to create a layout consistent framework. Their expansion move algorithm allows for a very unique form of deformation of object part, but the ability to handle complex pose variation is again limited.

Levin *et al.* [113] tackle the inadequacies in segmentation by exclusively top-down and bottom-up methods. Principled training is conducted for their joint top-down and bottom-up method by training the entire framework together instead of separate modules. This results in a more compact shape representation as opposed to other methods such as ObjCut [103]. However, their method relapses into the use of pixel-intensity based similarity measures instead of texture *etc.* and the quest for jointly training the framework limits the ability to handle minimal variation in pose.

Superpixel-based segmentation Some methods adopt a more modular approach to segmentation where they want to consider pre-partitioned parts of the image for the purpose of segmentation. By learning CRFs over higher level features such as image superpixels instead of pixels, more complex relationships can be modelled and learnt. Russell *et al.* [165] use multiple instances per image to learn visual words for object classes and then derive the correct final segmentation in an unsupervised way. In a supervised approach, Parikh *et al.* [144] learn spatial context (co-occurrence, relative location and scales) between different object categories from over-segmented images for scene understanding. This can be deployed for recognition and segmentation of images.

Higher order energy and large cliques Many methods have tried to incorporate higher-level in the form of class-specific models or special connectivity priors into graph cut based algorithms. Ability to specify long range relationships can help model complex objects. Such relationships can be captured by a fully connected graph, but such graphs are complex to define and expensive to solve even at small sizes. There has been some progress on solving MRFs with larger cliques for segmentation. Roth and Black [162] show how generic image priors can be learnt over images with larger cliques using a Products-of-Experts framework (similarly approximate methods for efficient belief propagation over graphs with large cliques and real valued variables have been demonstrated by Lan *et* *al.* [105] and Potetz *et al.* [152]). Kohli, Ramalingam and others [100],[157] show a class of energy functions for which multi-label segmentations can be computed optimally and efficiently over graphs with higher order cliques.

2.5.3 Edge and boundary detection methods

An object can be located in an image by accurately detecting its boundary. The basic image representation and edge finding approach used for subsequent analysis are important and many papers have been written on doing these better. Usually edges are detected using methods (Sobel, Laplacian, zero finding or Canny) prone to noise, smoothing and clutter due to the use of local information. Therefore it may be difficult to find a closed contour for an object and make object segmentation difficult.

The development of active contour models opened a new approach of boundary detection based on curve initialization followed by iterative optimization based on image forces designed so that the contour converges on the object boundary. Kass *et al.* [96] proposed one landmark idea for active contours–‘*snakes*’. The object boundary is represented by a curve (which may be open or closed) which is ‘attracted’ to image features such as edges. Desirable priors such as smoothness can be incorporated in the objective in order to combat noise, smoothing and clutter based problems. The objective function tends to be a complex nonlinear function, whose local optima correspond to the curve (snake) latching on interesting image features.

Berger [11] devised ‘growing snakes’ which allows the snake to extend at its ends and minimize the objective. The snake can subdivide, grow and parts of it may ‘die’. Higher energy bits can be discarded during the process leading to some robust solutions despite bad initialization, however there is high processing cost. Gradient vector flow snakes (Xu and Prince [208]) calculate the field of forces over the image at some resolution and drive the snake towards object boundary while ensuring that it is smooth. The diffusion operations used to calculate these fields claim to be effective even when the snake is far from the actual object. The drawback with finding local solutions with respect to snakes is their dependence on initialization and the image complexity. Also, smoothness priors can cause the snake to contract, due to which additional complex priors such as inflationary forces ([31]) have to be introduced. Regardless of these problems, this remains a popular idea, used extensively (see [168, 169, 29, 71]). Implicit functionals have been used to enable active contours to handle topological challenges as shown by Morse *et al.* [136].

Other methods in interactive segmentation are live wire [50] and intelligent scissors [137]. As opposed to snakes, these methods find globally optimal solutions. Given a fixed starting point, these methods solve a dynamic program over an image-based graph to calculate an active boundary that can be varied in length, complexity (knot points) etc. A detailed discussion about this method can be found in Appendix A.

Edges express an image concisely and therefore are important for the purpose of boundary finding. In the past algorithms have used edges without taking their neighbourhood information into account. In recent times however, many learning algorithms therefore have been proposed which deal with the end-to-end goal of integrating edge-based cues with object detection and segmentation.

Mikolajczyk *et al.* [133] use scale-invariant texture-based cues to recognize objects in the presence of ambiguous texture and clutter. Ferrari *et al.* [55] use contour segments derived from simple hand-drawn examples to detect objects with strong contour characteristics. The problem is formulated as one of finding object like shapes from an image-based contour network. Shotton *et al.* recognize the importance of contours for object recognition by integrate them in their feature set for recognition (see [178, 179]).

A common problem with edge finding and linking algorithms is that at best, one often only gets contour segments for objects. Ren *et al.* [158] advocate the use of piecewise linear curves defining a CRF and curvilinear continuity to form the energy. Upon optimizing this missing parts can be completed. Constrained Delaunay triangulation is used to generate potential completions, which seems like a slightly ad-hoc choice of algorithm, but demonstrates good results.

Martin *et al.* [125] use a supervised approach to detect boundaries in natural scenes using features based on brightness, colour and texture. A linear combination of the cues is used to capture the characteristic features displayed by object boundaries and a thorough investigation of these features is conducted at each pixel for all orientations and scales. A similar approach can be found in our work in Chapter 5. Dollar *et al.* [46] reiterate the inadequacy of traditional edge finding methods and stress the need to engage middle and high level information for this purpose. A supervised approach is used to learn discriminative models with probabilistic boosted tree classifiers. They argue in favour of finding edges that are relevant to the category/purpose at hand and accordingly their approach performs well for specific classes (though badly for generic image edges).

McHenry *et al.* [128] propose an interesting method for *finding glass* . They use the reflective and refractive properties of glass for segmenting it in images to hand-select features around glass edges (like colour, blurring, overlay consistency of internal and external regions, texture distortion and highlights). These are used to train a hierarchy of classifiers (single or multiple classifiers like SVMs) to identify glass edges. A global integration step follows, which merges fragments connected by a path. Active contour models like snakes that are attracted to the identified glass-edges are used to identify support regions as potential glass objects. More recently, Mairal *et al.* [123] proposed the use of a multi-scale discriminative framework based on learning sparse representations for class-specific edge detection which can be used to improve existing contour-based classifiers. They propose a multi-scale method to learn discriminative dictionaries from local edge and appearance cues. The formulation demonstrates both sparse reconstruction (based on l_0 and l_1 regularization constraints) and inter-class discrimination components. The use of sparse PCA techniques ensures fast updates and efficient optimization. The final classification is retrieved by building a linear classifier over each dictionary's reconstruction error at multiple scales and sparsity constraints.

Class-specific analysis combining various local and global cues, has seen increasing use in object detection and segmentation is therefore relevant to our work.

Chapter 3

Single View Reconstruction of Curved Surfaces

In this chapter we will show how silhouette-based user-driven cues from a single image can be exploited to effectively reconstruct fully freeform 3D models of various topologies.

3.1 Introduction and overview

The reconstruction of 3D objects from a single view is a severely under-constrained problem and has been of interest to researchers in vision and graphics. Image-based cues–texture, shading, silhouette *etc.*, must be employed to extract plausible shape from this image. Due to the need for many complex assumptions, cues from shading and texture (see § 2.1.4,2.1.5) cannot be used in most realistic situations. Comparatively, silhouette-based cues (§ 2.1.1 and 2.1.2) are easily employed with fewer assumptions.

The SVR problem can be framed as finding the most plausible 3D surface projecting to the given image silhouette. We show how silhouette-based cues can be used to find a closed, freeform 3D model of any topology given a single image. We also show how image-based cues such as singularities (cusps), object creases *etc.*, and intuitive user-driven cues can be inserted to model the object realistically.

Overview: This chapter is organized as follows. We will state the goal of single view reconstruction (SVR) mathematically as a constrained optimization problem in § 3.2. We introduce the surface representation in § 3.3 and the problem objective in § 3.4. We state the different kinds of constraints that can be imposed on such a surface in § 3.5. We first show how image-based silhouette constraints can be used to constrain the problem in § 3.6. We then show how the object can be inflated using a variety of methods in § 3.7. Complex viewpoints gives rise to image-based singularities, also known as cusps. These can be used to add important detail to the model as shown in § 3.8. Our freeform surface can be used to model objects of any topology as discussed in § 3.9. This is finally followed by the implementation details § 3.10 and the summary § 3.11.

3.2 Goal

Given a single image, we want to construct a plausible 3D model of the object. This is a severely under-constrained problem and there are an infinite number of candidates for the 3D shape. We describe the basic framework for obtaining a smooth 3D object from an image.

Given an image, any candidate 3D model must project to the 2D silhouette (see figure 3.1). There are an infinite number of candidates despite this constraint. Generally speaking we also want the 3D model to be as smooth as possible given the constraints. In the absence of specific information, this forms the following regularized objective: find the smoothest 3D surface obeying a set of image-based and user-specified constraints. To discuss this in detail, we need to formulate the problem mathematically. To do this, we first need to find a representation for the surface.

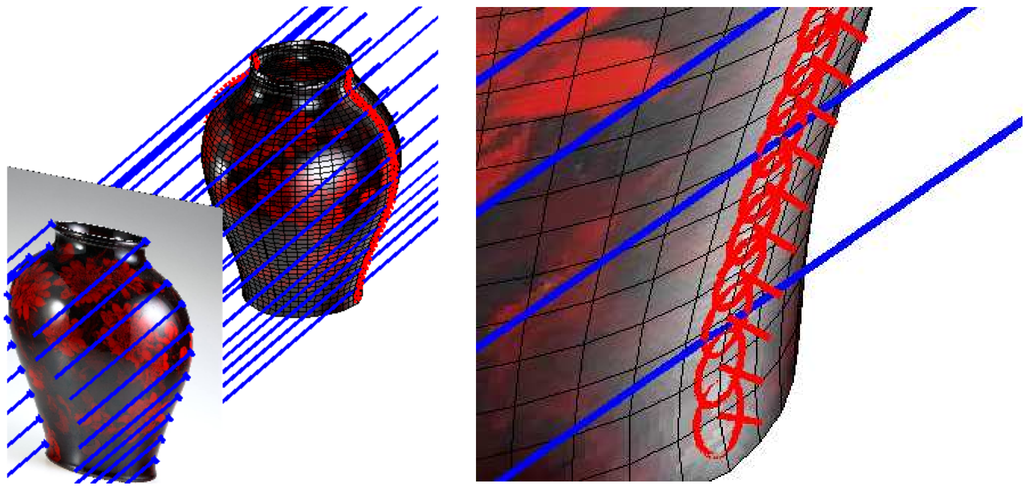

(a) (b)

Figure 3.1: "Lifting" 3D models from images: (a) Orthographic projection of the desirable 3D model to the given image is shown. (b) Close-up of a part of the surface at the Contour Generator (CG). Normals (in red) are plotted at the point of tangency of the CG with the viewing direction whose rays are plotted in blue.

3.3 Surface representation

Following Terzopoulos *et al*. [191] we use the parametric surface representation (see § 2.2), $\mathbf{r} : [0,1]^2 \mapsto \mathbb{R}^3$ since this allows us to model truly 3D freeform surfaces. The continuous surface S is denoted as a function of the parameters as shown:

$$\mathbf{r}(u,v) = [x(u,v), y(u,v), z(u,v)]^\top . \tag{3.1}$$

The surface (also called Geometry Image, see [82], § 2.2) is represented by three $M \times N$ matrices, one for each coordinate: $\mathtt{X}, \mathtt{Y}, \mathtt{Z}$, representing the surface defined along a discretized sampling of the parameter space. M, N are the number of samples along the parameters (u,v) of the surface required to represent surface detail adequately. When solving for the surface, the same matrices can be reshaped columnwise into vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$. These vectors can again be stacked into a single vector of unknowns $\mathbf{g} = \left[\mathbf{x}^\top \; \mathbf{y}^\top \; \mathbf{z}^\top\right]^\top$.

This is useful as a single vector representation for the surface coordinates.

For ease of notation any vertical constant u curve on the parameter space is called a longitude, while a horizontal, constant v line is called a latitude. For *e.g.* the longitude $u = u_1$consists of all points whose parametric values are: $u = u_1, v \in [0, 1]$.

Later in this chapter, we will introduce the concept of "identification" of curves in the parameter space. Two curves are said to be identified when they overlap so that their corresponding points have exactly the same 3D coordinates. In addition, the derivatives defined on the curves must be smooth across their seam. This is a particularly useful concept in representing complex topology.

3.4 Smoothness

We need a metric to define smoothness on the surface. This is important in order to regularize the surface in the absence of other constraints. The surface is computed by minimizing a smoothness objective function. Here, smoothness is measured by the thin-plate bending energy (3.2) of [186, 211, 80] as:

$$E(\mathbf{r}) = \int_0^1\!\!\int_0^1 \|\mathbf{r}_{uu}\|^2 + 2\|\mathbf{r}_{uv}\|^2 + \|\mathbf{r}_{vv}\|^2 \, du \, dv, \tag{3.2}$$

subject to the constraint that the surface's contour generator projects to the given image contour. Each term in equation 3.2 can be expanded using 3.1 as shown below.

$$\mathbf{r}_{uu}^2 = \mathbf{x}_{uu}^2 + \mathbf{y}_{uu}^2 + \mathbf{z}_{uu}^2. \tag{3.3}$$

This can be done for the other terms similarly. Central difference approximations are used for the first and second derivatives on the surface, and are represented by appropriate matrix operators [211]. Thus the derivative terms at a point (i, j) are discretely approximated as

$$\mathtt{X}_u(i,j) = \frac{1}{2N}\left(\mathtt{X}(i+1,j) - \mathtt{X}(i-1,j)\right) \quad \text{First derivative,} \tag{3.4}$$

$$\mathtt{X}_{uu}(i,j) = \frac{1}{4N^2}\left(\mathtt{X}(i+1,j) - 2\mathtt{X}(i,j) + \mathtt{X}(i-1,j)\right) \quad \text{Second derivative,} \tag{3.5}$$

ignoring issues brought up by curve parametrization for the moment. The vector of derivatives required to discretely evaluate (3.2) is conveniently represented [211] as a constant $(MN \times MN)$ sparse matrix $\mathtt{C}_u$, so that $\mathbf{x}_u = \mathtt{C}_u\mathbf{x}$. Second derivatives are similarly represented by $\mathtt{C}_{uu}, \mathtt{C}_{uv}$ and $\mathtt{C}_{vv}$, so the bending energy of (equation 3.2), can be expressed in discrete form as:

$$\epsilon(\mathbf{x}) = \mathbf{x}^\top(\mathtt{C}_{uu}^\top\mathtt{C}_{uu} + 2\mathtt{C}_{uv}^\top\mathtt{C}_{uv} + \mathtt{C}_{vv}^\top\mathtt{C}_{vv})\mathbf{x}, \tag{3.6}$$

$$E(\mathbf{g}) = \epsilon(\mathbf{x}) + \epsilon(\mathbf{y}) + \epsilon(\mathbf{z}) \tag{3.7}$$

$$= \mathbf{g}^\top\mathtt{C}\mathbf{g}, \quad \mid \mathtt{C} \text{ is square of side } 3MN. \tag{3.8}$$

Without constraints on the surface, $E(\mathbf{r})$ is minimized by the trivial solution $\mathbf{r}(u,v) = \mathbf{0}$. Also the resultant model may not produce a projection consistent with the image given to us. Imposing constraints generates more plausible shapes.

3.5 Constraints

We define constraints which impose one of the following on the surface: (i) position, (ii) normal, and, (iii) partial position.

Position constraints: are of the form:

$$\mathbf{r}(u,v) = [x(u,v), y(u,v), z(u,v)]^\top, \tag{3.9}$$

for known values of u, v and the 3D position coordinates.

Normal constraints: require the surface normal at (u, v) to equal a supplied normal $\mathbf{n}$. The normal to $\mathbf{r}$ at a point is the unit vector along $\mathbf{r}_u \times \mathbf{r}_v$. We can impose this as a pair of linear constraints in $\mathbf{r}$ as follows:

$$\mathbf{n} = \frac{\mathbf{r}_u \times \mathbf{r}_v}{\|\mathbf{r}_u \times \mathbf{r}_v\|} \tag{3.10}$$

$$\Rightarrow \mathbf{n}\cdot\mathbf{r}_u(u, v) = 0, \tag{3.11}$$

$$\mathbf{n}\cdot\mathbf{r}_v(u, v) = 0. \tag{3.12}$$

Partial position constraints: do not impose constraints on all 3 co-ordinates simultaneously. Sometimes we have only partial information at a point and therefore only a subset of the coordinates of a 3D position are available. *E.g.* only the z component could be constrained at a parametric point (u, v).

For all of the above, the constraints are linear in $\mathbf{g}$ and are easily represented as a separate constraint equation. Each of the constraints is linear and forms rows of $\mathtt{A}, \mathbf{b}$ where $\mathbf{g}$ is the vector holding the surface coordinates:

$$\mathtt{A}\mathbf{g} = \mathbf{b}. \tag{3.13}$$

This linearity is important because it leads to a simple quadratic minimization of a convex function as shown in (3.14-3.20)

Before describing the optimization itself, we will show how to extract constraints from readily available information. Depending on how they arise these constraints are either:

- *image-based* from the silhouette, or
- *user-defined* inflation

We now show how the silhouette constraint may be represented in a way that is linear in the unknown: $\mathbf{g}$.

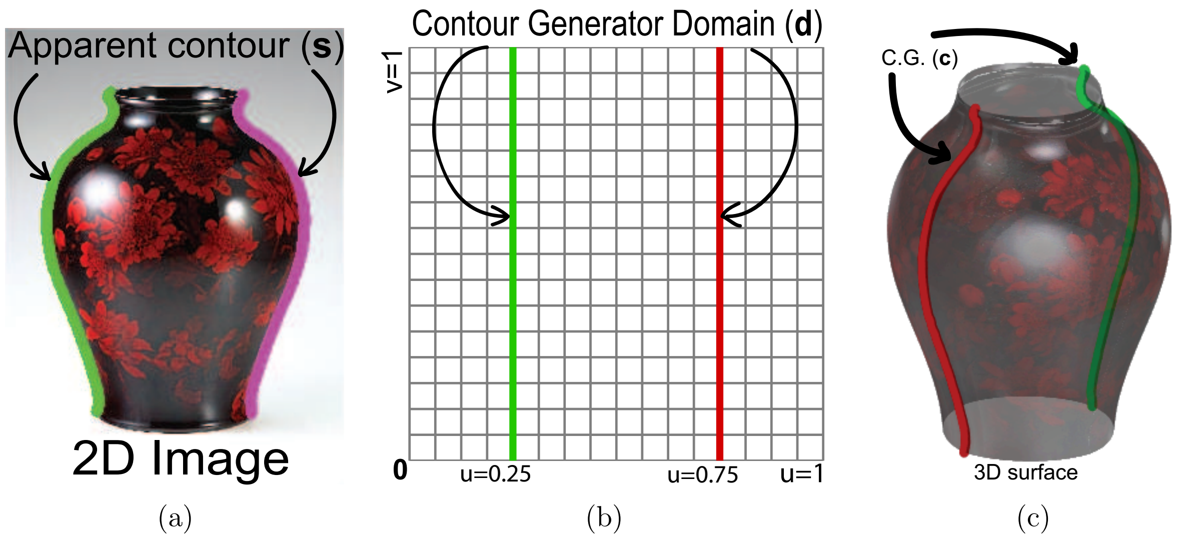


Figure 3.2: Reconstruction of 3D **surface from apparent contour constraints:** (a) The apparent contour **s** is marked on the input image. (b) If optimizing parametrization-invariant curvature, we would be at liberty to place the contour generator's domain (see § 3.6) anywhere in the parameter space, provided topological constraints are maintained. In practice we minimize an approximation to curvature. (c) Reconstructed surface, with contour generator **c** superimposed.

3.6 The Silhouette Constraint

The object silhouette is an important readily available image-based constraint. The object silhouette (refer to [99] for an in-depth study) is the image of the Contour Generator (CG). The CG is a 3D curve on the surface (red curves in figure 3.1, also in 3.2 column (c)), at which the viewing direction is tangent to the surface. If the surface is viewed in the direction of $\mathbf{r}$ from the camera center, then the surface appears to fold, or to have a boundary or contour generator. The CG's *domain* is a curve in the (u, v) parameter space (figure 3.2(b)) . If that curve is $\mathbf{d} = \{\mathbf{d}_t = (u_t, v_t)|0 \leq t \leq 1\}$ then the CG is $\mathbf{c}_t = \mathbf{r}(u_t, v_t)$. We are given an image silhouette $\mathbf{s} = \{\mathbf{s}_t|0 \leq t \leq 1\}$ *i.e.* $\mathbf{s}$ is the infinite set of 2D points on the silhouette (see figure 3.2(a)) and we are assuming it is parametrized by the same curve parameter t as the CG domain.

So far, our constraints have been associated with their corresponding (u, v) parameters. For the general 3D surface representation we have just discussed, this mapping is not available for the silhouette constraint. We show that the object's silhouette is a curve for which the mapping may readily be obtained. The discontinuity curves of [211] therefore fade in their importance for demarcating objects.

The silhouette constraints can be summed up as follows:

- The candidate 3D model must project to the 2D apparent contour.
- At the CG, the surface must also be tangent to the viewing direction.

We continue to assume orthographic projection like Zhang *et al.* [211]. For the moment let's assume that the surface does not exhibit self-occlusions along the silhouette. At each point on the 2D silhouettes**s**, we can compute the 2D unit normal (n_x, n_y). Under orthographic projection along Z, the corresponding 3D normal at the CG $\mathbf{n}$, must have a Z component of zero. Therefore, the 3D normal at t is given by $\mathbf{n}_t = (n_x, n_y, 0)$. This means we know the surface normal at any point on the contour generator, so the (infinite) set of linear constraints which force the silhouette of the 3D surface $\mathbf{r}$ to coincide with $\mathbf{s}$ are:

$$\left(\begin{smallmatrix}1&0&0\\0&1&0\end{smallmatrix}\right)\mathbf{r}(u_t, v_t) = \mathbf{s}_t \qquad \text{[Projection]}, \qquad (3.14)$$

$$\mathbf{n}_t^\top \mathbf{r}_u(u_t, v_t) = 0 \qquad \text{[Normal]}, \qquad (3.15)$$

$$\mathbf{n}_t^\top \mathbf{r}_v(u_t, v_t) = 0 \qquad \text{[Normal]}. \qquad (3.16)$$

There is a freedom in the parametrization, so that the curve in (u, v) space which is the pre-image of the contour generator can be chosen.

If we were minimizing curvature, we would be at liberty to choose any re-parametrization of (u, v) without changing the energy or loss of generality. This means that (u_t, v_t) in the above constraints are *known* points. In reality, our energy $E(\mathbf{r})$ (3.2) is an approximation to surface curvature. We use this freedom to identify arbitrary *longitudes* (constant u) curve in (u, v) as the domain of the CG and fit the surface, subject to that constraint. For the case of a cylinder from a simple viewpoint, curves of constant u in the (u, v) are chosen as the domain of the CG—see the vertical bold lines in figure 3.2. In particular we choose the uniformly spaced curves $u = u_l = \frac{1}{4}, u = u_r = \frac{3}{4}$, to coincide with the CG.

3.6.1 Are Silhouette Constraints Enough to Model an Object?

In a discrete setting, each segment of the contour generator is a set of 2D points. Let us consider just the left segment, corresponding to $u = u_l$. The 2D curve is approximately arc-length sampled at n points, giving 2D points $\{(s_j, t_j)\}_{j=1}^{n}$, with associated 2D normals (p_j, q_j). If we let (i_l, j) in general be the integer grid coordinates corresponding to parameter space location (u_l, v), then we may write the above constraints in terms of our discretization as:

$$\mathtt{X}(i_l, j) = s_j, \qquad j = 1..n \tag{3.17a}$$

$$\mathtt{Y}(i_l, j) = t_j, \qquad j = 1..n \tag{3.17b}$$

$$p_j \mathtt{X}_u(i_l, j) + q_j \mathtt{Y}_u(i_l, j) = 0, \qquad j = 1..n \tag{3.17c}$$

$$p_j \mathtt{X}_v(i_l, j) + q_j \mathtt{Y}_v(i_l, j) = 0, \qquad j = 1..n \tag{3.17d}$$

amounting to $4n$ linear constraints on $\mathtt{X}$ and $\mathtt{Y}$ (remembering that $\mathtt{X}_u$ is linear in $\mathtt{X}$ *etc.*). By reshaping matrices appropriately, these may be rewritten as a matrix equation of the form $\mathtt{A}_l\mathbf{g} = \mathbf{b}_l$, where $\mathtt{A}_l$ is of size $4M \times MN$. We can repeat this process for the other contour generator segment, giving constraints $\mathtt{A}_r\mathbf{g} = \mathbf{b}_r$. Stacking these matrices, and others we shall see later, into a single large matrix yields the complete set of linear constraints $\mathtt{A}\mathbf{g} = \mathbf{b}$ as previously mentioned in (3.13). $E(\mathbf{g})$ is then minimized as shown below: our objective function is quadratic and constraints are linear:

$$\min_{\mathbf{g}} \frac{1}{2}\mathbf{g}^\top \mathtt{C}\mathbf{g}, \text{ subject to } \mathtt{A}\mathbf{g} = \mathbf{b}. \tag{3.18}$$

We can formulate a Lagrangian for the problem as shown:

$$E(\mathbf{g}) = \frac{1}{2}\mathbf{g}^\top \mathtt{C}\mathbf{g} + \boldsymbol{\lambda}^\top (\mathtt{A}\mathbf{g} - \mathbf{b}) \,. \tag{3.19}$$

Optimizing the Lagrangian for this quadratic programming problem is equivalent to solving the linear matrix equation given by

$$\begin{bmatrix} \mathtt{C} & \mathtt{A}^\mathtt{T} \\ \mathtt{A} & \mathtt{0} \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}, \tag{3.20}$$

which can be readily achieved using sparse methods. The four constraints expressed in (3.14), (3.15) and (3.16) constrain the x and y coordinates of the surface only. Our approximation to curvature in the form of the bending energy does not couple the z energy to that of the x and y terms, meaning that a surface which projects correctly to the apparent contour but has $z(u, v) = 0 \ \forall u, v$ is a trivial solution to this problem. In order to avoid this we need to "inflate" the surface and flesh it out into a plausible model.

3.7 Inflation: for shaping the object

Inflation can be performed in the following two ways:

- One method is to model the inflationary force as a function, incorporated in the objective function. Optimizing the objective function, will automatically inflate the surface.
- Alternatively, full or partially supplied 3D points can be used to supply constraints to the object causing inflation. We use this method in our work because of its simplicity in optimization. These may be derived from image information or can be user-driven.

Inflation constraints can be of two types:

1. Interpolation constraints: The surface is nailed to a few points at certain locations.

2. Approximation constraints: The surface is encouraged to pass close to certain points.

Given a few constraints of either type, the rest of the surface adjusts itself to form the smoothest surface given the objective function. A detailed discussion of such constraints follows.

3.7.1 Interpolation constraint

An interpolation constraint is of the form:

$$\mathbf{r}(u_k, v_k) = \mathbf{r}_k,$$

for given $(u_k, v_k, \mathbf{r}_k)$. The subscript indicates that there might be several such constraints. Constraints may also be supplied on just a single component of a point's position, such as its depth, $z(u_k, v_k) = z_k$. For example, we may insist that the surface passes through the plane $z = 1$ at $(u, v) = (\frac{1}{2}, \frac{1}{2})$, and the plane $z = -1$ at $(u, v) = (0, \frac{1}{2})$. Each of these constraints may again be expressed as linear functions of the surface discretization $\mathbf{g}$. We now take a look at some kinds of interpolating constraints.

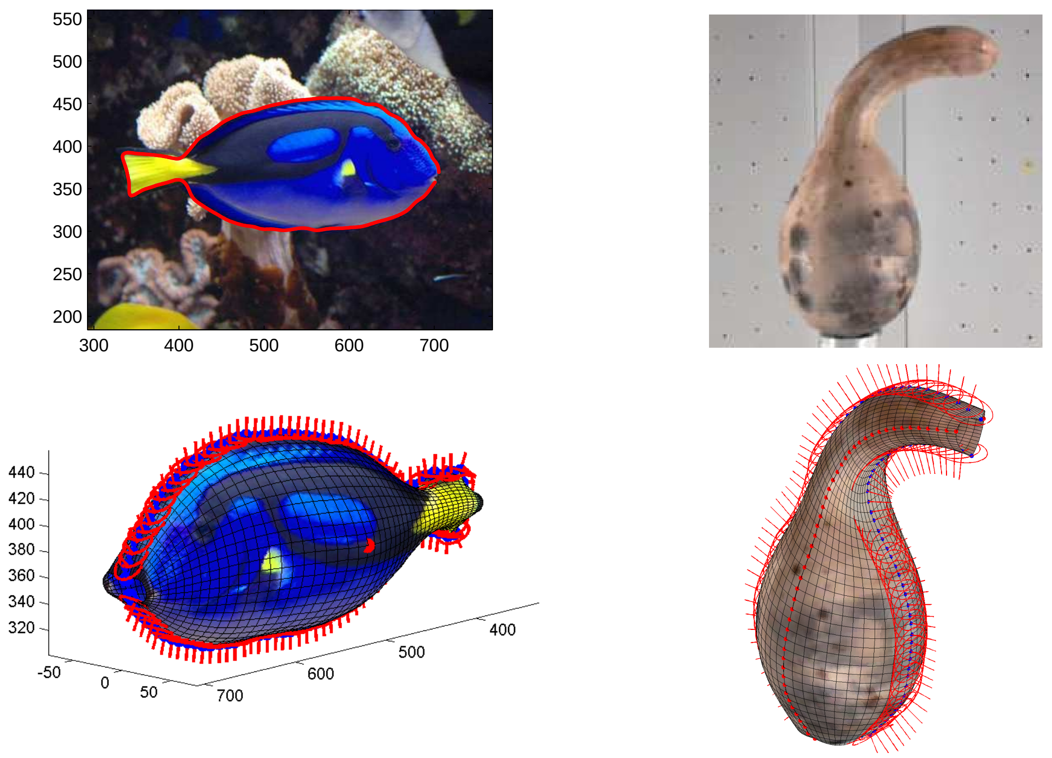
3.7.1.1 Inflating from Adjacent Edges of the Silhouette (*cylindrical inflation*)

A Generalized Cylinder is a surface which is cylindrical about a virtual freeform 3D curve/spine (see § 2.2). If the longitude $u = 0$ is identified (as explained in §3.3) with the longitude $u = 1$ the resultant surface will be of cylindrical topology, where the spine can be parametrized along its length by v. One such example is the vase shown in figure 3.1. The silhouette provides a frontal cross-section that can be used to guess the inflation for the object. Note however, that we do not enforce the contour generator of the object to be parallel to the image plane. This generalized cylinder can be completely determined given the radius along its spine. Assuming frontal projection, the left $(u = 0.25)$ and right

($u = 0.75$) parts of the silhouette can be sampled at uniform speed to find corresponding points in v that define the diameter of this cylindrical object, as computed below:

$$\begin{aligned}\text{radius}(v) =& \frac{1}{2}\|silhouette(0.25, v) - silhouette(0.75, v)\| \\ =& \frac{1}{2}\|x(0.25, v) - x(0.75, v), y(0.25, v) - y(0.75, v)\|, \quad |\text{assume CG lies in plane } z = 0.\end{aligned}$$

This radius can be used as an inflation constraint on diametrically opposite longitudes ($u = \{0, 0.5\}$) for each value of v. Depending on the complexity of the object, inflation can be done at select v points or all along the spine, as shown in figure 3.3.This method has some similarity to the process of inflation proposed by Terzopoulos [191].



Inflation at the two red dots:
$z(0.5, 0.5) = \text{radius}(0.5)$
$z(0, 0.5) = -\text{radius}(0.5)$

Inflating all along the spine:
$z(0.5, v) = \text{radius}(v)$
$z(0, v) = -\text{radius}(v)$

Figure 3.3: Inflating with adjacent parts of silhouette. The silhouette position and normal constraints are superimposed on the reconstructions. The inflation constraints are in bright red.

3.7.1.2 Inflating by Distance Transform

When faced with objects that don't belong to the cylindrical topology, we need methods that will require minimal user input. At the same time, we want to use the dimensions of the object to guess its inflation. The distance transform of an object's silhouette acts a measure of an object's size. At any given point on the image $\mathbf{p} = (x, y)$, the distance transform is defined as its distance to the closest silhouette point $(\mathbf{s}_t, 0 \leq t \leq 1)$, as shown:

$$\mathrm{DT}(\mathbf{p}) = \min_t \|\mathbf{p} - \mathbf{s}_t\|. \tag{3.21}$$

Assuming the object is uniformly spread out in space, the maximum value of the distance transform is indicative of the amount of inflation required to make a realistic reconstruction of the object. In Figure 3.4, the distance transform of one silhouette marking for a teapot (sans handle) can be seen. A select few points as shown in Figure 3.4 are chosen for inflation. The value of the distance transform at these locations on the object's projection is used for inflating the object. It is possible to compute the distance transform of all pixels in the image with respect to the (closed) boundary. However the total distance transform is rarely smooth and therefore we avoid using all points on it for inflation purposes. We then inflate select points on the image by their distance transform (see Fig. 3.4). The constraint can be a position or a partial position constraint. Igarashi *et al*. [89] use a similar inflation mechanism in *Teddy* (see § 2.1.3).

As seen in the figure, the annotations are users clicks in the (x, y) space. Inflation constraints must always be enforced on (u, v). Therefore, a mapping $(x, y) \rightarrow (u, v)$ is needed. At the outset, this mapping is unavailable. The following two step process is needed:

- We use the image-based silhouette constraints (projection and normal) as shown in equations (3.14-3.16) to compute $\mathbf{r}$ with $\mathbf{z} = 0$. We now have a mapping from (u, v) to $\mathbf{r} = (x, y, z)$. With this we can compute the parameter values at which to insert the

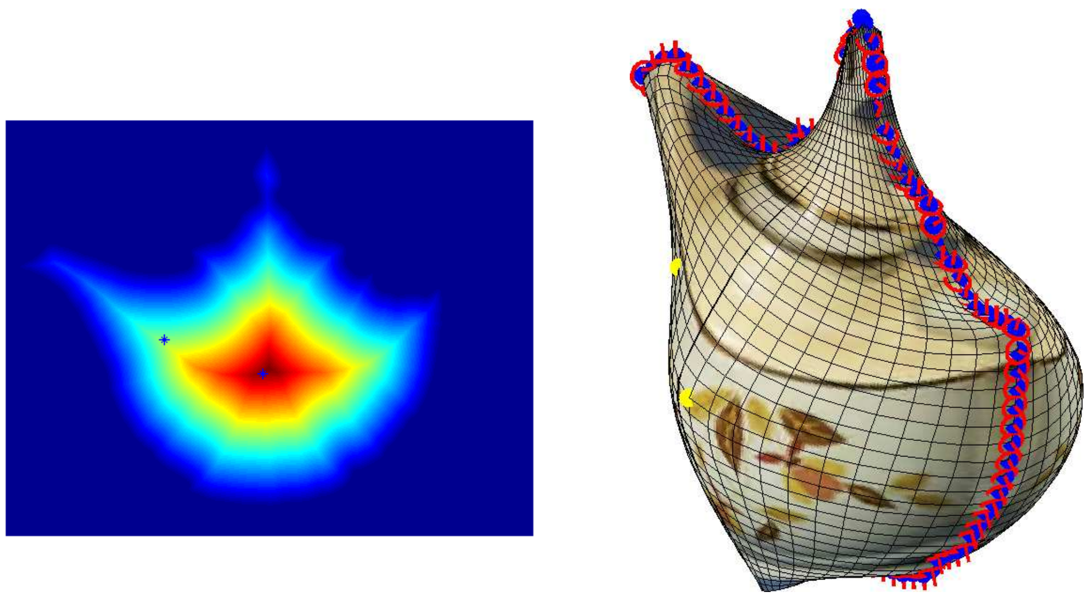


Figure 3.4: Inflation using the distance transform

inflation constraints arising from the distance transform. Given a user click (x_1, y_1), we can retrieve the corresponding (u_1, v_1) from the initial surface.

- Partial (optionally full) point constraints in z can then be used in addition to the framework with existing constraints to inflate the object:

$$\begin{aligned} z(u_1, v_1) &= \mathrm{DT}([\begin{smallmatrix} x_1 \\ y_1 \end{smallmatrix}]), \\ z(0.5 + u_1, v_1) &= -\mathrm{DT}([\begin{smallmatrix} x_1 \\ y_1 \end{smallmatrix}]), \\ x(u_1, v_1) &= x_1 \quad \text{(optional)}, \\ y(u_1, v_1) &= y_1 \quad \text{(optional)}. \end{aligned}$$

The system is solved again to find the 3D model. The resultant model will adhere to the given constraints and therefore no further iterations are necessary. This method is similar to the scheme of inflation used by Garish *et al.* [89] and works well for rotund objects. Rotund objects are loosely defined as objects with simple silhouettes with few concavities and spherical topology.

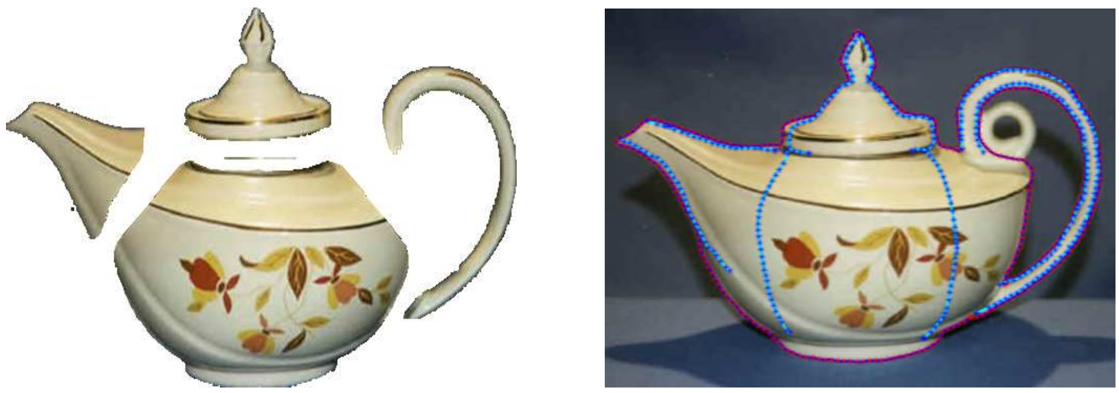
3.7.1.3 Inflation Curves

We have shown how cylindrical and distance transform based inflation can be done for objects with simple silhouettes. Assumptions about object shape being generalized cylinders work well for inflating a number of simple objects as shown in the illustration from "Alice in Wonderland", in the gallery (see figure 1.4).

We often want to reconstruct objects that are complex and not simply rotund or topologically cylindrical. Intuitively, complex objects are composed of simple parts such as generalized cylinders. Therefore one solution for reconstruction of a complex object is to individually reconstruct each generalized cylinder part and stitch them all together to create the larger model. However, we would like to keep the elegance of our existing framework in which the surface is solved for, as a whole, without building individual components. Even while treating the surface as one whole, we can use the part based information to perform intelligent, selective, part-based inflation, removing the need for stitching. For example, the teapot can be split into a spout, belly and lid as shown in figure 3.5 (a). The blue inflation curves in figure 3.5 (b) denote the hypothetical silhouettes of each part in the image used for inflation, while the red curve denotes the actual teapot silhouette used to provide the silhouette constraints for the actual reconstruction. Each part is a generalized cylinders, but when used for inflation together, they yield this complex teapot.

- *Hypothetical* silhouettes of the sub-objects (could be spherical, toroidal or cylindrical but represented as generalized cylinders) in the image are identified (see blue curves in figure 3.5).

- The inflation constraints are determined as the radii along the corresponding parts of the silhouette along the spine for each part. This yields us a set of inflation constraints of the form $z(x_i, y_i) = r_i$. For example, the pair of blue curves denoting the hypothetical spout silhouette are used to measure the radius of the spout along



(a) Intuitive decomposition of teapot (b) red silhouette and blue inflation curves

Figure 3.5: User-assisted inflation: The parts of a teapot are used to intelligently inflate different parts of it. These inflation constraints are used to reconstruct the entire teapot at once instead of selectively reconstructing and stitching together the parts.

its hypothetical spine. The spout's spine itself lies at the midpoint of a warp between equidistant points along the pair of the blue curves.

- The (u, v) locations of the resultant inflation constraints (radii along the spine) are unknown (similar to discussion in §3.7.1.2). Unlike the silhouette constraint, we do not have the freedom to assign an arbitrary (u, v) parameter curve to these inflation constraints, so the system is solved first with the silhouette constraint only (resulting in a flat teapot), yielding a mapping from image locations (x_i, y_i) to parameter locations $(u_i, y_i)\,\forall i$. The mapping is used to identify the locations for the inflation constraints $z(u_i, v_i) = r_i$, which can be augmented to the constraint set to yield a realistic inflated object as seen in figure 3.5.

Our approximation of curvature is handy for this form of two stage surface computation. Since our smoothness is uncorrelated over x, y, z we can solve for x and y only, in the first step to find the mapping. This can quicken our computation because we are dealing with smaller matrices.

On the other hand, because of the simple approximation of the bending energy to curvature, the resulting surface behaves like a fishnet stocking wrapped around an invisible model. Our methods of inflation are useful for simple problems. However, with complex silhouettes, when the actual 3D form of the object becomes complex, problems with the approximation to curvature, come to the fore. In such cases, the surface can spill over the contour, as shown in figure 3.6. The contour generator limits the surface to be on its inside in its image projection. However, it is difficult to constrain our equations to obey this physical phenomenon. It is necessary to make some modifications to handle this spillage.

3.7.1.4 Spillage issues

The resultant surface computed from the above methods behaves like a stocking wrapped tightly around an invisible object. This could result in a certain amount of surface spillage as seen in figure 3.6 (a). This is because

1. Even though the silhouette constraint guarantees that the surface is *locally* consistent with the image silhouette, it does not prevent unconstrained parts of the surface spilling out into the background. Ensuring the whole surface lies within the silhouette is not trivial to implement.

    (a) While the surface at the contour generator can be encouraged to be locally convex by constraining its Gaussian curvature, spillage can still occur away from it.

    (b) The intrinsic annotation is a set of silhouettes which may not be closed. Thus, defining the inside and outside of the object is difficult. We can insist on closed silhouettes thus removing this issue. Each point on the surface is projected to the image and a check is performed to see if this point lies within the intricate polygon defined by the silhouette. This is an expensive operation. Despite being limited to the silhouette, the surface mesh does not necessarily look more plausible, often being distorted and unreal.

2. If our curvature measure were perfect, high curvature regions resulting from silhou-

ettes could be discouraged.

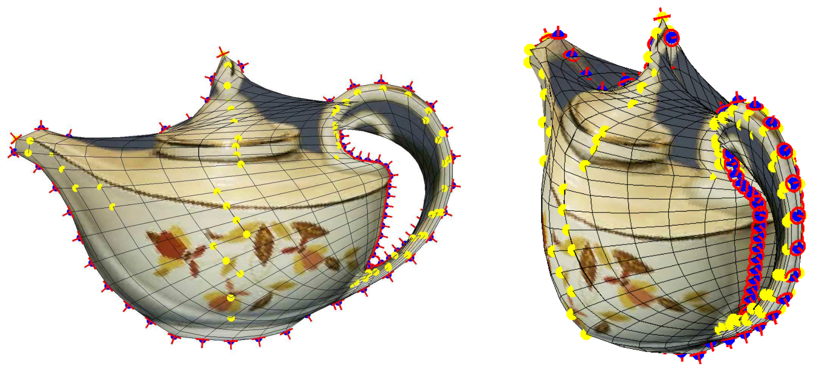
3. We have a fixed parameter assignment resulting from many heuristics. This compounds the problem. Iterative reassignment of parameters to constraints can ameliorate the problem.

It is important to remember that even if some of the above problems were fixed, we might not be able to effectively find the perfect model. The image of a model such as the current teapot, need not come from an object which minimizes curvature in the first place. In the absence of better information, we use it to regularize our shape but we may be far from perfect in doing so.

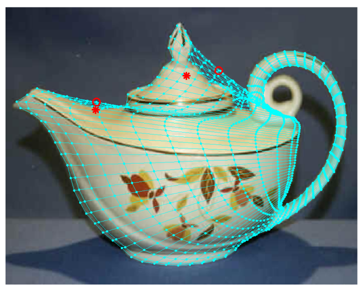
It is possible to correct this spillage by further constraining these surfaces. The user uses his mouse to drag and drop the surface from a point of spillage (figure 3.6 (b), red circle) to a position inside the silhouette where that point on the surface is expected to be (figure 3.6 (b), red asterisk). This does not interfere with any of the previously inserted silhouette constraints, but only indicates how the parameter lines must move in order to prevent spillage. By clicking on the spilled surface, the user selects the parameter lines at that point. Dragging and dropping it in the interior of the object's silhouette introduces partial position constraints on the x and y values at the selected parametric points corresponding to the spillage. This additional constraint ensures that the spillage is corrected without undue influence on the 3D shape as seen in figure 3.6 (c).

3.7.2 Approximation constraints

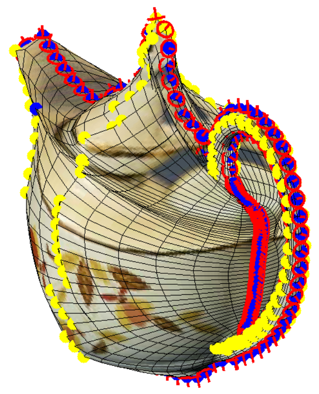
For the vase in figure 3.2, the inflation constraints provided were interpolation constraints which assumed the surface is topologically cylindrical. In this case, its axis of revolution is also parallel to the image plane. The simple strategy for inflation ensures that the silhouette constraints (see § 3.6) are met. The reconstruction, produced by these constraints, expressed at particular parameter locations, depends on the (u, v) locations. But for simple surfaces the sensitivity to choice of (u, v) is generally low, as seen in §3.9.1 and figure 3.11.



(a) Spillage in different views



(b) Drag and drop correction



(c) Final output

Figure 3.6: Occurrence of spillage and fixing it

For more complex surfaces however, or where there is significant foreshortening, it is useful to supply **approximation constraints**. Approximation constraints cause the surface to pass near certain 3D points, for example, by minimizing an additional energy term of the form:

$$\alpha_k \|\mathbf{r}(u_k, v_k) - \mathbf{r}_k\|^2.$$

The factor α_k controls the extent to which each constraint should be satisfied. A collection of such constraints may be represented as the quadratic term $\|\mathtt{M}\mathbf{g} - \mathbf{m}\|^2$ (embedded with α), giving the modified Lagrangian:

$$L = \frac{1}{2}\mathbf{g}^\top \mathtt{C}\mathbf{g} + \frac{1}{2}\left\|\mathtt{M}\mathbf{g} - \mathbf{m}\right\|^2 + \boldsymbol{\lambda}^\top\left(\mathtt{A}\mathbf{g} - \mathbf{b}\right). \tag{3.22}$$

Optimization of this Lagrangian boils down to the slightly modified matrix equation shown below:

$$\begin{bmatrix} \mathtt{C} + \mathtt{M}^\top\mathtt{M} & \mathtt{A}^\top \\ \mathtt{A} & \mathtt{0} \end{bmatrix} \begin{bmatrix} \mathbf{g} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathtt{M}^\top\mathbf{m} \\ \mathbf{b} \end{bmatrix}. \tag{3.23}$$

As an example of the application of these constraints, the surface in figure 3.8 was constrained to approximately meet the planes $z = my \pm 1$ using the constraints $z(0.5, v) \approx mv + 1; z(1.0, v) \approx mv - 1$. The value of m is determined interactively in order to obtain the best-looking shape.

3.7.3 Surface creases

One final generalization is to allow the surface to crease. This means, as with the apparent contour, specifying a curve in the parameter space (say $u = u_c$), and constraining points along this curve to project to the image of the crease as in (3.14). The second modification is to $E(\mathbf{r})$: replacing the bending energy, computed from second derivatives, with a membrane tension energy—the sum of squares of first derivatives across the curve. Considering the contribution to $E(\mathbf{r})$ at a point (u_c, v) on the crease, we replace

$$\|\mathbf{r}_{uu}\|^2 + \|\mathbf{r}_{uv}\|^2 + \|\mathbf{r}_{vv}\|^2 \tag{3.24}$$

with

$$\|\mathbf{r}_u\|^2 + \|\mathbf{r}_v\|^2, \tag{3.25}$$

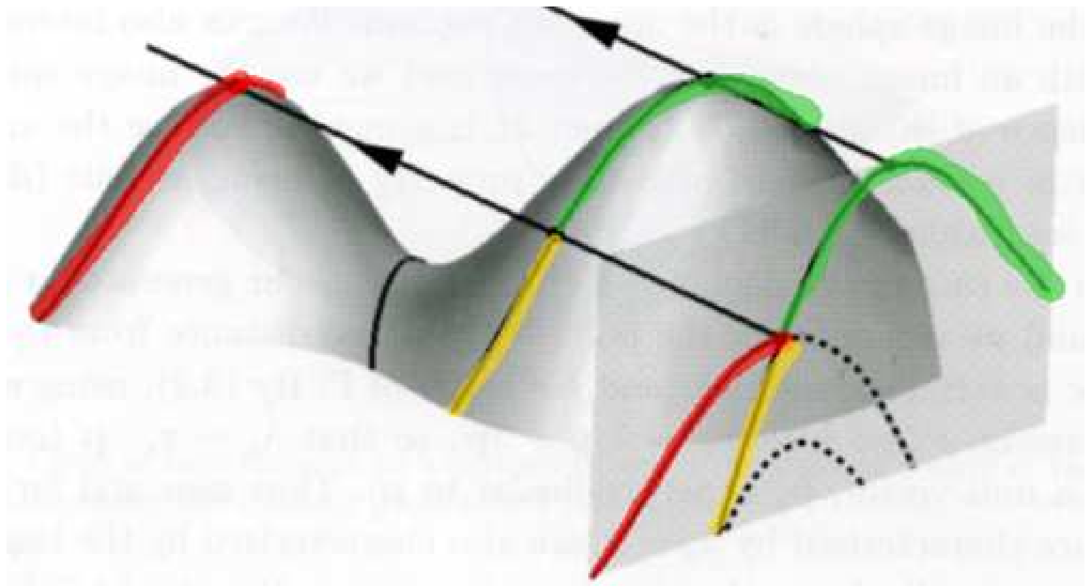


Figure 3.7: Multi-local singularities: The red and green parts of the silhouette in the image are continuous. However, they arise from different parts of the contour generator on the object's surface. (Picture courtesy: Roberto Cipolla)

so that the term transverse to the curve permits high second derivatives, allowing the crease to form when E is minimized. This is implemented by augmenting $\mathtt{C}$ with first-derivative terms, and adding weights to control the influence of each row of $\mathtt{C}$. The energy remains quadratic in the surface $\mathbf{g}$ and a global minimum is easily found.

3.8 Complex viewpoint

In this section we increase the scope of single-view reconstruction in two ways: first, we deal with more complicated viewpoints. These are still generic viewpoints, but now the apparent contour may terminate (where the view direction is asymptotic) or be self occluded (a bi-local event). An example is seen in figure 3.7.

3.8.1 Discontinuous contour generators

In the case of objects which are self-occluding, the entire object's contour generator cannot be identified with a constant u curve in the parameter space. Continuous parts of the apparent contour which project from locally continuous regions of the object are identified from their image projections (see the coloured curves in figures 3.7,3.8). The ordering and relative scale and position of these curve segments on the surface is associated approximately with their locations on the parameter space. Again we use the freedom in the (u, v) parametrization to choose these as parts of constant u or constant v curves in the parameter space. The position of the constraint curves on the parameter space is flexible, up to ordering, and if scale and position in the mapping of these curve segments are preserved roughly, the reconstruction will adhere well to the apparent contour constraints. Given the parameter-space cuts, a number of new constraints are introduced into the system of equations. The four curves on the banana image in figure 3.8 introduce new constraints of the type outlined above. The left-hand side of the apparent contour is handled just as above (3.17) (note that in this example the CG is non-planar—this is handled perfectly using the previous machinery). Curves 4 and 3 are simply segments of contour generator handled as above, noting that they must not occupy the same $u =$ constant curve in parameter space because they are disjoint on the object. Denote curve 3 as extending from $(u_3, 0)$ to (u_3, v_3), and curve 4 from (u_4, v_4) to $(u_4, 1)$. It is not important where the curve endpoints are placed in the v direction, although it uses grid resolution most effectively if they are placed so that the curves are roughly arc-length parametrized when re-projected into the image.

Curve 2 is a particular class of curve on a 3D object. Because it corresponds to a crease discontinuity, it is visible in the image after it has ceased to be part of the contour generator. Thus we can identify it, and assign it to the same u-constant parameter line as curve 3. This creates a more pleasing parametrization and generates a more plausible 3D model, in conjunction with modelling creases realistically (§3.7.3). Another example can be seen in figure 3.9.

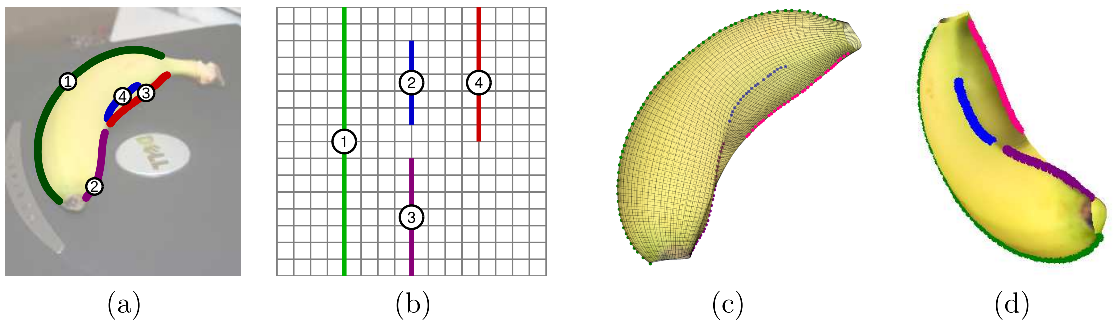


Figure 3.8: Reconstruction from a complex apparent contour. (a) Input image, with user-selected Canny edge chains. Curve 1 is a simple segment of the apparent contour. Curves 3 and 4 represent a continuous segment of the apparent contour which corresponds to a discontinuity in the 3D contour generator. Curve 2 is a crease discontinuity which extends curve 3. (b) These curves may be laid out in parameter space so as to preserve their incidence relationships. (c,d) Projections of the recovered 3D surface from two views.

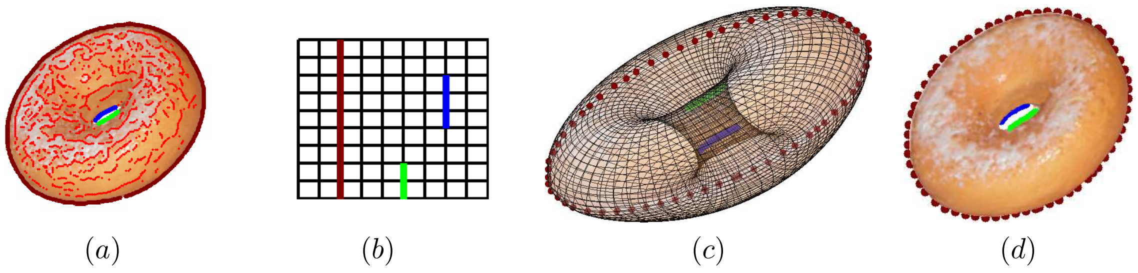


Figure 3.9: Discontinuous contour generator, torus topology. (a) Donut image, with Canny edgels superimposed. The three segments of the contour generator are shown thickened. (b) Contour generator segments in parameter space. (c) Inflated surface, contour generator visible through transparency. (d) Textured model.

3.9 Complex topology

One of the challenges in modelling shapes is dealing with complex topology. An object surface can take a variety of shapes ranging from Monge patches (see §2.2 for discussion about surface representations) to 3D closed surfaces which may even have holes in them. We show how to model genus 0 surfaces like the fish and genus 1 like the teapot. The only change between the schemes is the setting up of continuity around the ends of the (u, v) parameter space. We can switch between cylindrical and toroidal topology easily. By maintaining connectivity-graphs like Zhang *et al.* [211], one could model higher genus surfaces, similarly. Topology does not make a difference to the quality of the output shape.

The parametric surface representation (see §2.2) allows us to represent surfaces of different topology and shape. As with the contour generator, topology can be represented quite readily by various cuttings of the parameter space. Although somewhat complex, these cuttings need be worked out only once per topological class. Figure 3.10 illustrates the various cases, and we consider some examples to show how these are implemented in our optimization framework.

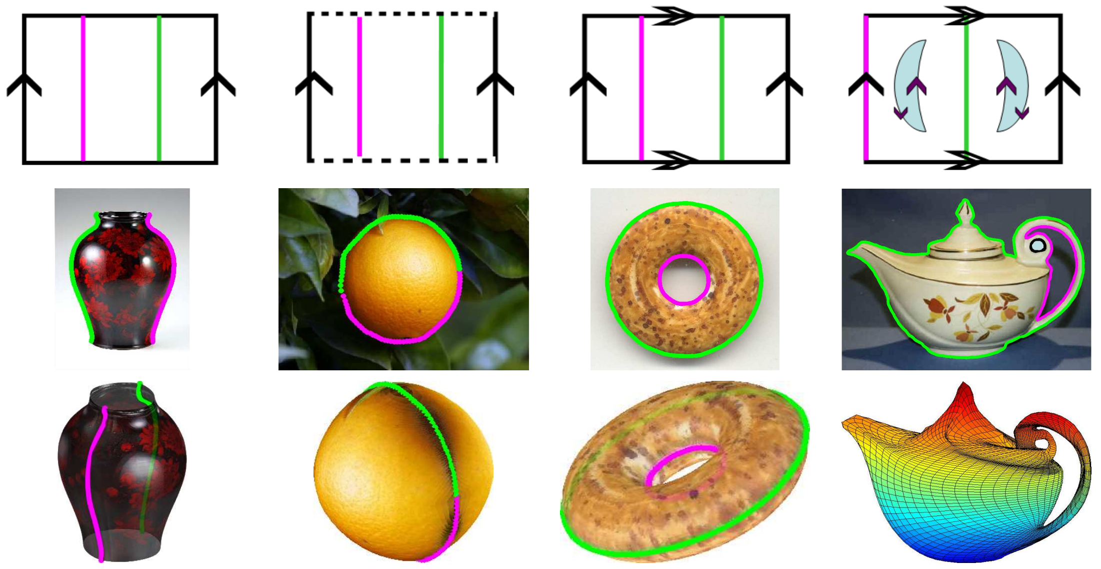
For the **cylindrical** topology of figure 3.10, the essential constraint is that points on the $u = 0$ curve (i.e. the 3D curve $\{\mathbf{r}(0, v) | 0 \leq v \leq 1\}$) are neighbours of points on the $u = 1$ curve. In a continuous parametrization, it would be natural to implement this as a set of incidence constraints on the surface and its derivatives as shown:

$$\mathbf{r}(0, v) = \mathbf{r}(1, v) \quad \forall v, \tag{3.26a}$$

$$\mathbf{r}_u(0, v) = \mathbf{r}_u(1, v) \quad \forall v, \tag{3.26b}$$

$$\mathbf{r}_{uu}(0, v) = \mathbf{r}_{uu}(1, v) \quad \forall v, \textit{ etc.} \tag{3.26c}$$

which is again linear in $\mathbf{r}$. In a discrete implementation, this is a waste of $1/m$ of the parameter space and complicates bookkeeping. In practice it is simpler to make $\mathtt{X}(1, j)$ and $\mathtt{X}(m, j)$ neighbours in the derivative computations by modifying the Jacobian operator matrix $\mathtt{C}_u$, as well as the appropriate second derivative operators. For **spherical** topology, these incidence constraints are augmented with the constraints that: (i) $\mathbf{r}(u, 0) = \mathbf{r}(0, 0) \forall u$ which is again linear in $\mathbf{r}$, and is implemented as the $3m$ linear constraints; and (ii) $\mathbf{r}_v(u, v) = \mathbf{r}_v\left(\left(u + \frac{1}{2}\right) \mathtt{mod}\, 1, v\right) \forall u; v \in \{0, 1\}$. The **torus** topology is a straightforward modification of the cylindrical case. Finally, higher genus surfaces require somewhat more profligate use of the parameter space. To make a **genus two** surface requires that two loops of parameter space are identified. In this case, modification of the derivative operator matrices is not for the faint-hearted, and recourse to a simple parameter identification as in (3.26) is probably the best course of action. One such implementation of a genus 2 surface can be seen in figure 3.10. In general a surface of genus $n(\geq 2)$ needs $2n$ holes in the parameter surface. For a logical choice of the loop and a reasonable resolution, the precise choice does not greatly affect surface shape.



(a) Genus 0 (cylindrical) (b) Genus 0 (spherical) (c) Genus 1 (toroidal) (d) Genus 2

Figure 3.10: Topology: The first row shows the parameter space for each of the topologies listed in the bottom row. The arrow types indicate which ends of the parameter space must join with each other. The dotted line in column 2 indicates that all points on that parameter line join at one point. For surfaces of genus > 1, the hole in the surface translates to two holes in the parameter space as shown in column 4. Images and reconstructions of surfaces of the respective topologies are shown on Rows 2 and 3, with the apparent contour constraint represented as green and magenta curves. The genus 2 teapot without texture mapping demonstrates the power of the parametrization

3.9.1 Sensitivity to parameter assignment

At several points we have made special choices of parameter-space points and curves, noting that the precise values chosen will have a minimal effect on the recovered surface. In theory this is true if the energy being minimized is invariant to parametrization, for example surface curvature. In practice we are not minimizing curvature, merely a proxy for it. This proxy computes derivatives in the parametric space and does not take actual 3D distances into account. So the parametrization does affect the object shape. We illustrate that this effect is small by perturbing our assignments in the parameter space by approximately 20% of their original assignment values for the banana example. The models produced using identical inflation are shown in figure 3.11. Varying the parameter assignment results in different but all convincing models. The sensitivity to perturbation also depends on the object complexity and the availability of constraints. One workaround is to use derivatives weighted by 3D distances and find the optimal surface by iteratively fitting and reparametrizing the surface.

The object shape is closely related to the density of parametrization. For an object with large variations in curvature, parametrizing sufficiently densely that the high-curvature sections are modelled will waste effort in densely sampling near-planar areas. These questions are among those addressed by Gu *et al.*'s "geometry image" work [82] (also see § 2.2).

3.10 Implementation details

The optimization involved is equivalent to solving a linear equation with largely sparse matrices. This is generally fast, *e.g.* solving the constrained optimization problem on a $64 \times 64 \times 3$ grid takes under 1 second. In certain cases, we can exclusively solve for the x and y mapping before solving for z, which means that the system can be solved for fewer unknowns at a time, which ensures higher speed. The MATLAB backslash operator is used, which internally uses the LU factorization method. The texture mapping used on any point on the object surface is simply the image colour at its projection in the image. As the surface curves away from the user at the CG, this can result in a patchy seam. We regenerate texture in this band using available methods for texture synthesis such as [48, 49].

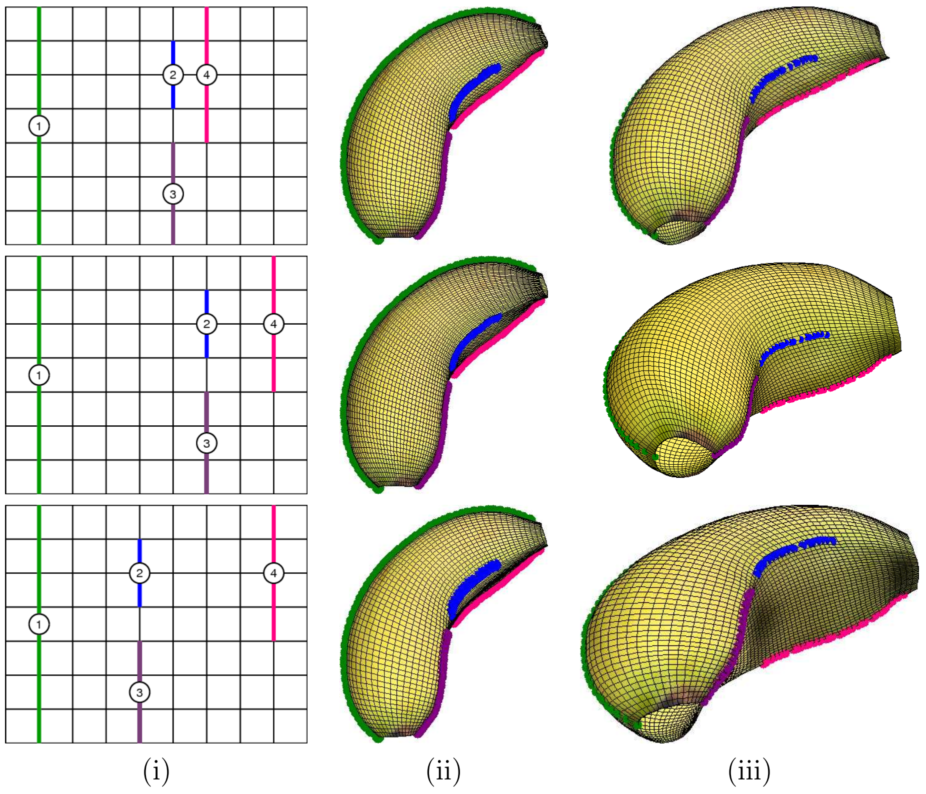


Figure 3.11: Perturbing the parameter space assignment: Column (i) shows the parameter assignment for each case. Columns (ii) & (iii) show the front and oblique views of the model respectively.

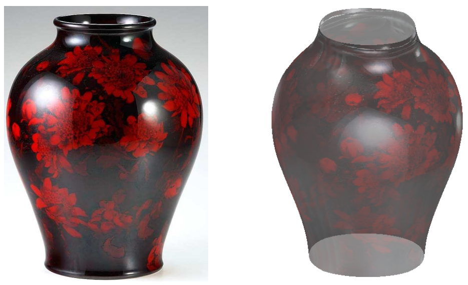
3.11 Summary

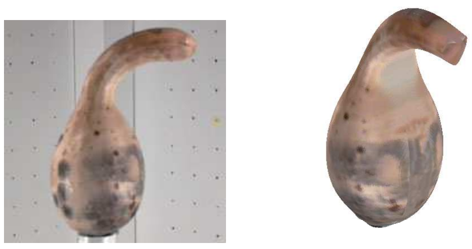
We have shown how truly 3D freeform, surface modelling from silhouettes can be achieved (algo. 3.1). Silhouette-based information can be used to constrain our surface solution. Various other image-based cues, such as user-input, images of creases and multi-local singularities (cusps), discontinuities along the surface can be introduced as simple linear constraints in our problem. The under-constrained reconstruction is solved by a quadratic smoothness function in the presence of these linear constraints. The globally optimal surface can be found. We use a parametric surface representation and complex topology and discontinuities (caused by self-occlusion *e.g.* torus swallowtails) are handled by careful associating or cutting of the parameter domain respectively. Previous energy-

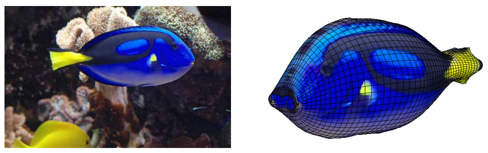
Algorithm 3.1 Summary

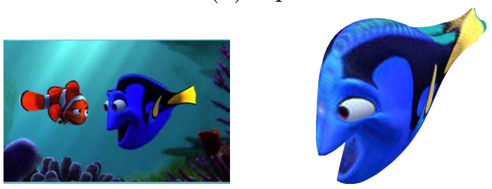
```
1: Get object silhouette constraints (automatic or user-drawn), topological constraints
   (genus $0, 1$, etc.) and features such as creases and multi-local singularities. Construct
   smoothness matrices (§ 3.4) appropriate to these image-based constraints.
2: if cylindrical inflation then
3:   Derive inflation constraints from silhouette (§ 3.7.1.1). Solve for surface.
4: else if Distance transform or Inflation curve based inflation then
5:   Solve for surface to get $(u, v) \to (x, y, z)$ mapping (§ 3.7.1.2,§ 3.7.1.3)). Now add
     inflation constraints using this mapping. Solve for surface.
6: end if
7: while Spillage do
8:   Allow user to perform drag-and-drop corrections (§ 3.7.1.4). Solve for surface.
9: end while
10: Create texture map for the surface from its image projection. Texture fill the seams if
    necessary and display texture-mapped surface.
```

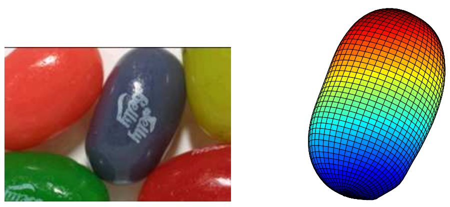
based approaches such as Terzopoulos *et al.*, [191], have relied on iterative optimization strategies which frequently fell into local minima or mangled the surface mesh; and previous ad-hoc approaches could not guarantee to maintain the silhouette without complex polygon book-keeping. A recap is seen in figure 3.12 and a few more examples in figure 1.4.

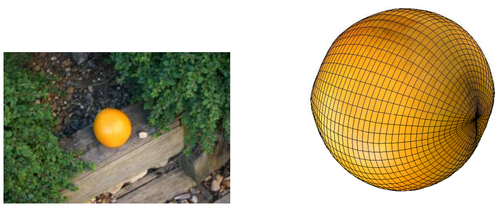

(a) Vase

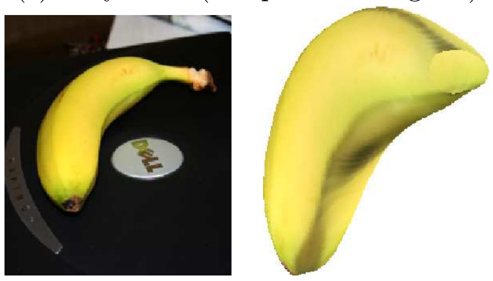

(b) Squash


(c) Monterey fish


(d) Dory: Finding Nemo (includes concavity)


(e) Jelly bean (compare with fig 2.5)


(f) Orange


(g) Banana (self-occluding)

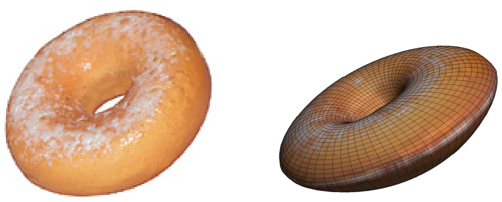

(h) Donut (self-occluding)

Figure 3.12: 3D models of varying complexity produced by our method are shown. These range from cylindrical topologies such as the vase (a), squash (b) and the simple monterey fish (c) to objects with concavities in their silhouette such as a more complex monterey fish in (d). Objects with spherical topologies can be handled much more easily (e,f) than previous work in fig 2.5. Objects with self-occlusions such as the banana (g) and donut (h) can also be easily reconstructed. (See figure 1.4 for more examples).

Chapter 4

Learning Wireframe Class Models of 3D Object Categories from 2D data

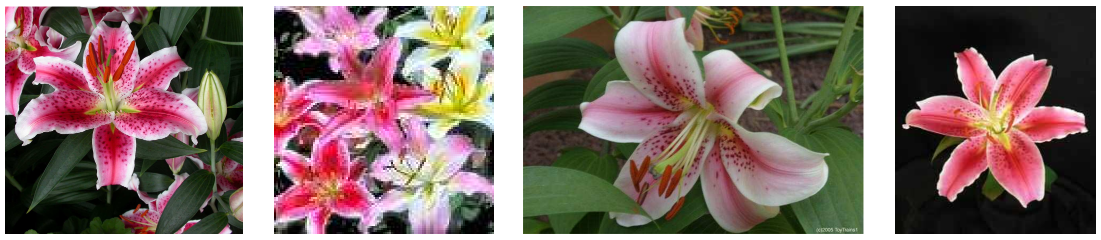


Figure 4.1: A deformable object class: The lily petal may not have a reliable number and nature of features in all instances but has a unique reliable structure. Each instance is photographed in an unknown view, adding to the ambiguity in recovering 3D object structure. The challenge of reconstructing each 3D shape instance and the class shape model is addressed in this chapter.

In Chapter 3, we showed how 3D reconstruction can be performed with generic knowledge about object topology given silhouette based cues from a single image. We now explore the much more challenging problem of learning class-based information (or class-based shape distributions) from image collections.

4.1 Introduction and overview

The typical approach to class-based single view reconstruction treats it with a two-step approach: (i) learn a class-based shape model from several 3D exemplars (*e.g.* a PDM, see § 2.3); (ii) fit the class model to new unseen images to produce plausible 3D shape hypothesis. This two-step approach has been used by many methods such as [15], an implementation of which is shown in figure 1.1 (also discussed in § 2.3.2). Such 3D exemplars are scarce and often inaccessible. While 3D or 2D data captured under controlled conditions (calibration) may be hard to procure, unordered images (without specified calibration, lighting *etc.*) of distinct object instances of a class are aplenty. Community photograph collections are one such example of a rich source of information about the world, particularly when many different photos of the same subject are captured over varying pose and imaging conditions.

In this chapter, we show how to learn a deformable object class from images in a photo collection such as Flickr. Such classes often occur in nature–oak leaves, lily petals (plant types), dolphin (animals), buildings (piecewise planar objects), human faces *etc.* Different 3D instances of a class such as 'lilies' (see figure 4.1), have different shape, colour and texture features as well as projection parameters. However, class-specific features and the presence of common structure, bind the different instances together. This commonality is used to learn deformable object models.

Overview: We start with a formal statement of the deformable object reconstruction problem in § 4.2. The ambiguity in the problem necessitates the use of good priors and regularization as seen in § 4.3. We finally present the objective function and optimization process in 4.4. We present results in two experiments:

1. In § 4.5, we learn deformable wireframe-based object class models from multiple uncalibrated images of photo collections (such as Flickr). We don't have reliable point correspondences or temporally smooth image sequences. Instead of first finding

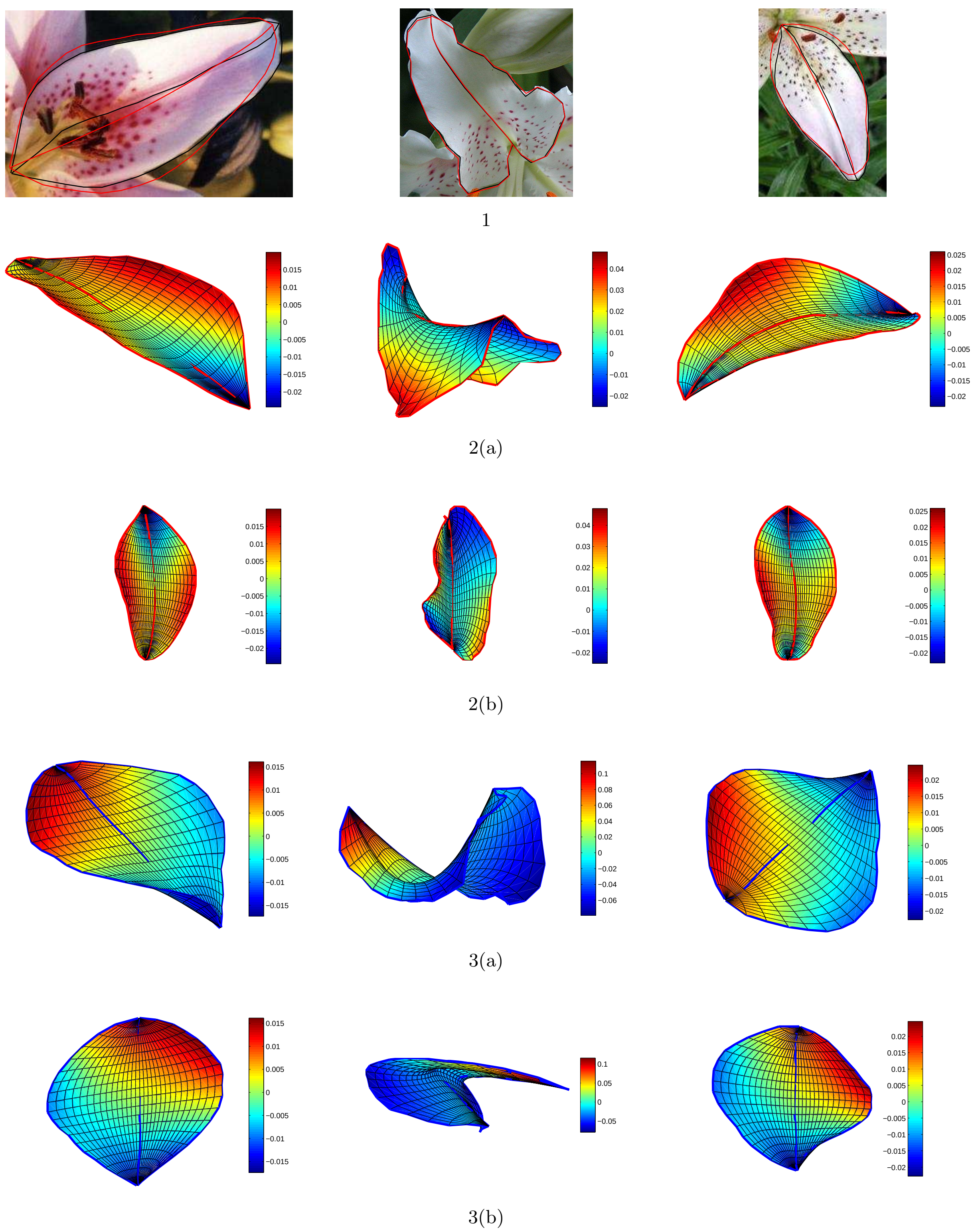


Figure 4.2: A learnt 3D object class: (1) Three examples from a training set of lily shapes are shown. 2(a,b) Two views of 3D wireframes (and interpolated petal surfaces) estimated by our WCM method using flexible correspondences are shown. The reprojected ribs are superimposed over (1) in red. The 3D wireframes and surfaces for standard NRSfM techniques based on fixed uniformly selected correspondences are shown in 3(a,b). The petal surfaces are coloured by their depth (z) values in their own coordinate frame. Our WCM method produces petal-like surfaces (with only 4 bases) while standard NRSfM produces mostly flat petals (except for the centre column where the recovered petal is far too bendy).

correspondences and then recovering the parameters of the deformable shape model as existing methods [22, 23, 194] do , all variables are expressed jointly in one global analytic objective. We show different ways of performing this optimization.

2. In § 4.6, we show how our method can be used with calibrated stereo images to improve generic stereo reconstruction itself for such an object class. In addition to creating better 3D exemplars, we also show how better PDMs can be learnt from calibrated stereo images.

4.2 Problem statement and notation

Given an image collection, we want to simultaneously discover: correspondences, camera matrix parameters, a global parametric shape model, and the parameters fitting it to each image. We want to learn this for shape classes such as the lily petal, which have a wireframe representation. We call the new method WCM, for "wireframe class model". Figure 4.2 shows a preliminary comparison of our method against existing ones.

Each image is of a unique object instance (different lily petal), therefore, despite broadly similar texture, the presence of a particular interest point in one instance cannot be guaranteed to occur in another (figures 4.1,4.3). Therefore, correspondences are defined across higher-level primitives–the petal ribs defining silhouette curves. In order to understand the problem and its solution, we need to introduce the different terms in our optimization formally. We will then express the objective and the notion of variable correspondences.

4.2.1 Input data

We are given N images, each of a distinct lily instance. Features like speckles are unique to each flower, and cannot be correlated across views, as seen in figure 4.3. However, other features such as the silhouette and vein can be reliably identified as a collection of 2D curves, subject to occlusion. Henceforth, these three features *i.e.* the left, right part

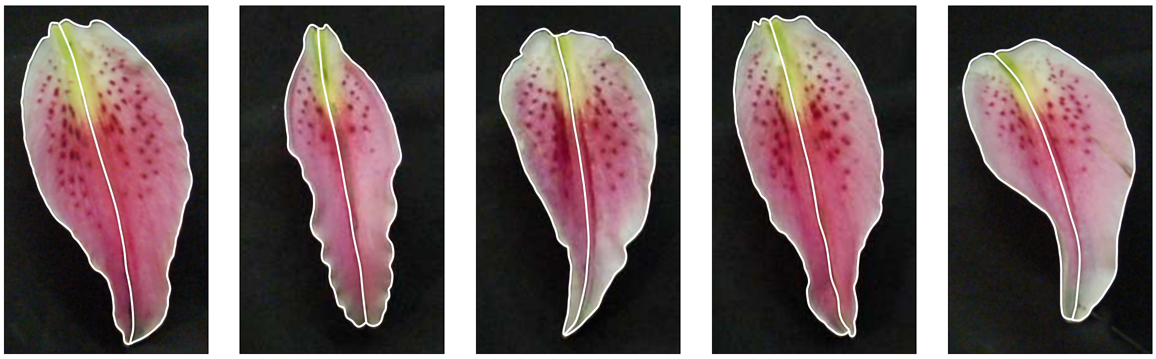


Figure 4.3: Instances of object class: Lily petals have different width, texture, number and type of undulations. Common defining ribs in the silhouette can be seen in white.

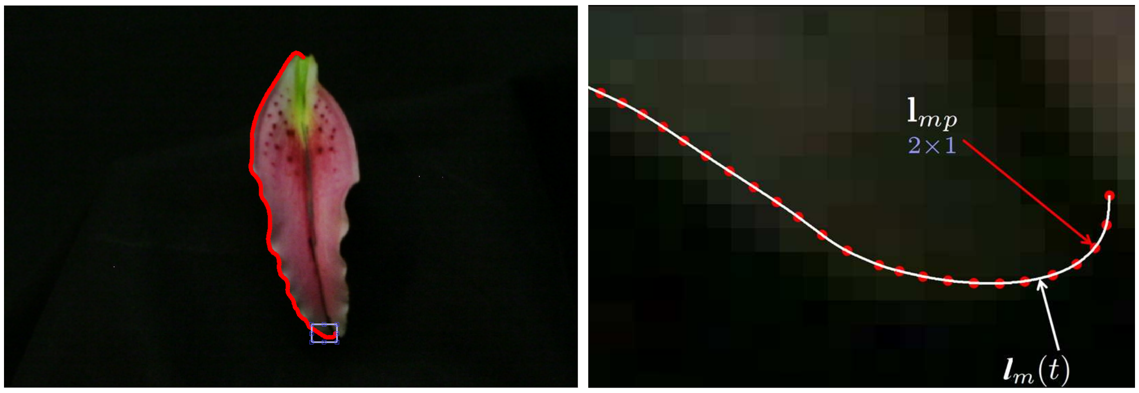


Figure 4.4: Contour input. Contours are represented by edgel chains computed to sub-pixel precision, and spline interpolated, so may be treated as continuous curves $\{\mathbf{w}(t) \mid 1 \leq t \leq P\}$. A part of the petal on the left is zoomed in to reveal the discrete and analytic curve representation.

of the silhouette and the vein will be known as the 'ribs' (illustrated in figure 4.4). In order to simplify the problem, we assume that the flower shape is completely defined given the wireframe of these ribs. Therefore the goal is now to determine the shape of each object instance, where the shape is defined by a 3D wireframe. The image observations are silhouettes of the ribs as observed in figure 4.4. In the experiments reported here, the image based silhouettes are extracted semi-automatically. The user helps to localize the curves by drawing a rough boundary and the curves for the silhouette are latched on automatically to the closest Canny edges. Thus, a smooth silhouette representation faithful to the image information is retrieved. The method is described in detail in A.2.

This is a quick and reliable process.

4.2.2 Curve representation

The annotation yields a set of point observations along the respective ribs in the image. The points along the 3 ribs can be stored in one vector of P points for the image. We assume an equal number of point observations $P/3$ per curve as shown in figure 4.5. Though we know which curves correspond to each other across images, the actual point correspondences are unknown. Dense correspondences are important for reconstruction and representation of the 3D shape. Therefore there is a need for candidate points to able to slide along their ribs until correspondences have been correctly identified across the images. It is useful therefore, to have an analytic representation of the ribs. A piecewise smooth cubic spline representation can be fit to its representative points for each rib. For convenience of exposition we parametrize all curves by a single parameter t, so the set of points traced by the curves in image n maps as following $\{\mathbf{w}_{np} \mid 1 \leq p \leq P\} \rightarrow \{\boldsymbol{\omega}_n(t) \mid 0 \leq t \leq 1\}$ (see figure 4.5). We can now smoothly traverse the parameter space occupied by the ribs instead of being restricted to hopping across the discrete representation (see figure 4.4). Any 2D point on the ribs is now a function of a 1D parameter point and precomputed parameters. Also analytic curves such as these allow computation of smooth derivatives whose importance is discussed in Appendix B.

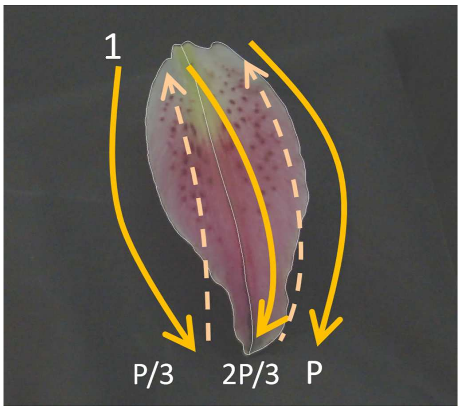


Figure 4.5: Parametrization for lily example: All contours are numbered in the same scheme.

4.2.3 3D model representation

The model for the 3D object class is assumed to lie in a linear subspace (see Bregler *et al.* [23]) of *basis shapes*, each is a matrix $\underset{3\times P}{\mathtt{B}_k}$ [1]where each column $\underset{3\times 1}{\mathbf{b}_{kp}}$ represents a 3D point. We denote by $\mathcal{B}$ the set of shapes $\{\mathtt{B}_k\}_{k=1\ldots K}$. In a particular image, n, the observed instance of the object class is defined by a vector of shape parameters $\boldsymbol{\alpha_n}$, and the 3D shape is denoted $\mathtt{X}_n$, where

$$\underset{3\times P}{\mathtt{X}_n} = \sum_{k=1}^{K} \alpha_{nk} \underset{3\times P}{\mathtt{B}_k} . \tag{4.1}$$

We adopt the convention that $\alpha_{n0} = 1\,\forall n$, so that $\mathtt{B}_0$ behaves like the mean in principal components analysis (PCA).

4.2.4 Camera matrices

A 3D model is projected into the image via a 3×4 camera matrix as shown:

$$\underset{3\times 4}{\mathtt{P}_n} = \left[\underset{3\times 3}{\mathtt{A}_n} \mid \underset{3\times 1}{\mathbf{T}_n} \right] . \tag{4.2}$$

For $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underset{3\times 4}{\mathtt{P}} \underset{4\times 1}{\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}}$, perspective projection is given by: $\begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} := \pi \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ (more details in Appendix B). Therefore the image projection of the 3D shape is given by:

$$\underset{2\times 1}{\hat{\mathbf{w}}_{np}} = \pi \left(\mathtt{A}_n \sum_{k=1}^{K} \alpha_{nk} \cdot \underset{3\times 1}{\mathbf{b}_{kp}} + \underset{3\times 1}{\mathbf{t}_n} \right) . \tag{4.3}$$

Assuming a full projection matrix (11 d.o.f, or degrees of freedom) can make it difficult to find sensible solutions. We want a simpler and more constrained camera representation to solve the system more effectively. In many cases, a scaled-rotation camera matrix (7

[1]It is not necessary that the number of points in the basis shape be the same as the number of image curve control points, but it loses no generality to make it so, and makes the new model "backwards compatible" with nonrigid SfM models.

d.o.f) shown can be employed:

$$\underset{3\times 4}{\mathtt{P}_n} = \begin{bmatrix} s_n \cdot \underset{3\times 3}{\mathtt{R}_n} & \underset{3\times 1}{\mathbf{t}_n} \end{bmatrix}. \tag{4.4}$$

If appropriately used, such an assumption constrains the camera sensibly without seriously affecting the plausibility of the resulting solutions. The matrix can be expressed as a function of an underlying set of Q parameters (equals the d.o.f), which determine the entries of the matrix. The scale is determined by one such parameter s_n. The rotation and translation each have 3 degrees of freedom. In scenes where the depth of the object is much smaller than the distance from the camera, a scaled-orthographic camera model (Q = 6) also makes for a good approximation (refer Hartley and Zisserman [84], Forsyth and Ponce [62]). The altered camera projection is unaffected by depth and therefore uses only a sub-matrix of the system above as shown:

$$\underset{3\times 4}{\mathtt{P}_n} = f\left(\underset{Q\times 1}{\boldsymbol{\theta}_n}\right) = \begin{bmatrix} s_n \cdot \underset{2\times 3}{\mathtt{R}_n} & \underset{2\times 1}{\mathbf{t}_n} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}, \quad Q = 6 \text{ for scaled-ortho.} \tag{4.5}$$

In addition to constraining the problem, scaled-rotation (and scaled-orthographic) matrices yield derivatives which are easy to express and compute. This is useful for optimization (see Appendix B).

4.2.5 Objective

In NRSfM (Non-Rigid Structure from Motion), the correspondences between model and image $\hat{\mathbf{w}}_{np}$ are assumed known. The model fit may be assessed by minimizing the reprojection error of the 3D model's projection with respect to the actual image-based correspondences as:

$$E = \sum_{np} \|\underset{2\times 1}{\mathbf{e}_{np}}\|^2 = \sum_{np} \left\| \underset{2\times 1}{\hat{\mathbf{w}}_{np}} - \underset{2\times 1}{\mathbf{w}_{np}} \right\|^2. \tag{4.6}$$

While each $\mathbf{w}_{np}$ is available from the image, the current estimate $\hat{\mathbf{w}}_{np}$, varies with the parameters. The model can be fit by minimizing E *w.r.t* parameters

$$\Theta = \{\{\boldsymbol{\alpha}\}_{1...N}, \{B\}_{1...K}, \{\boldsymbol{\theta}\}_{1..N}\}. \tag{4.7}$$

The energy and its Jacobian (see Appendix B for more detail) can be computed exactly for every configuration of the parameters, facilitating an efficient least-squares (Levenberg-Marquardt) bundle adjustment over the unknowns.

4.2.6 Variable correspondences and inter-curve distances

We want to maintain the notion of (analytic) curves while constructing our models or measuring error. Calculating a new analytic representation for the 3D model through each iteration is difficult during optimization. In 2D however, we do not have to limit ourselves discrete points to represent image-based silhouettes. Corresponding silhouette-based ribs are given across images. These can be converted to analytic form in a one-off computation. Therefore, reprojection error as the difference: $\mathbf{e}_{np}$ (see equation 4.6) can be replaced by a closest-point calculation between the projected rib curves (in the form of discrete projected points) and the analytic image curve as

$$\mathbf{d}_{np}^{\min^2} = \min_t \mathbf{d}_{np}^2(t) = \min_t \left\| \hat{\mathbf{w}}_{np} - \mathbf{w}_n(t) \right\|^2. \tag{4.8}$$

Replacing the error term in (4.6) with (4.8) gives us a reprojection error across images for variable correspondences. This small modification makes the optimization rather more difficult. We shall now discuss the several options for minimizing $D = \sum_{np} \mathbf{d}_{np}^{\min^2}$.

Ways of minimizing D

In our experiments, a number of strategies for minimization of D were considered: distance transform, point-to-spline distance, and augmented bundle adjustment. We now discuss these three methods.

Distance transforms The distance transform approach maintains a distance transform for each training image. For each point on a grid (at some reasonably fine resolution) superimposed over the curve, the closest curve point and its distance is pre-computed. The typical DT stores the closest distances in the form of scalars. However, for the purpose of our least-squares optimization, it is useful to maintain the x, y parts of this distance separately. Thus the DT can be expressed by the following 2 vector:

$$\underset{2\times 1}{\mathrm{DT}_n(\mathbf{x})} = \min_t \|\mathbf{x} - \mathbf{w}_n(t)\|. \tag{4.9}$$

Therefore the explicit computation of the closest point on the image silhouette-curves for each candidate projection point can be avoided. Instead a quick interpolated look-up can be performed on the pre-computed DT to give the following:

$$\mathbf{d}_{np}^{\min} = \mathrm{DT}_n(\hat{\mathbf{w}}). \tag{4.10}$$

The distance transform at a given grid resolution and the derivatives (see Appendix B and also [57]) can be computed offline. This look-up based distance computation is fast and cheap on computational resources. Accuracy depends on the discretization of the DT, but the resolution is expensive on memory resources and often limits the number of images we can train on.

Explicit closest point computation in each function evaluation to measure curve-to-curve distances We can also minimize the actual directional distance between the projected silhouette at any given instant, and the reference silhouette. Bi-directional distance calculation is ideal, however it is much more expensive to compute. The 3D candidate ribs and their 2D silhouettes at any given instant are stored in discrete form. For each discrete 2D candidate silhouette point, the closest point on the analytic reference image curve can be found and distances can be summed up across the candidate silhouette. This implies a minimization for each representative model point in each evaluation of the error function in equation (4.8). Because the image curves are defined as interpolating splines, the closest point to any query point can be efficiently and quite reliably found using Newton-Raphson iterations. However the derivatives of equation 4.8 are not available in closed form as $\hat{\mathbf{w}}_{np}$ itself is unavailable in closed-form. Therefore the only options are to either use expensive explicit iterative closest-point optimization, or faster finite-difference approximations for derivative computation. Details about this optimization can be seen in Appendix B.

Slack approach Consider the image n and vertex p of the candidate rib $\mathbf{X}_{np}$ and its estimated projection in the image $\hat{\mathbf{w}}_{np}$. Let the closest point for $\hat{\mathbf{w}}_{np}$ on the reference image silhouette $\mathbf{w}_n(t)$ be denoted by the parameter $t = t_{np}$. Instead of explicitly computing this in each function evaluation during optimization, we can make it a part of the larger objective. This amounts to increasing the variable space with NP extra parameters t_{np} and rewriting the objective as follows:

$$\min_{\Theta} \sum_{np} \min_{t} d_{np}(t) = \min_{\Theta, t_{11}..t_{NP}} \sum_{np} d_{np}(t). \tag{4.11}$$

We use this approach. This adds NP parameters to the optimization but does not greatly increase the computational load as the variables are uncoupled (read further in Appendix B).

4.3 Priors and point constraints

The number of variables in the problem far exceed the number of image observations. In such situations, the models retrieved from each image are prone to noise and lack the smoothness characteristics of the actual 3D model. For example, a global optimum of the objective can be found by setting $\mathtt{B}_k = 0$ for all k, and choosing $\mathtt{P}_n$ to project the resulting point onto any point on the image curve, *e.g.* $\mathbf{w}_n(0)$. To combat degenerate solutions and noise, regularizers on the 3D shape are required.

Smoothness and Tension The quest for minimal reprojection error can cause degenerate loops and self-entanglement in the 3D curves. Also when using DTs, their discretization can cause the curves to be jagged in 2D and 3D. Therefore having some smoothness regularizer is essential. This smoothness can be simply second-order *smoothness* w.r.t. the curve parameter, also known as *bending energy*, as shown below:

$$E_{\text{bending}} = \sum_n \sum_{p=2}^{P-1} \lambda_p \|\mathbf{X}_{n,p-1} - 2\mathbf{X}_{n,p} + \mathbf{X}_{n,p+1}\|^2. \tag{4.12}$$

In addition to smoothness, we also want the points on the curve to be equally spaced along the curve, or in other words, we want the 3D rib curves to have unit speed parametrization. To encourage this, we include a first-order *tension* term.[2]

These priors of smoothness and tension are therefore represented by the following regularizer on $\mathtt{X}$, and hence on $\mathtt{B}$:

$$\begin{aligned} E_{\text{smooth}} = \sum_n \sum_{p=2}^{P-1} & \underbrace{\lambda_p \|\mathbf{X}_{n,p-1} - 2\mathbf{X}_{n,p} + \mathbf{X}_{n,p+1}\|^2}_{E_{\text{bending}}} + \\ & \underbrace{\psi_p \|\mathbf{X}_{n,p-1} - \mathbf{X}_{n,p+1}\|^2}_{E_{\text{tension}}}, \end{aligned} \tag{4.13}$$

where the parameters λ_p, ψ_p switch off the regularizers at the discontinuities in parametrization (e.g. $P/3$ in figure 4.5). They also additionally allow these to be relatively weighted depending on the problem specifics.

[2]For a curve which has no other constraints except that its ends are fixed, such a tension term causes the curve to stretch out tautly as a straight line, with the representative points on the curve being at equal distances.

Coincidence The end points of the 3D ribs are not fixed in space and the tension term can causes the ribs to contract. The ribs being completely collapsed to a point which projects somewhere on each image's silhouette leads to a degenerate but valid solution. This must be prevented in a principled manner. In our example images, the tip and base of the petal are identifiable in many views, and can be included as conventional end-point constraints. However, there is observation error associated with these. A weaker, but more principled constraint is to encourage the tips and the base of each of the 3 ribs of a petal to be coincident in 3D. Therefore, a coherent petal in 3D with its veins meeting at the ends leads to coherent projections in images too. This regularization in 3D on the ribs allows them to find the best base-tip projections in the images despite observation error and without manual annotation. The smoothness regularizer (4.13) can now be used without degeneracies. Using the scheme in figure 4.5 the following *coincidence* term can be added to the optimization. $\chi_{\text{base}}, \chi_{\text{tips}}$ control the relative weights accorded to the coincidence at the base and tips of the petal:

$$E_{\text{coincident}} = \sum_n \begin{array}{c} \chi_{\text{base}} \left(\|\mathbf{X}_{n1} - \mathbf{X}_{n\frac{P}{3}+1}\|^2 + \|\mathbf{X}_{n1} - \mathbf{X}_{n\frac{2P}{3}+1}\|^2 \right) \\ + \\ \chi_{\text{tips}} \left(\|\mathbf{X}_{n\frac{P}{3}} - \mathbf{X}_{nP}\|^2 + \|\mathbf{X}_{n\frac{P}{3}} - \mathbf{X}_{n\frac{2P}{3}}\|^2 \right). \end{array} \tag{4.14}$$

4.4 The total objective function

Combining these terms gives our primary objective:

$$E^{\text{wcm}}(\Theta, t_{11}, ..., t_{np}) = \sum_{np} \mathbf{d}^2_{np}(t) + E_{\text{coincident}} + E_{\text{smooth}}. \tag{4.15}$$

The actual optimization procedure is discussed in fine detail in Appendix B. Basically the Levenberg-Marquardt least-squares optimization used here needs a residual vector which contains all the elements of the summation defined in equation (4.15). The introduction of the regularizers (4.13,4.14) introduces new terms to the reprojection-error based residual vector defined in Appendix B. This in turn results in the addition of new blocks to the Jacobian as shown in figure 4.6.

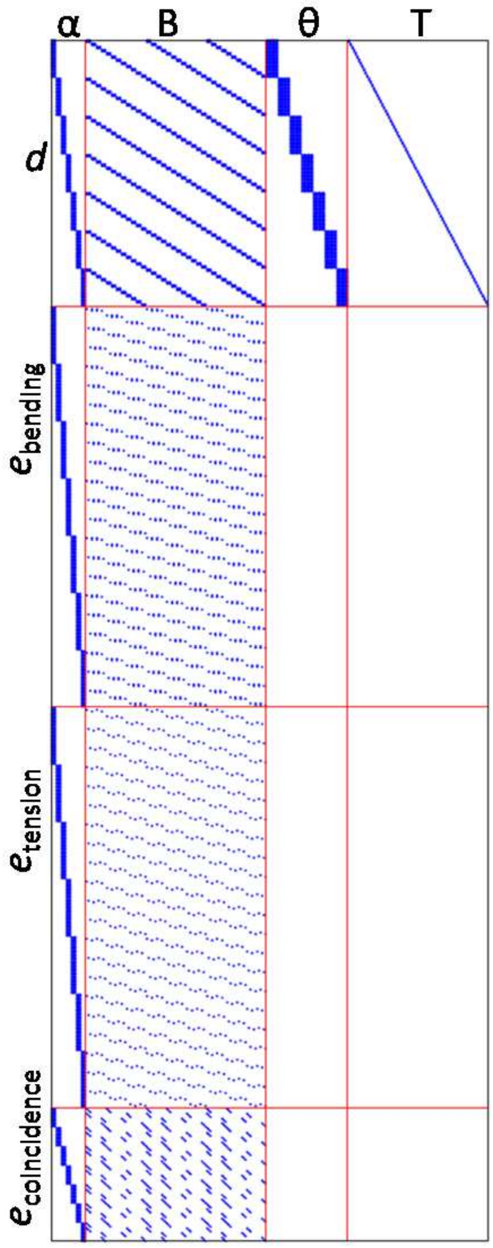
4.5 Experiments I: Flickr

The goal is to compare deformable object reconstruction from fixed correspondences (Optimizing (4.11) with fixed $t_{np} \, \forall n, p$ is equivalent to a regularized version of nonrigid Structure from Motion) with our new method (Wireframe Class Models WCM) which includes correspondence finding within the optimization (4.15). In this experiment, we do not have the ability to compare with ground truth. We will show our contribution in terms of the theory and the qualitative evaluation of the results clearly show that our method performs better.

Data The first set of experiments deals with the example in figure 4.2. A collection of 56 "lily" photos were downloaded from Flickr, and manually annotated as described above, to produce a three-rib curve for each view. The curves were sampled at unit speed into 20 samples per rib, so $P = 60$. The optimized code (as discussed in Appendix B) can handle much higher resolutions but at the expense of compute time.

Results The experiment is performed for $N = 56$ images, each model having $P = 60$ vertices and $K = 1 \dots 4$ bases. The projection matrices $\mathtt{P}_n$ are rarely completely unknown, so may be parametrized by fewer than 12 variables. When the camera instrinsics are known we use a 7-parameter similarity transform (unnormalized quaternion and translation); and for distant features, an affine camera (8 parameters, third row of $\mathtt{P}_n$ is $[0\,0\,0\,1]$ for all n).

The key to any large optimization is in the choice of initialization, and in having a hierarchical minimization strategy. If we were to start the full optimization over all parameters, it would almost certainly not reach a good optimum. This we relax towards the full solution by varying K from 1 to the desired target, and for each K, we first



J	$\alpha(NK)$	$B(3KP)$	$\theta(7N)$	$T(NP)$
$\mathbf{d}$	$2PNK$	$6PNK$	$14NP$	$2NP$
$\mathbf{e}_{\text{bending}}$	$3PNK$	$\approx 9PNK$	0	0
$\mathbf{e}_{\text{tension}}$	$3PNK$	$\approx 9PNK$	0	0
$\mathbf{e}_{\text{coincidence}}$	$12PNK$	$72NK$	0	0

Figure 4.6: Jacobian structure: The variables are plotted along horizontal while the terms contributing to the objective are along the vertical.

K	standard regularized NRSfM for fixed correspondences				our variable correspondence WCM			
	Timings		Errors		Timings		Errors	
	optimization time from $K-1$ to K	time per 100 func evals	Reproj error: pixels per point	Objective func (averaged per point)	optimization time from $K-1$ to K	time per 100 func evals	Reproj error: pixels per point	Objective func (averaged per point)
1	5 minutes	6.42 s	12.16	0.4247	1 day	6.50 s	10.59	3.6974
2	0.5 minutes	7.23 s	9.22	0.3219	3 days	7.21 s	8.99	0.31
3	2 minutes	7.75 s	7.56	0.26	3 days	7.79 s	7.19	0.25
4	2 minutes	9.41 s	5.57	0.19	1 day	9.36 s	5.38	0.18

Table 4.1: Table showing approximate times for optimization using the competing methods ($N = 56, P = 60$ and K as shown). The total optimization times can be sensitive to initialization, therefore we also show the timings per 100 function evaluations. As seen above the WCM method achieves a better minimum. Surprisingly it does better even on the reprojection error.

estimate the `P` matrices, then the bases, with fixed correspondences, and finally vary the correspondences t_{np}. Optimization time increases with the number of bases K and points P but also depends on the initialization and nearest minimum available (see table 4.1). For effective optimization the sparsity in the derivative computations is utilized, as discussed in Appendix B.

The number of terms in the residual is $7NPK + 12N$. The NRSfM (with $NK + 7N + 3KP + NP$ variables) produces rather flat reconstructions, at best. In contrast, WCM (with $NK + 7N + 3KP$ variables) produces moderately realistic 3D models with reassuringly little parameter tuning, which can be used to predict the shape of examples without re-fitting the entire model or resorting to even more complex priors. The results of the two competing methods can be compared in figure 4.2. The performance is summarized further in table 4.1. A gallery of many more images and their reconstructions by our method can be viewed in figures 4.7,4.8 and 4.9. Despite the NP extra variables, the WCM approach takes roughly the same time per function evaluation. Each K is initialized from the results of the previous stage $(K-1)$. Depending on how close the starting point is to the minimum, the total time taken by each optimization stage varies. In general, the WCM method takes a lot longer to converge than NRSfM. The ability to optimize the problem over a larger domain (inclusive of correspondences), enables the WCM method to reach a better minimum. Interestingly the flexibility also results in reduced reprojection error. While the per pixel reduction may seem small, the sum over all the images and points makes for a considerable saving.

Algorithm 4.1 Experiment I: Summary

```
1: Take dataset of images and annotated petal ribs. Fit piecewise splines to the petal ribs
   in each image (§ 4.2.2).
2: Represent correspondences ($\mathbf{w}_{np}$) by the parameters $t_{np}$ on spline-based ribs. These
   are initialized by $P$ equidistant points on the petal ribs in each of the $N$ images.
3: For ($k = 1$) assume rigid body. Bundle adjustment (reprojection error combined with
   smoothness and petal-specific coincidence regularization, see § 4.3, § 4.4) is performed
   over camera parameters, rigid petal and correspondences.
4: for $k = 2 \cdots K$ do
5:    The object is represented by $k$ bases. Use results from $k - 1$ to initialize system for
      $k$ bases.
6:    Perform bundle adjust over camera parameters, bases, shape-fitting coefficients and
      the correspondences. The optimization is performed over all parameters including
      the parameters corresponding to the $k - 1$ initialization.
7: end for
8: Tune the hyperparameters $\lambda, \psi, \chi$ according to best visual quality.
```

4.6 Experiments II: Stereo reconstruction

Problem setup: The goal is to compare two methods of building deformable shape models: (i) a standard PDM based method using 3D shape exemplars built using calibrated stereo frameworks; (ii) The WCM method adapted to build better deformable shape models for stereo reconstruction and PDMs. We will call this new method the sWCM (s–stereo).

In the case of learning a class model from stereo data, the conventional approach would be to reconstruct a 3D wireframe for each set of stereo images, and then to model the 3D distribution using PCA. However this does not take into account the very non-isotropic

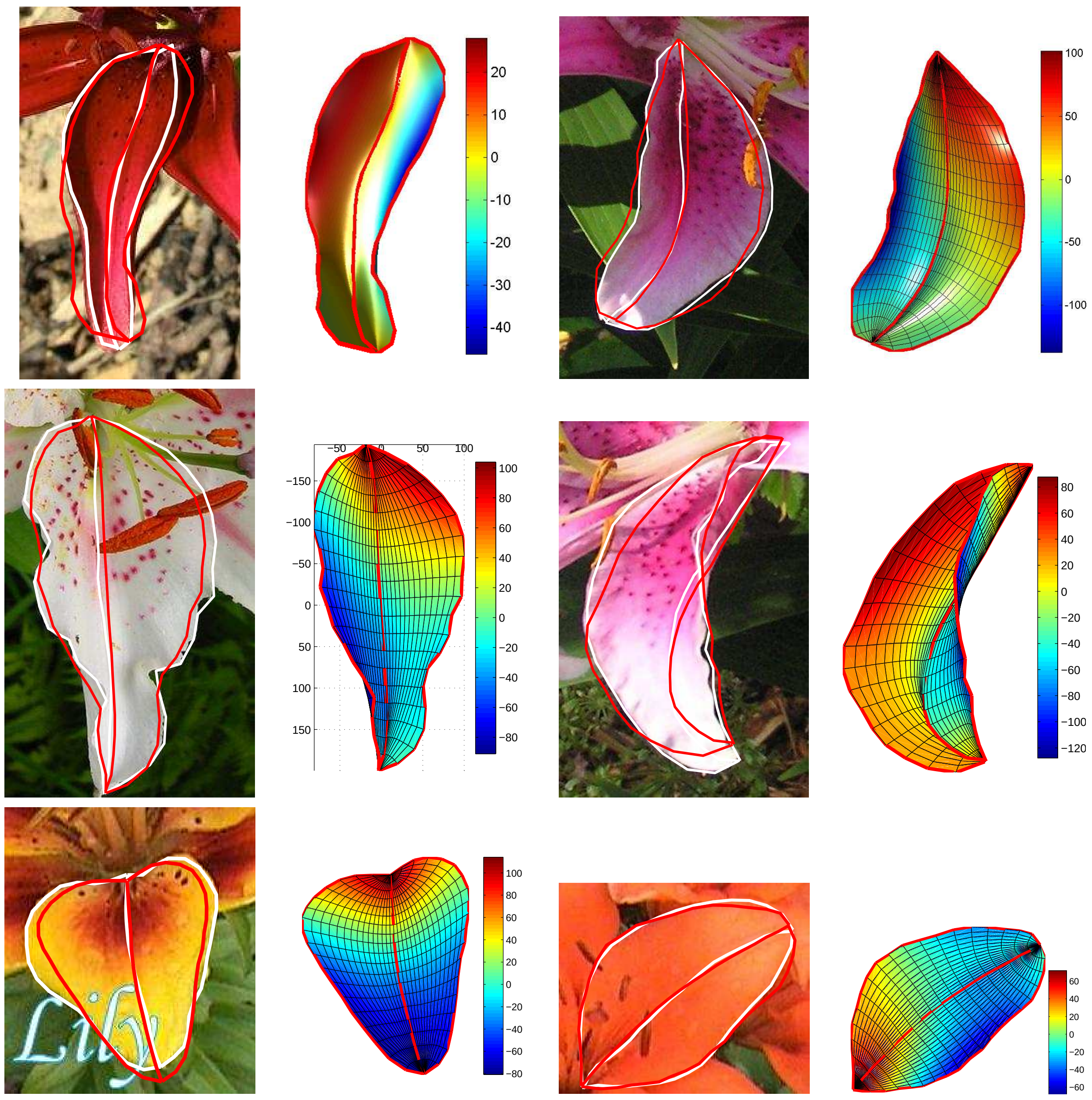


Figure 4.7: Gallery I: Some images from the dataset and their reconstructions are plotted. Surface colour corresponds to surface depth. Some images are plotted without lines or with lighting to highlight the shape. Coordinate axes give an idea about the relative petal size in x, y, z.

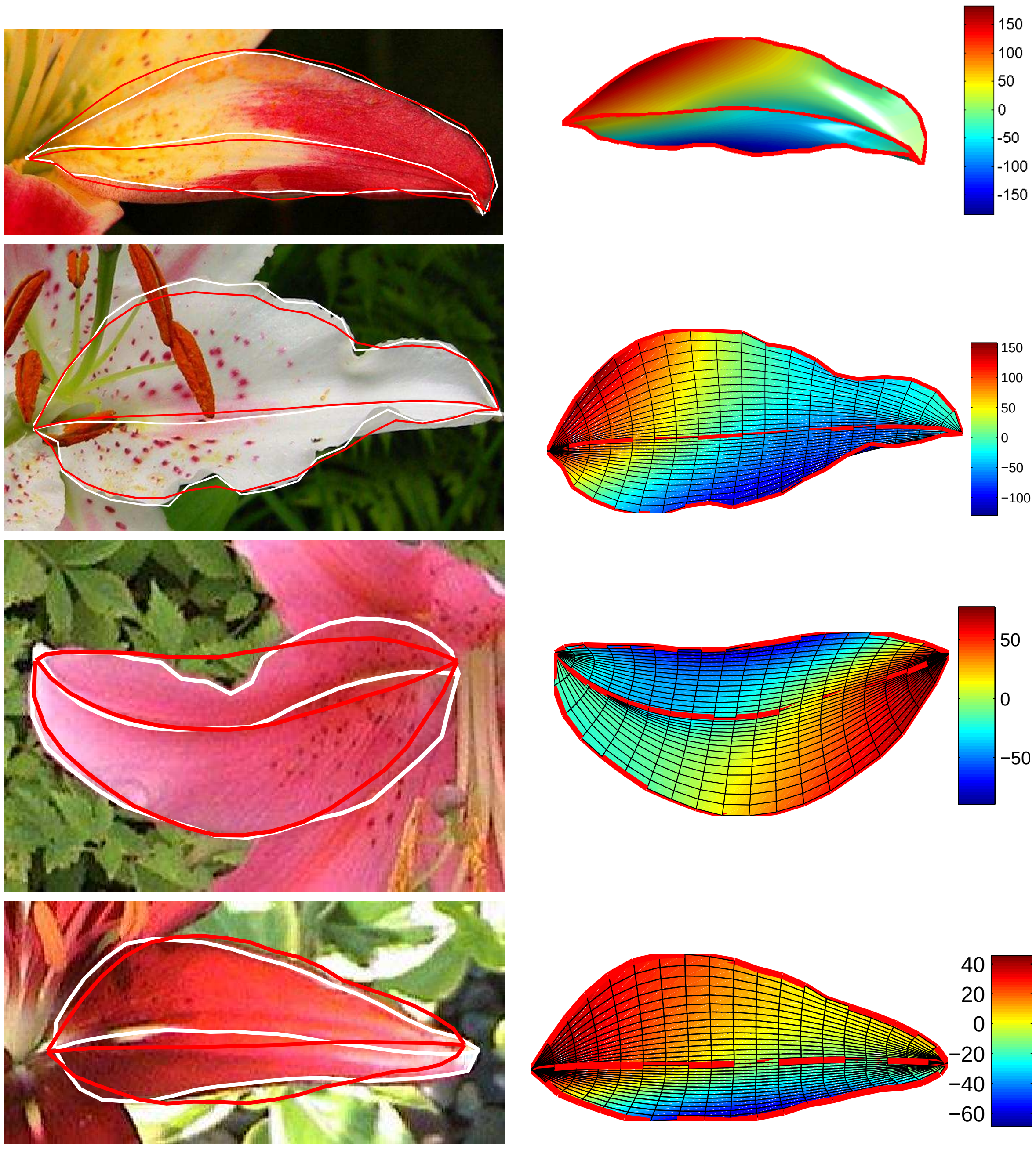


Figure 4.8: Gallery II: Some images from the dataset and their reconstructions are plotted. Surface colour corresponds to surface depth. Some images are plotted without lines or with lighting to highlight the shape.

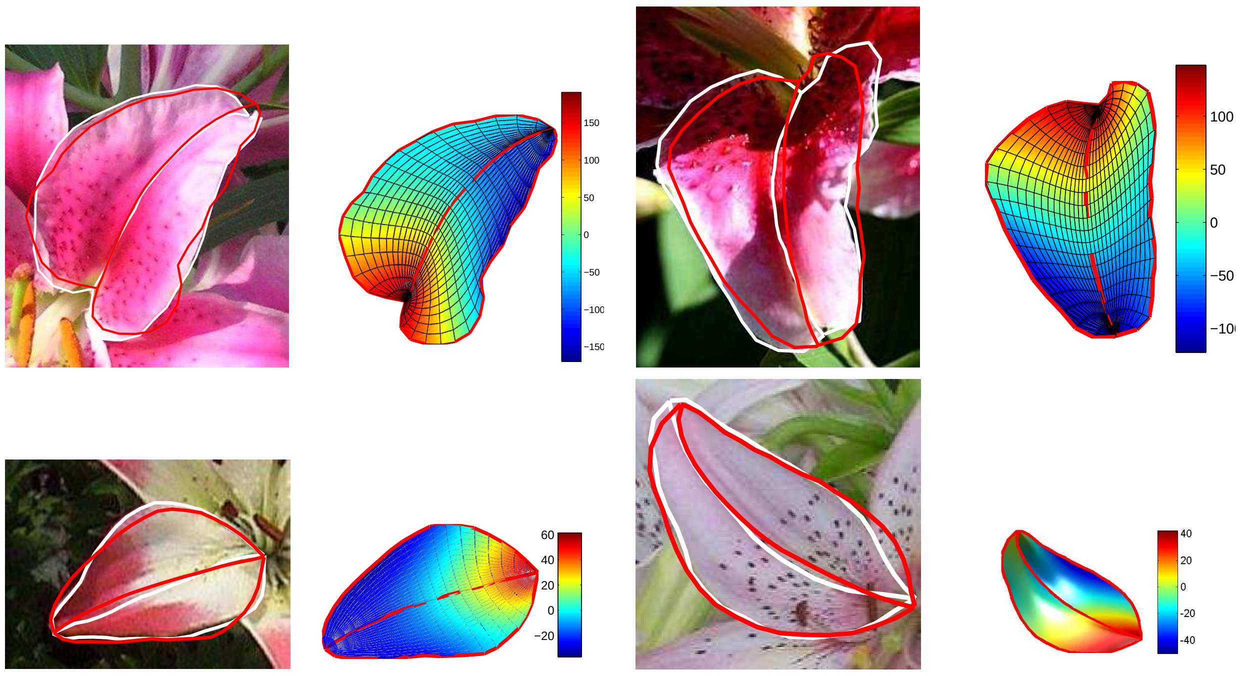


Figure 4.9: Gallery III: Some images from the dataset and their reconstructions are plotted. Surface colour corresponds to surface depth. Some images are plotted without lines or with lighting to highlight the shape.

uncertainties in the wireframe coordinates when the model has been acquired using stereo. PCA with anisotropic data uncertainties is a rather harder problem than simple PCA, and will at best allow the use of Gaussian approximations to the uncertainties. Any error in registration will add to errors in the model. There is therefore, the need to investigate better methods for PDM building.

Data: Multiple images of an object are captured simultaneously using a calibrated stereo rig. In particular $M = 20$ object instances are taken, each of which is photographed by a set of 2 cameras, with matrices $\mathtt{P}_l, \mathtt{P}_r$. Reliable reconstruction in the presence of the minimal two images is challenging. Different methods for 3D reconstruction of the object are explored and the best one is used for reconstruction of exemplars $\mathtt{X}_m$. As each petal is captured in a different frame of reference, a similarity transformation $\mathtt{H}_m$ relates the different exemplars and must be estimated. Similar to § 4.2.1, image based curves for the left and right images of the m^{th} stereo pair are captured by piecewise spline representations stored in $\boldsymbol{\omega}_{rm}(t), \boldsymbol{\omega}_{lm}(t)$ (see Appendix B for more detail).

The PDM approach assumes that the 3D models for an object are generated by a distribution [37]. We continue to use the assumption that the 3D models lie in a linear subspace of bases, so that $\mathtt{X}_m = \sum_k \beta_{mk} \cdot \mathtt{B}_k$, (see § 4.2.3). The goal is to recover the best 3D model $\mathtt{X}_m$ for each set of stereo images of the m^{th} petal. The most effective way of building PDMs is then explored. For this it is imperative to recover $\mathtt{H}, \mathtt{B}, \beta$.

The problem is explored in two stages:

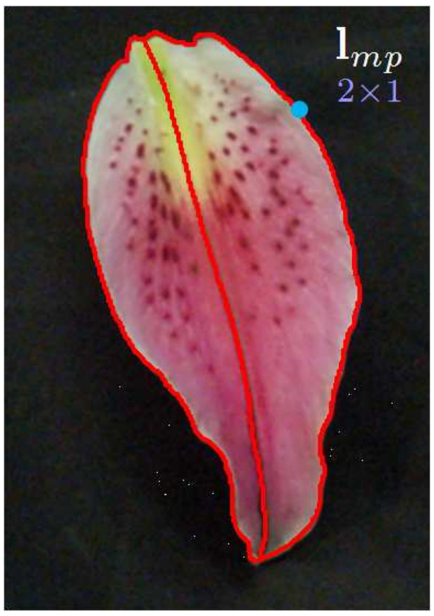
1. We will first examine the most effective methods for reconstructing wireframe based shapes from stereo images.

2. We will then explore how our class-based WCM framework can be used to most effectively extract a model from such exemplars.

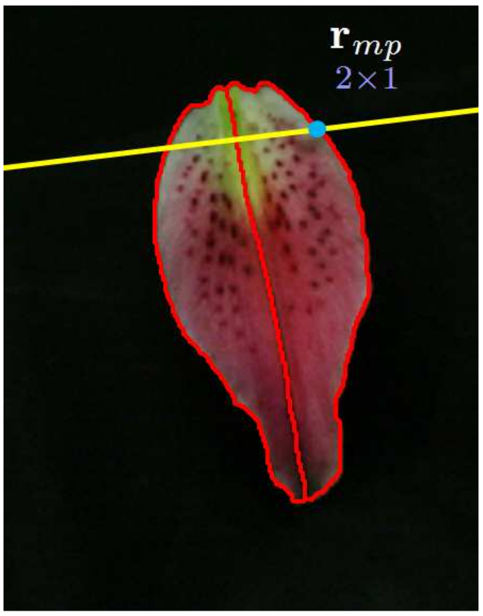
4.6.1 Stereo reconstruction of exemplars

We will explore the different methods of stereo reconstruction for this problem here.

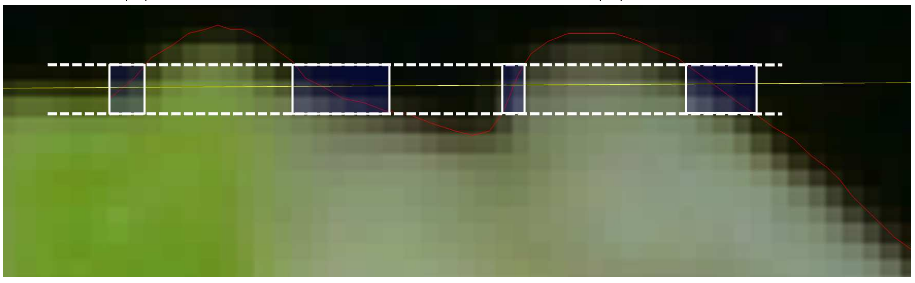
Naïve methods: The simplest method for reconstruction is to find a set of reliable correspondences across image pairs and triangulate to get 3D structure. In the absence of features and only curve-based information usually only sparse correspondences are available. There are some ways of generating more dense correspondences. For *e.g.* for each point $\mathbf{l}_{mp}$ on the rib in the left image, its epipolar line (say $\mathbf{e}$) in the right image can be intersected with the corresponding rib (given by curve ω_{rm}) in the right image to yield its correspondence (see figure 4.10). This is equivalent to minimizing the transfer error defined as:



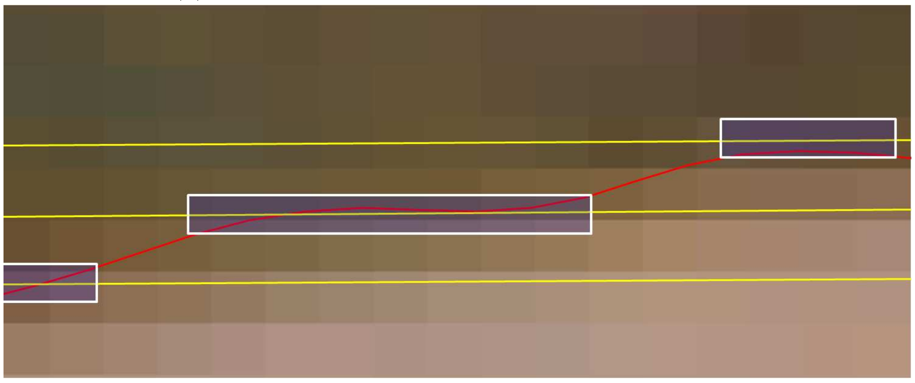
(a) left image



(b) right image



(c) multiple intersections of $\mathbf{e}$ with silhouette



(d) tangency of $\mathbf{e}$ with silhouette

Figure 4.10: Finding correspondences for stereo: To find the correspondence for point p in left image (a) of pair m: $\mathbf{l}_{mp}$, the intersection of its epipolar line $\mathbf{e}$ (in yellow) must be found with the corresponding curve in the right image (b). (c,d) illustrate some problems with such an approach. (c) shows how multiple intersections of $\mathbf{e}$ with the silhouette (in red) leading to ambiguity in correspondences. (d) Epipolar lines can be tangent to the silhouette causing a range of positions on the silhouette to be candidate correspondences.

$$\mathbf{e} = \mathtt{F}_m \begin{bmatrix} \mathbf{l}_{mp} \\ 1 \end{bmatrix}. \tag{4.16}$$

$$\text{Ideally, } \mathbf{e}^\top \begin{bmatrix} \mathbf{r}^*_{mp} \\ 1 \end{bmatrix} = 0, \tag{4.17}$$

$$\text{where } \mathbf{r}^*_{mp} = \boldsymbol{\omega}_{rm}(t^*), \tag{4.18}$$

$$t^* = \underset{t}{\operatorname{argmin}} \left\| \mathbf{e}^\top \begin{bmatrix} \boldsymbol{\omega}_{rm}(t) \\ 1 \end{bmatrix} \right\|^2. \tag{4.19}$$

Note: ω_{rm} is the piecewise cubic spline representation of the silhouette curves in the right image of the m^{th} stereo pair. $\boldsymbol{\omega}_{rm}(t)$ denotes the point represented by a parameter t on this curve. The minimization of this transfer error can be done analytically with the spline-based representation and produces good results except at the ends of the petals, or in curvaceous regions, where the ribs are tangential to the epipolar lines, or where multiple intersections occur with undulated curve silhouette (see figure 4.10).

The minimization of such an error also causes the selection of 3D points that correctly project *somewhere* on the ribs in the left and right images, but may not be actual valid points on the petal. There may also be situations where due to observation error, there is no intersection with the image curve. A combination of these factors result in the 3D reconstructions with stray vertices as shown in row 1 of figure 4.11.

Therefore, the correspondences must have some properties to prevent such degeneracies and tackle noise. The problems due to multiple intersections, tangencies *etc.* can be addressed by using techniques such as dynamic programming.

DP approach: Valid correspondences across two curves must display properties such as monotonicity (Roy and Cox [164]). In other words, the criss-crossing seen in (1,a) figure 4.11 must be prevented. Given that the ribs in the left and right images are sampled at a reasonably fine resolution–$\mathbf{l}_u, u \in [1, U], \mathbf{r}_v, v \in [1, V]$, the problem of finding correspondences can be recast as a labelling problem. For each point $\mathbf{l}_u$ on the discretized left curve, as before, we want to find the corresponding point (label) $\mathbf{r}_v$ on the discretized right image curve that minimizes transfer error. Additionally we penalize the lack of monotonicity.

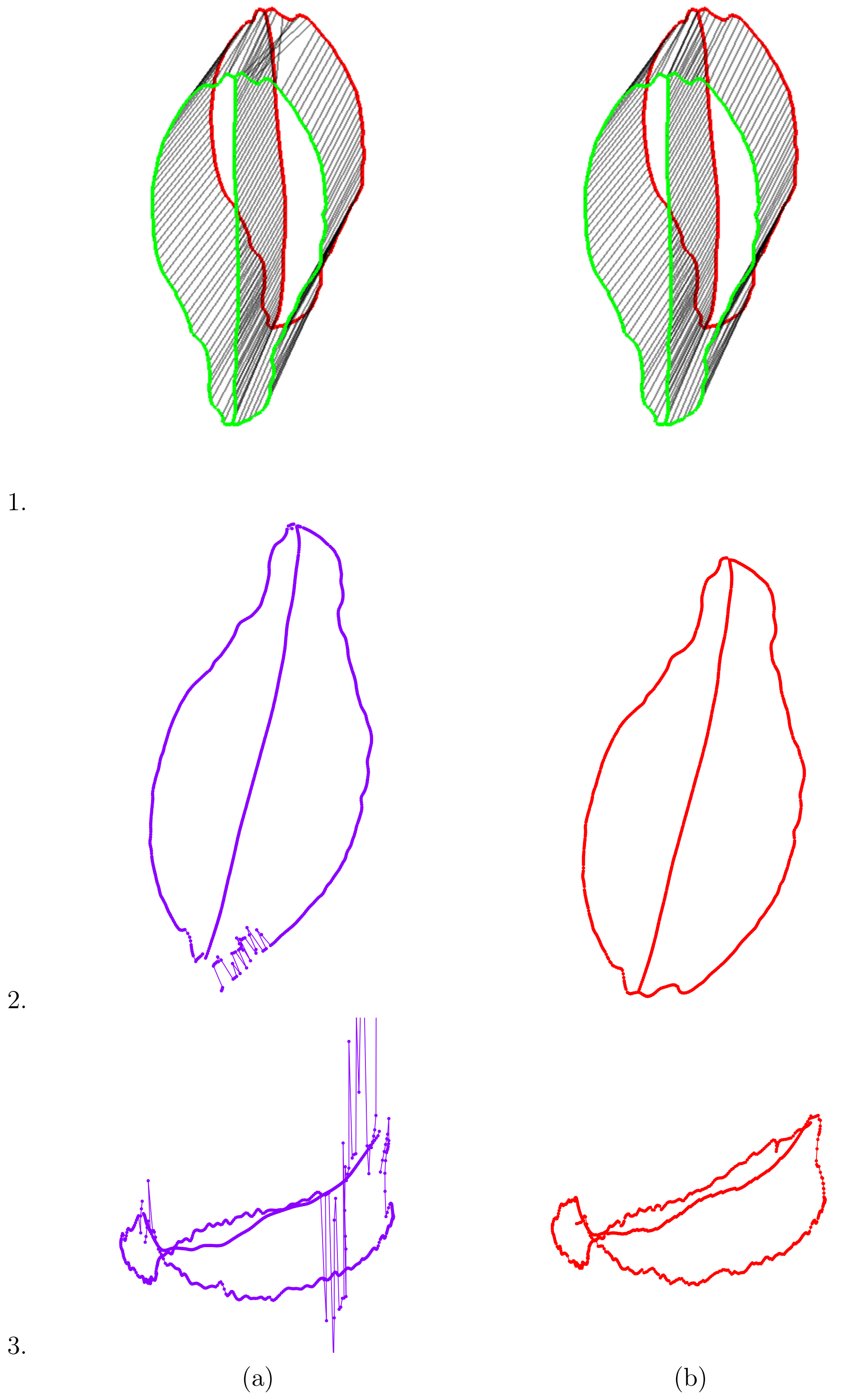


Figure 4.11: Naïve transfer error minimization *vs.* DP based correspondences: 1(a,b) show the correspondences between the left (green) silhouette and the right (red) silhouette. The naive approach leads to ambiguity and the violation of monotonicity in the correspondences (1,a). The resulting reconstruction $(2-3,a)$ contains vertices that minimize reprojection error but do not belong to the petal. With the regularized DP-based approach this is solved (1,b) and a sensible petal $(2-3,$ b) is recovered.

On moving forward from point u_1 to u_2 in the left image, the correspondences for them v_1, v_2 should also result in moving along the same direction. This cost is summarized in D, in (4.21). The total objective (4.22) is a combination of transfer error (4.20) and regularization in the form of monotonicity (4.21) similar to Ohta and Kanade [141]. The total objective (4.22) is expressed as a cost of assigning label v to node u and includes the cost incurred until node $u-1$. Mathematically, the problem can be expressed as ϕ and this can be solved by a simple dynamic program as shown.

$$C(u,v) = \mathbf{e}_u^\top \mathbf{r}_v, \text{ where } \mathbf{e}_u = \frac{\mathbf{x}}{\sqrt{x_1^2 + x_2^2}}, \text{ for } \mathbf{x} = \mathtt{F} \begin{bmatrix} \mathbf{l}_u \\ 1 \end{bmatrix}. \tag{4.20}$$

$$D(u_1,u_2,v_1,v_2) = \begin{cases} \lambda \big| \|\mathbf{l}_{u_1} - \mathbf{l}_{u_2}\| - \|\mathbf{r}_{v_1} - \mathbf{r}_{v_2}\| \big| & \text{, if } (u_1-u_2)(v_1-v_2) > 0. \\ \infty & \text{otherwise.} \end{cases} \tag{4.21}$$

$$\phi(u,v) = \min_k \left\{ \phi(u-1,k) + D(u-1,u,k,v) + C(u,v) \right\}. \tag{4.22}$$

The combination of transfer error (4.20) and monotonicity (4.21) make this method less prone to the problems of the naïve method. This minimization of this DP (4.22) can be performed along a unit-speed sampling of the left image to retrieve dense correspondences. The improved reconstruction results can be compared with the naïve approach in figure 4.11. The average reprojection error for this method can be seen in table 4.2. This optimization is performed at increasing resolutions of the right curve, to examine its effect on the reprojection error. First the right curve is sampled at nearly the pixel resolution resulting in a reprojection error of nearly half a pixel per point. Upon increasing the sampling resolution of the right label space to 0.1 pixels, the reprojection error falls to nearly 0.1 pixels. This shows that the best achievable reprojection error depends on the sampling resolution. However, the sampling resolution is limited by memory and time of optimization. Though the errors of DP and the naïve method are comparable, the lack of regularization allows the naïve method to achieve marginally better optima.

Stereo-aware WCM (sWCM): The DP based results still exhibit some jaggedness and noise as seen in figure 4.11. Simply smoothing the 3D coordinates is not an option as this will increase the reprojection error. The curves must be smooth while being faithful to the silhouettes. This brings about the need to vary the correspondences along the ribs until both objectives are met (as in WCMs 4.15). Additionally our smoothness (and tension) regularizers should not cause degenerate solutions as discussed in § 4.3. Taking all the desiderata into account, it is obvious that once more we must use the WCM (4.15) formulation to solve this time for a rigid object model from multiple calibrated views. The number of unknowns are much fewer than before and the optimization can be summed up as:

$$\min_{\underset{P\times 1}{\mathbf{t}_l},\underset{P\times 1}{\mathbf{t}_r},\underset{3\times p}{\mathtt{X}}} I_m = \min_{\underset{P\times 1}{\mathbf{t}_l},\underset{P\times 1}{\mathbf{t}_r},\underset{3\times p}{\mathtt{X}}} \left(E_{\text{smoothness}} + E_{\text{coincidence}} + \sum_p \mathbf{f}_{mp}^2 \right), \tag{4.23}$$

$$\begin{aligned}
\text{where}\, \mathbf{f}_{mp}^2 &= \left(\mathbf{l}(t_{lp}) - \pi\left(\mathtt{P}_{lm}\left[\begin{smallmatrix}\mathbf{X}_p\\1\end{smallmatrix}\right]\right)\right)^2 + \left(\mathbf{r}(t_{rp}) - \pi\left(\mathtt{P}_{rm}\left[\begin{smallmatrix}\mathbf{X}_p\\1\end{smallmatrix}\right]\right)\right)^2, \\
E_{\text{smoothness}} &= \lambda\|\mathbf{X}_{p-1} - 2\mathbf{X}_p + \mathbf{X}_{p+1}\|^2 + \psi\|\mathbf{X}_{p+1} - \mathbf{X}_{p-1}\|^2, \\
E_{\text{coincidence}} &= \chi_{\text{base}}\left(\|\mathbf{X}_1 - \mathbf{X}_{P/3+1}\|^2 + \|\mathbf{X}_1 - \mathbf{X}_{2P/3+1}\|^2\right) \\
&\quad + \chi_{\text{tip}}\left(\|\mathbf{X}_{P/3} - \mathbf{X}_P\|^2 + \|\mathbf{X}_{P/3} - \mathbf{X}_{2P/3}\|^2\right).
\end{aligned} \tag{4.24}$$

The correspondences are now expressed in terms of the parameters $\mathbf{t}_l, \mathbf{t}_r$ representing the positions of points along splines representing the stereo silhouettes. The best reconstruction for each image pair can be performed by optimizing E^{wcm} of (4.15) with $N = 1$, $K = 1$, variable correspondences $(\mathbf{t}_l, \mathbf{t}_r)$, fixed camera pair–$\mathtt{P}_{lm}, \mathtt{P}_{rm}$ and low weights on $E_{\text{coincident}}$ and E_{smooth}. The results are seen in figure 4.12. The reprojection error from the three methods can be compared in the table 4.2.

The naïve method minimizes reprojection error most effectively but result in faulty models. The DP based approach optimizes reprojection error almost equally well. The DP 3D models do not have the stray vertices of the naïve method. They are plausible although being slightly noisy and jagged. This is improved in the sWCM approach. This function firstly takes the more accurate reprojection error into account instead of the directional transfer error. Variable correspondences are incorporated for greater flexibility. The all encompassing objective(4.23) strikes a balance between optimizing reprojection error and accounting for image noise. Thus it produces the best 3D models qualitatively (figure 4.12), while compromising minimally on the reprojection error (see table 4.2).

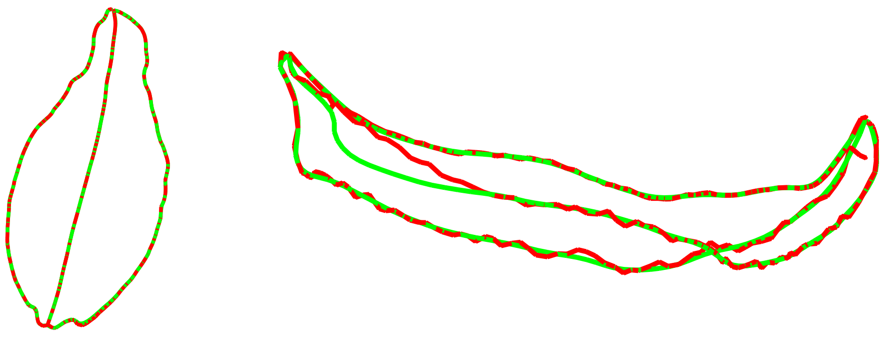


Figure 4.12: Final sWCM reconstruction: The best reconstruction using sWCM (in green) is compared against the DP-based solution (in red) of figure 4.11. The sWCM optimization is performed upon a comprehensive analytic objective function, inclusive of variable correspondences and regularization. The resulting reconstruction is much more smooth and plausible, despite minimizing reprojection error equally well.

4.6.2 Improving PDMs

In typical PDM based methods, the shape exemplars $\mathtt{X}_m$ are used to learn a compact basis representation. One common method is to use Principal Component Analysis to extract eigen-vectors (see Blanz and Vetter [15]). The first K eigen-vectors are used to represent the shape in its linear subspace. This method assumes that the shape distribution can be approximated with a Gaussian. Though variations have been proposed, the ease of optimization of the PCA-based method makes it most popular.

Method	Average Reprojection error per image vertex
Naïve transfer error	0.04
DP (label space at pixel resolution)	0.51
DP (label space at 10×pixel res)	0.09
sWCM	0.12

Table 4.2: Reprojection error across the different methods for stereo reconstruction are compared. Each method minimizes a slightly different objective. Minimizing naïve transfer error results in the best reprojection error but the worst 3D models (figure 4.11). DP based discrete optimization corrects some of the problems with almost equally good reprojection error. The sWCM (figure 4.12) yields the best combination of reprojection error while maintaining a good 3D model. The performance of the DP-based method is subject to the resolution of discretization of the label space and computational constraints.

Problem The PCA based method doesn't take individual image-based errors of projected vertices into account and assumes that each vertex has uniform isotropic noise in 3D space. We argue therefore, that in order to accurately compute the compact shape representation, the image errors should be accounted for. Instead of treating this as a modular problem of (i) first building shape exemplars and (ii) converting them to compact representations, we should be jointly working towards finding the best shape bases that explain all the stereo images well.

Data The stereo data allows us to perform a number of experiments. The first test compares conventional 3D PCA (i.e. building a 3D point distribution model [37]) and stereo-aware WCM (sWCM).

We can directly optimize the likelihood of the bases by running WCM with a reduced parametrization, where every pair of views is parametrized by a single set of rotation and translation parameters. These parameters are used to create a 4×4 similarity transformation $\mathtt{H}$, and the values of $\mathtt{P}_{2m-1}$ and $\mathtt{P}_{2m}$ are then given by

$$\mathtt{P}_{2m-1} = \mathtt{P}_L \mathtt{H}_m, \tag{4.25}$$

$$\mathtt{P}_{2m} = \mathtt{P}_R \mathtt{H}_m. \tag{4.26}$$

This helps to relate the coordinate frames of the individual petal instances. Running this optimization effectively measures the error of the bases in the image coordinate system, from whence it came, rather than in 3D. Optimization over all the parameters–camera, correspondences, bases and mixing coefficients, allows us to build a shape model while taking the stereo pairs that they came from into account.

Experiments and results We use the stereo reconstruction of § 4.6.1 as our base method for computing the compact, linear subspace defined by bases $\mathtt{B}$, using PCA with the first $K = 6$ eigen-vectors from $M = 20$ exemplars. Now using the bundle adjustment (Appendix B) to optimize upon the special case sWCM (4.23, 4.4) we optimize upon 14180 variables including the bases. The sWCM algorithm is initialized with the bases from PCA. The resulting new bases $\mathtt{B}^*$ show a reduced reprojection error compared to the original bases (see table 4.3). The trade-off made in terms of the 3D modelling error is negligible, and the sWCM successfully decreases the objective.

As seen in figure 4.13, the optimization yields a good estimate of shape (in the K = 6 subspace) to each training image pair. The original ground truth shape is seen in blue. For the 3D PCA approach, we took the estimated 3D shape for each stereo pair, registered them as a pre-processing step, then fit a 3D PCA model. From this model, we reconstructed the 3D shapes shown in green. Fitting the sWCM on the dataset yields the shape approximation seen in red.

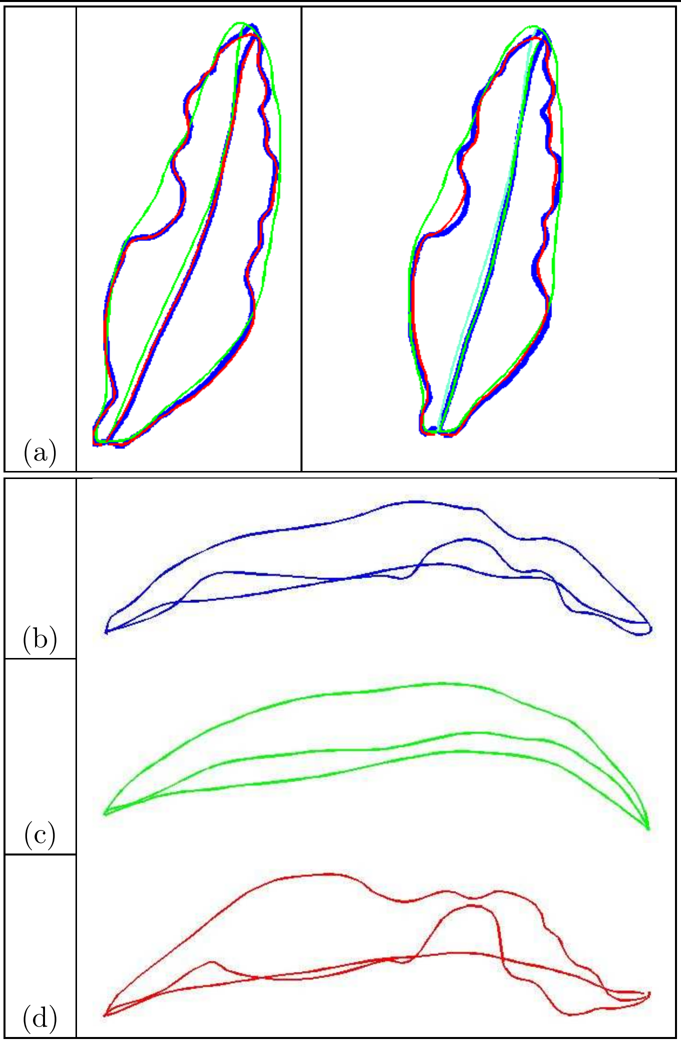


Figure 4.13: Training on the stereo dataset (Blue-original shape, Green-PCA fitted to original shape, Red-sWCM fitting (a) Shows the left and the right image of a stereo pair. (b) Best ground truth estimate (c) PCA bases fitted to the ground truth shape (d) The sWCM trained model fitted on this image pair.

K=6	Before		After	
	3D error	SRP error	3D error	SRP error
Initial	0.035	6.54	0.035	6.54
Final	0.073	3.83	0.06	1.25

Table 4.3: For the basis representation using $K = 6$, it can be seen that further bundle adjustment using (4.23) improves the reprojection error of the PCA bases acquired from stereo. While doing so, there is no loss of 3D modelling accuracy as seen in the table.

Algorithm 4.2 Summary: Stereo-aware wireframe class models.

```
1: For a stereo pair, take the left image and identify $P$ equidistant points on it. Using
   these as the nodes of a graph and the candidate correspondences as labels (discretized
   points on the right image rib silhouettes), run a dynamic program on a regularized
   objective of transfer error and curve consistency (described in § 4.6.1).
2: Use the correspondences to initialize the regularized objective of reprojection error
   with smoothness and petal-specific coincidence (see eqn. 4.23). The optimization is a
   bundle adjustment performed over correspondences and 3D structure.
3: For the purpose of learning a compact PDM representation: use PCA on the aligned
   stereo-built 3D models to extract bases (§ 4.6.2).
4: In order to learn an improved basis set: optimize over the regularized objective of re-
   projection error (eqn. 4.23). Now the optimization is performed over correspondences,
   bases, shape-fitting coefficients and inter-shape alignment (similarity matrix) parame-
   ters.
```

4.7 Summary

We introduce a novel extension to nonrigid SfM, and show that it constructs better deformable class models from a collection of photos of an object class. This is done in the absence of any initial shape information (Experiment I). We show this method extends to cases where stereo information is present (Experiment II). Preliminary results on improving PDMs show that they can be tested in a variety of environments where class-based models are used, such as single view reconstruction.

In Experiment I (algo. 4.1), we have shown how a single framework, built around regularized reprojection error with variable cameras, 3D models and correspondences. The framework can be expressed as one joint objective and solved in a principled manner with bundle adjustment. In contrast to existing state of the art, our images are not temporally smooth, nor do we have reliable point correspondences. We use the notion of having higher-level features such as object curve correspondences in images. We perform a variety of 3D reconstruction tasks on a collection of similar object class instances (as opposed to just deformed versions of the same object).

Unseen stereo examples provide us with a testbed consisting of ground truth. We show how conventional stereo reconstruction methods can be improved for wireframe based object classes (algo. 4.2). Conventionally PDMs are built in a modular framework–first build exemplars and then fit a PDM to these exemplars. In Experiment II, we also show how conventional PDMs can be improved by joining these steps and optimizing PDM bases over their source images. Effectively we show how to train the novel PDM framework with the sWCM approach. This must be further tested on testbeds to examine the improvement in performance in greater detail. This is the next natural step for this experiment.

Chapter 5

Class-based Object Detection and Segmentation

One of the goals of 3D modelling is to make the whole process as easy as possible for the end user. To this end, we would like to create a completely automatic end-to-end system. Given an instance of an object with information about its class and topology in an image, the following steps need to be automatically performed: (i) Detect the apparent contour curves—for example by segmenting out the object; (ii) identify the contours in the parameter space; (iii) generate a surface consistent with the apparent contours and texture map it. We have demonstrated steps (ii)-(iii) in Chapter 3, we now address step (i). To this end, we show how object class-based edge classification can be used to modify state of the art methods in object detection and segmentation to achieve our goal.

5.1 Introduction and overview

Edges are important features of an image. They are coupled to the geometry and reveal important information about the scene. They can be found reasonably reliably and are partially invariant to illumination or moderate changes in contrast *etc*. Importantly, they reduce the amount of image information that needs processing for regular image processing and vision tasks, by expressing an image concisely.

Object detection relies on the use of class features. These can vary in complexity from simple ones such as locally computed intensity and gradient-based features to more complex statistically computed ones such as SIFT *etc.* For practical purposes, features such as colour, texture, gradient features *etc.* are each treated as independent features. In algorithms such as [67], recognition is performed while treating all image edges equally regardless of their context. However, all edges are not equal. The edges on the boundary of an object from a specific class have the characteristic local colour or texture of that object class on one side and can have anything (*i.e.* class or non-class information) on the other side. Similarly, class specific edges may also have a characteristic shape. Thus these features–texture, edge information *etc.* are inextricably tied together and must be jointly analyzed. In this context, our objective is to learn an edge classifier to differentiate between edges lying on the boundary of an instance of an object class, from others. For such classification it is necessary to make use of other local information surrounding the edge–appearance, texture and shape (see Figure 5.1). These edges will have the standard gradient properties but in addition the classification differentiates between edges that lie on the boundary of the object, and those that don't. Therefore, foreground *vs.* background image appearance can be differentiated in the context of an image edge gradient. This is studied at a local level thus making this process fast and easy. Such bottom-up information can be combined efficiently with a top-down imposition of structure and constraints. While conventional cue integration tends to occur later in the processing pathway, this "early vision" integration means that it is easy to modify existing applications to use our class-specific edges, offering the potential for improved performance across a range of applications.

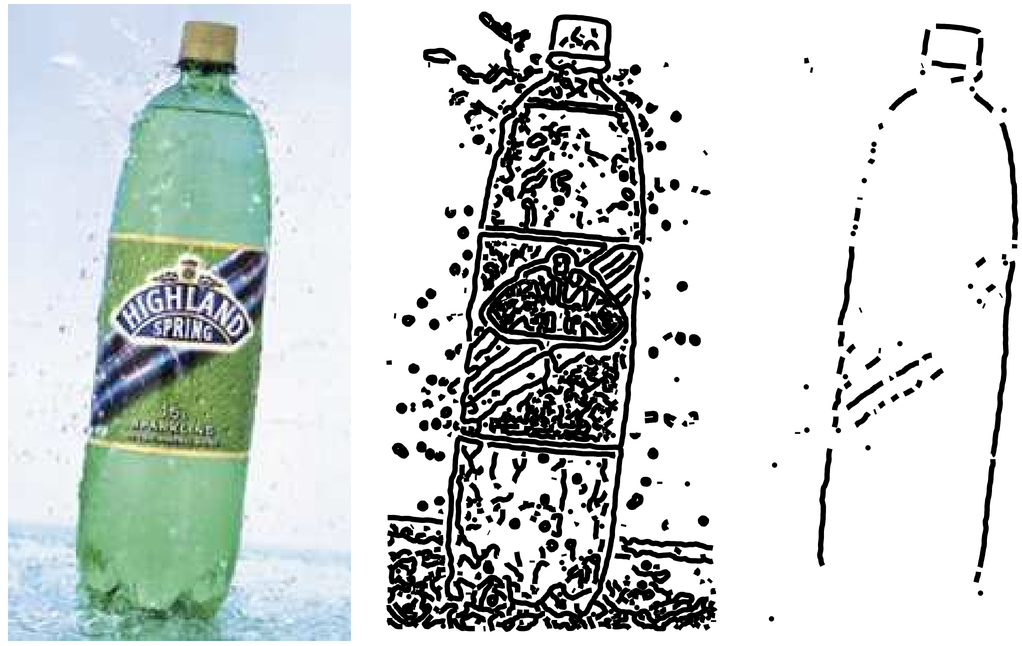
Object segmentation is another much researched topic in computer vision that can benefit from class-based edges. Recent approaches rely heavily on many forms and modifications of the mincut-max-flow algorithm over an image treated as a Markov Random Field (MRF). In particular we modify and employ the ObjCut formulation of Kumar *et al.* [103]. In this method, segmentation is treated as a binary MRF labelling problem. The MRF includes likelihoods from the foreground and background colour (or texture) distributions, and the segmentation is guided by a pictorial structure object model (ObjCut adds the pictorial structure prior to earlier work such as GrabCut [163]). We will add our edge classification to this model to improve performance.

In this chapter we describe our method for edge classification in §5.2 (an example of this can be seen in figure 5.1). We show how this can be used to improve object detection based on the chamfer matching method (Gavrila [67]) in §5.5. Subsequently this is used to improve the ObjCut algorithm of [103] to yield more accurate object segmentation in §5.6. Finally, class-based single view reconstruction can be automated using this as shown in §5.7.

5.2 Classifying edges for an object class

We now describe how local information neighbouring an edge can be learnt to classify detected edges into those arising from the boundaries of an object class or not. Our objective is to separate class-specific boundary edges from others—image edges arising from internal discontinuities, specularities, and background clutter. For each edge, simple features that encapsulate the neighbourhood are extracted. These features are then used to learn a classifier to predict whether the edge lies on the object boundary.

There is the question of how to represent the appearance *e.g.* colour distribution, texture and shape of the edges. Traditionally this has been handled by extracting easily discriminable complex features by convolving the regions of interest with a wide range of filters. However the contrasting approach is to avoid this and let the classifier discern between the unprocessed samples. We follow the latter, by simply extracting a rotationally invariant (up to a flip factor) patch around each edge point using a feature vector



(a) A simple image (b) Canny edges (c) Validated edges

Figure 5.1: Overview. The background and internal gradients in (b) make object detection difficult. (c) The class-specific validated edges help in removing clutter and internal gradients. Most of the non-class edges have been suppressed, greatly simplifying subsequent processing such as detection or class specific segmentation.

consisting of the pixel colours. This representation implicitly captures the shape from the edge boundary running through the patch, and the colour and texture of the patch. On the other hand, with such a simple representation we are not explicitly recording that the distributions on each side of the object boundary edge chain may be different. The idea is that the ability to discriminate can be gained either during the stage of extracting features or while classifying them. Varma and Zisserman [200] demonstrated this by classifying texture in monochrome images with the use of simple patch features.

Feature extraction Given an image, a set of edges are extracted using the Canny edge detector. This algorithm is inherently robust for the job because of its use of non-maximal suppression and hysteresis [27, 28]. The parameters in Canny are: (i) the initial amount of smoothing before edge detection, and (ii) the thresholds for qualifying high contrast regions as 'edges'. The Canny algorithm is available in many implementations and the parameter values depend on the implementation and the image it is applied to. The actual parameters are not important as long as the following characteristics are ensured. Smoothing can help reduce image noise but detail should not be lost and the edges should not be offset from their true location. Similarly, the thresholds must be set to allow enough object detail to come through, while admitting minimal clutter. In practice the parameters are never perfect and for an image, some clutter will always come through, while some relevant detail may be lost due to insufficient contrast in some regions.

To simplify ground truth annotation, edges are linked into chains automatically according to their spatial proximity as shown in figure 5.1(b). Now edges are associated with these edge chains instead of individual sub-pixel level detections. As a result, stray single pixel detections can be discarded easily enabling usage of edge chains that form more meaningful demarcations in the image. All edge chains are manually annotated for training and verification. They are marked positive if they lie on the boundary of an object instance; all other edge chains are negative by default (see figure 5.2). This is a simple procedure as only a few user clicks are required per image for annotation of positive examples. This can be read in detail in Appendix A.

We use simple $m \times n$ (where m, n have been appropriately chosen) image patches as our features. Each point along the edge chain is the center of a patch feature. For standardization, each patch is centered at its corresponding point on the edge chain and rotated so that its x-axis is aligned with the tangent at that point along the edge chain. An example can be seen in figure 5.2. This involves a simple interpolation to extract the rotated patches from the image pixels. The colour values of each pixel in this patch are now used to represent the appearance ([200] only used grey values).

Flip ambiguity Our patches extracted from the images are rotationally invariant up to a flip factor. For *e.g.* in a positive patch, the object region can lie either on the top or bottom

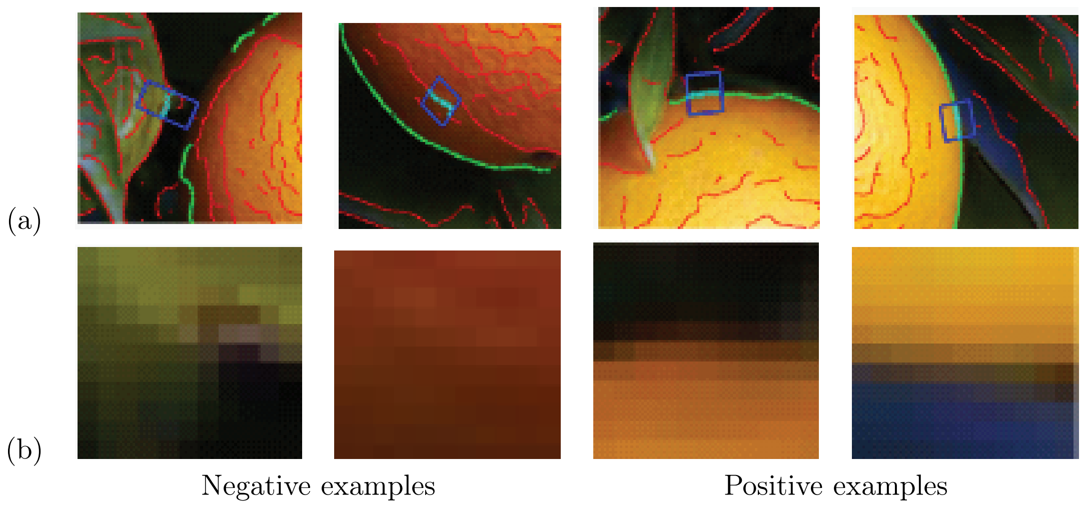


Figure 5.2: Annotation and feature patches: Annotated edge chains can be seen in cropped images in the row (a) Positive edge chains are marked in green, and the negative ones in red. The patches extracted as each edge chain is traversed are shown in row (b). Patch in (b) corresponds to boxed location in (a).

half of the patch (figure 5.2). This means that the domain of valid patches forms multiple modes in the patch space. Ideally, the training data should capture sufficient examples with the object on either side and the learning should efficiently handle this flip ambiguity. However, sufficient examples with object on either half of the patch cannot be guaranteed in the process of data extraction. Also the classification algorithm may not be equipped to deal with multi-modal domains. The problem of choosing the right model and parameters for the classification can add to the difficulty. We resort to a simple trick for removing this ambiguity. In the presence of a dominant class characteristic such as colour or texture, the patches can be flipped to remove this ambiguity. Consider the class–orange (fruit); a patch from an image of an orange, can be flipped so that the more "orange" region always lies on one half-patch (say upper side). The "orangeness" of a half-patch can be measured by taking the joint probability of each pixel in that half-patch according to a colour or texture distribution for that class. This can be understood as a representation where all data lies in one half of the patch space, therefore eliminating the multi-modality caused by the flip factor. Classes like *bottles* lack such a dominant colour or texture characteristic.

The approach for such a class is described in § 5.3.

Classification is basically the task of assigning a label (from a space of two or more labels) to a variable, or a set of variables and can be performed by a variety of methods. We use the Support Vector Machine (SVM) [172] to learn an edge classifier (object boundary *vs.* others) for the patch features. The parameters we need to learn are: the size of the patch; and for the SVM: the kernel type (we compare linear, RBF and polynomial kernels of degree $1, 2$ and 3) and parameters such as the cost factor (weight of errors on negative samples *vs.* those of positive ones). The performance of the classifier on a test set is based on measures such as accuracy, precision and recall. These are defined as follows:

$$\left.\begin{array}{ll} \text{Accuracy}= & \left(\frac{tp+tn}{tp+fp+fn+tn}\right), \\ \text{Precision}= & \left(\frac{tp}{tp+fp}\right), \\ \text{Recall} = & \left(\frac{tp}{tp+fn}\right). \end{array}\right| \begin{array}{l} \text{tp} = \text{True Positive}, \\ \text{tn} = \text{True Negative}, \\ \text{fp} = \text{False positive}, \\ \text{fn} = \text{False negative}. \end{array} \tag{5.1}$$

The output of the SVM is a real valued function which is thresholded to get binary classification.

5.3 Edge classification: experiment and results

We illustrate our method on three classes here: oranges, bananas and bottles. We follow the standard procedure of assembling a set of images which are used to first train and then evaluate the classifier. We assemble a database for each class of about 100 images (examples can be seen in figures 5.3-5.4). Each dataset is split into half for training and testing (the training set is split further for training and validation). The images cover a wide range of scale, pose, perspective distortion and illumination conditions, and include multiple object instances, partial occlusions, and background clutter. Other state of the art, such as Lempitsky *et al.* [111], undertake the challenge of finding object pose for object detection via a globally optimal branch-and-bound algorithm. However, they consider only a limited range of pose variation thus making their method unviable. Across the large variations exhibited by these datasets (figure 5.3,5.4) their method would prove extremely expensive. Examples are shown in figures (5.1,5.6-5.8).

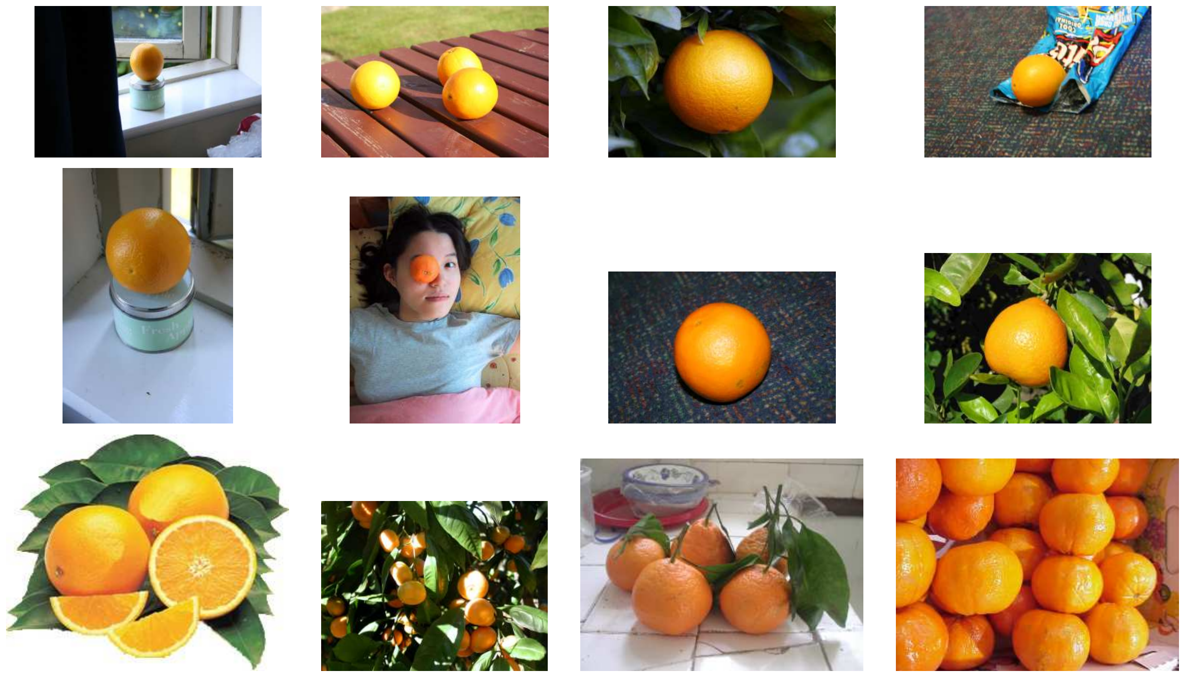
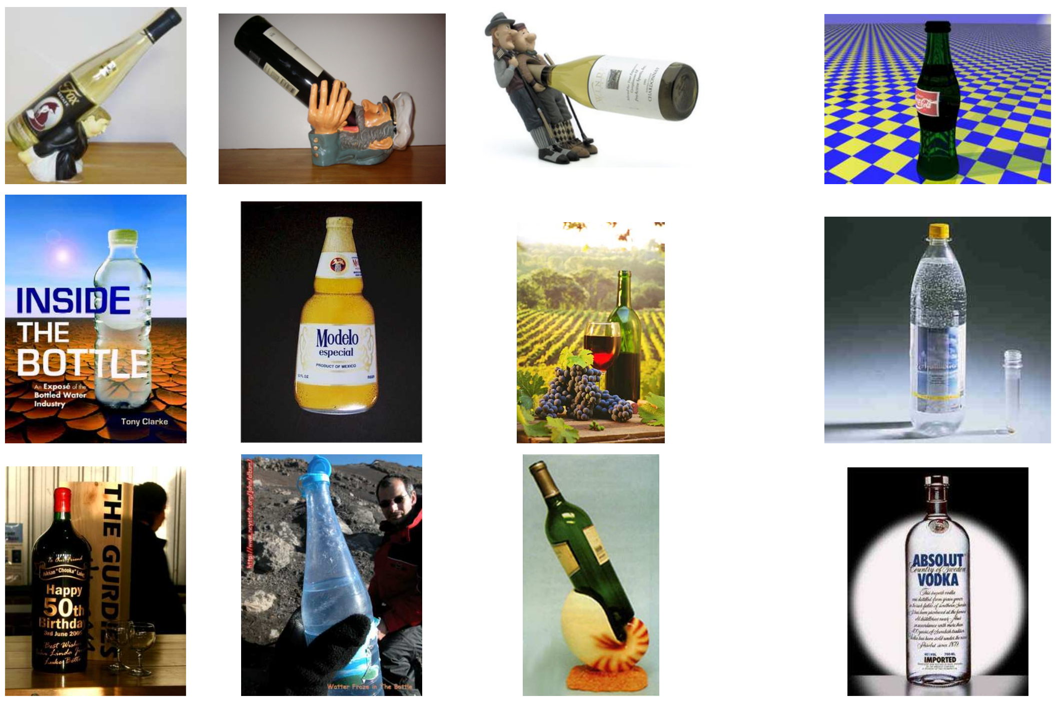
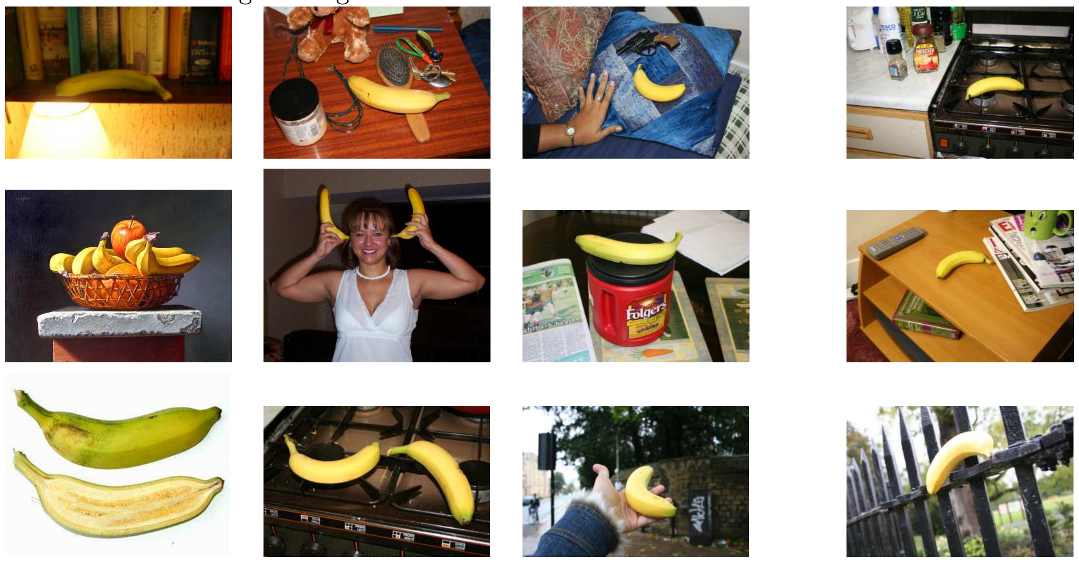


Figure 5.3: Some pictures from the Orange dataset. Note: the specularities, shading and texture on the object surfaces. Some occlusion and clutter can be observed. Contrary to expectation, the shape is not always a perfect circle. Other circular objects in the images (tin box, face, circular specularities, shadows *etc.*) can confuse standard object detectors.

We use the SVM for edge classification and learn the model via cross validation. We find that the best performance for the orange and banana datasets is achieved with a polynomial kernel of degree 2, and with patches of size $m = 11$ (along the y axis of patch coordinate frame) and $n = 11$ (along the x direction of the patch coordinate frame). The ratio of the number of available positive and negative examples is used to relatively weight the training errors on negative and positive samples (Morik *et al.* [134]). For the bottle dataset, the RBF kernel shows superior performance, the size of the patches being the same.



The bottles vary widely in shape, colour, labels and are often occluded. They range from being opaque to translucent or transparent. Images downloaded from the web may also have compression artifacts contributing to image noise.



The colour varies between yellow and green, with or without black stains. There is often some occlusion and lots of clutter. The dataset is collected over wide changes in illumination and pose parameters (scale, rotation, foreshortening *etc.*)

Figure 5.4: Some examples from the bottle and banana dataset.

The SVM output f_i at location $\mathbf{i}$ in the image is usually thresholded to retrieve binary classification c_i. Alternatively it can also be interpreted as a probability into a probability $isedge(i)$ that location i is an object edge. Logistic regression is performed on the f_i as shown:

$$isedge(i) = p\left(c_i = 1 | f_i\right) = \frac{1}{1 + \exp(Af_i + B)}. \tag{5.2}$$

A and B are parameters that must be learnt by minimizing the negative log likelihood of the training data *i.e.* the SVM outputs and thresholded classifications f_i, c_i:

$$\{A, B\} = \underset{A,B}{\operatorname{argmin}} \sum_i \frac{(1 + c_i)}{2} \log(p_i) + \frac{(1 - c_i)}{2} \log(1 - p_i). \tag{5.3}$$

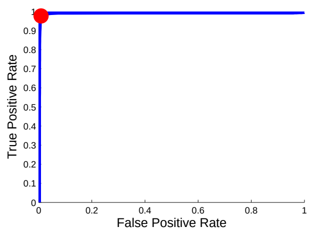
This can be read up in further detail in work by Platt [149] and more recently by Grandvalet [77].

Flip ambiguity: For the orange and banana classes, Gaussian mixture models (GMMs, see [129, 130]) over pixel colours are built from training data and these are used to flip the training and test features. For comparison, the best performance of SVMs for the flipped *vs.* unflipped cases is compared in the tables in figure 5.5. Flipped features perform better and are used for the orange and banana classes. The colour and texture of the 'bottle' class is much more variable and hence Gaussian mixture models for colour will not be helpful in disambiguation. For such categories the classifier has to handle the ambiguity by choosing the appropriate support vectors and slack variables.

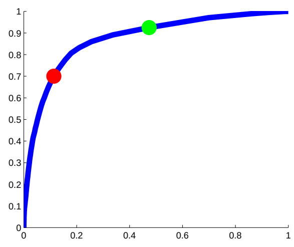
The performance of the classifier is summarized in the table of figure 5.5 for the enumerated classes. Receiver operator characteristic curves (ROC curves: see [156] and [201] for a more modern approach) summarize the performance of the classifier as the number of occurrences of true positives (sensitivity) is traded against those of true negatives (specificity). This can be done by varying the operating point along the ROC curve. Figure 5.5

shows the result of varying the extent of suppression of false edges with the variation of the operating point. A high area under the ROC curve is desirable and indicates good learning across a range of sensitivities. The models are fairly well learnt as can be seen from the ROC curves in the top row of figure 5.5. In the case of bottles, the lack of one distinctive colour or texture reduces our accuracy. For such object classes, (lacking distinctive colour or texture) other representations may be necessary. For example, texton distributions or separate local colour histograms for each side of the boundary. The classification results on several example images can be verified in figures (5.6-5.8). The choice of the operating point is important and usually this is based on the relative importance assigned to misclassification of positive *vs.* negative features. We assign equal weight to misclassification of positive and negative samples and this cost is represented by a line ($y = mx + c$), where slope: $m = 1$. The operating point is chosen to be the point of tangency on the ROC curve with this line such that the offset c is maximized. This operating point is also the closest of such points to point $(0, 1)$ on the axes.

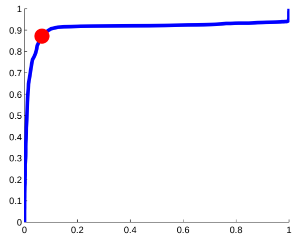
The method performs well on all datasets. Since the feature used is a simple vector of neighbourhood pixel colour values, this method works best for classes with characteristic texture, such as bananas and apples. Bananas and oranges show good classification. There are hardly any false negatives. Some false positives are detected on orange like regions and even orange reflections (see figure 5.6). The bottle class is more difficult and has more false classifications. The labels on bottles are particularly difficult as they have widely varying colour and texture as seen in figure 5.8.



(a) Orange

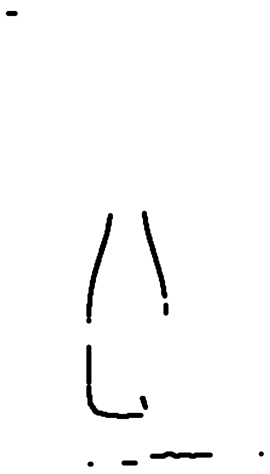


(b) Bottle

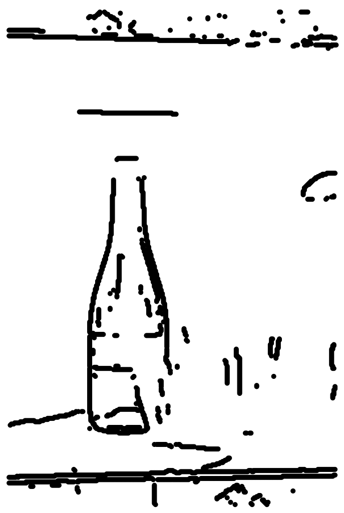


(c) Banana

(e) Image



(f)Lower false positive rate



(g)Higher false positive rate

Best classification without flipping			
Class	Accuracy (%)	Precision (%)	Recall (%)
Orange	98.59	71.46	67.11
Banana	96.37	82.03	58.18
Bottle	82.01	90.03	72.00
Best classification with flipping			
Orange	98.48	99.39	97.57
Banana	90.37	92.79	87.53

Figure 5.5: Edge Classification Results. The ROC curve plots the True Positive Rate against the False Positive Rate as the threshold is varied for classification between the minimum and maximum values of the SVM output. (f),(g) show edge classification with a variation in the operating point for the bottle image of (e). In (b) the operating point towards the left results in lower false positives as seen in (f) and a change to the green operating point on the right results in a higher false positive rate (g). The red points on (a),(b) and (c) show the operating point used for the datasets. The classification results at these operating points are given in the table.

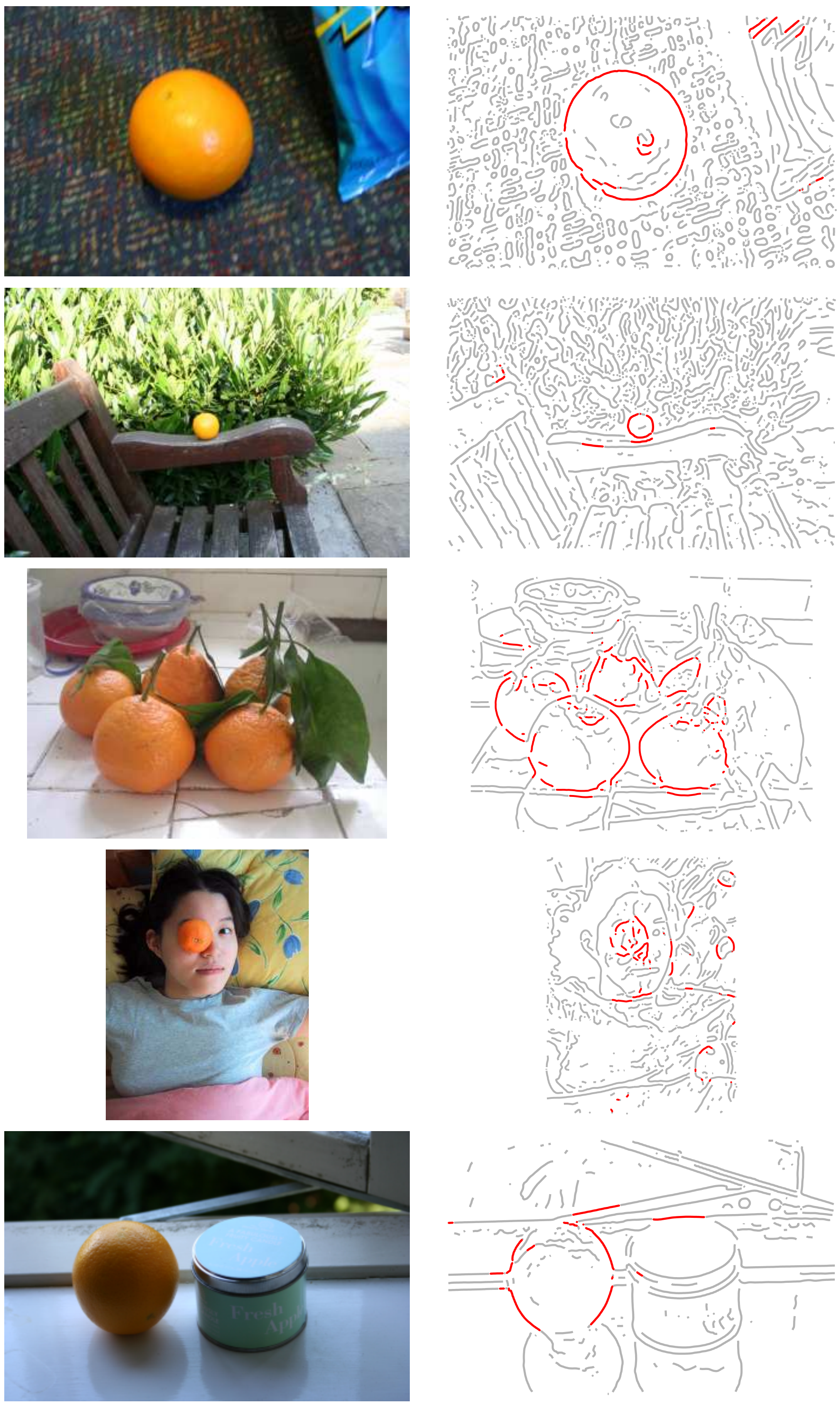


Figure 5.6: Example images and class-based edge classifications for ‘orange’: Edges are red if classified as +ve object boundary else they are plotted in gray. A large number of edges from clutter, specularity and internal gradients that confuse template matching are discarded by this classification. Most orange boundaries are correctly classified, with the exception of orange-like regions. The last row shows the misclassification of orange reflections on the sill and the tin box.

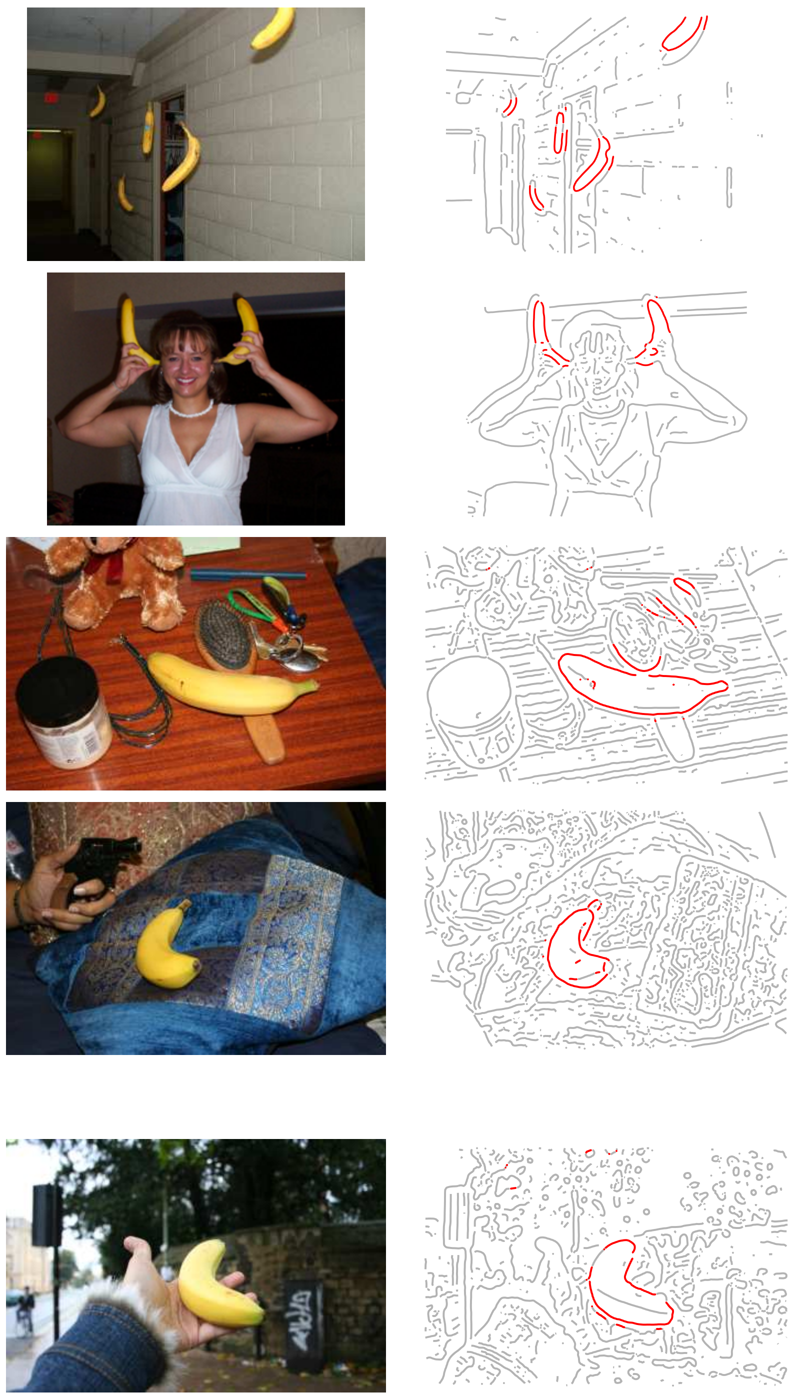


Figure 5.7: Bananas: Edge classification on some example banana images. Edge labels are labelled red if classified as +ve object boundary else they are plotted in gray. A large number of edges from clutter, specularity and internal gradients that confuse template matching are discarded by this classification. A few false positives can be seen from confusion banana-coloured key-chains in row 3.

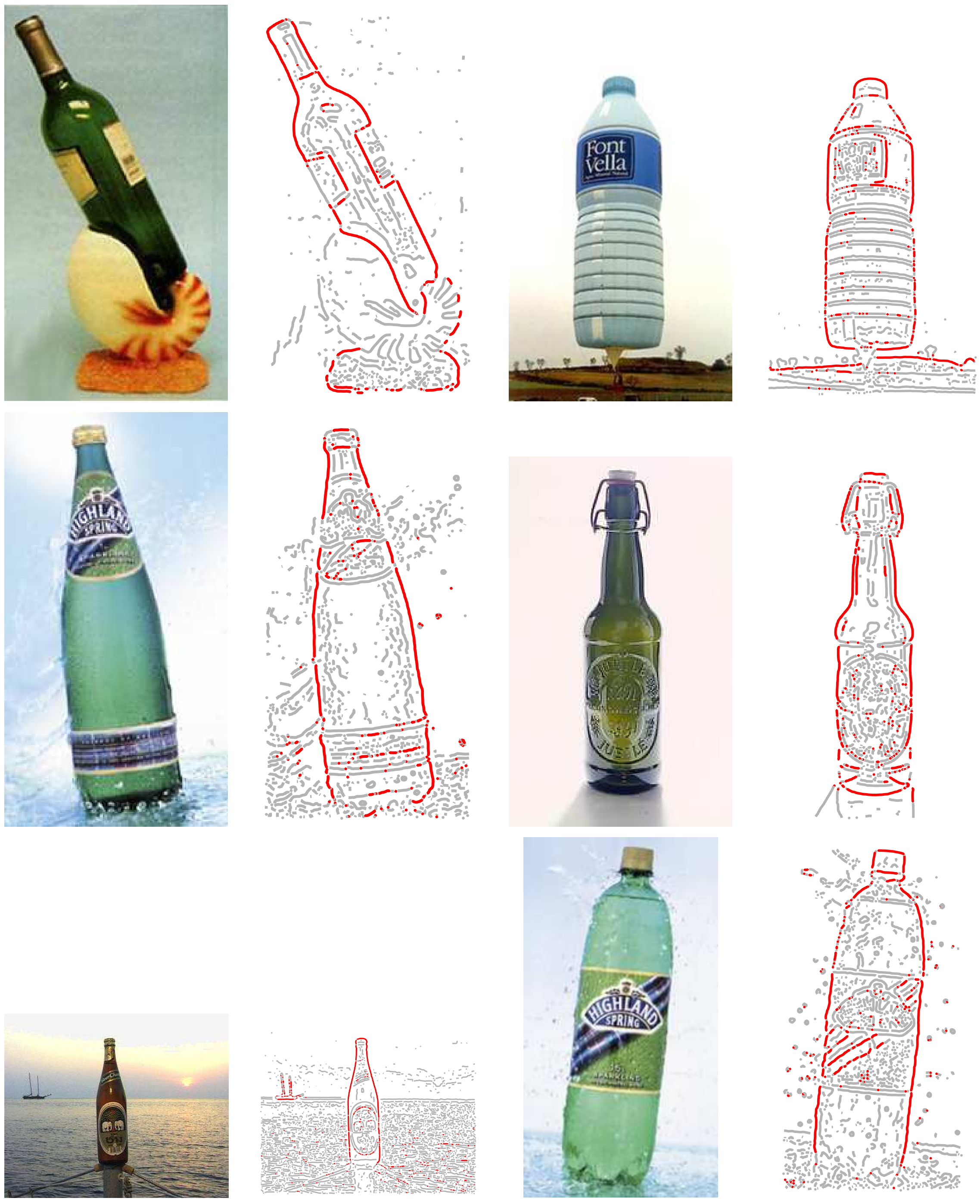


Figure 5.8: Edge classification on some example bottle images. Edge labels are labelled red if classified as object boundary else they are plotted in gray. While significant clutter is eliminated, the lack of characteristic texture causes more misclassification. Bottle labels are especially confusing due to their variety of text and colour.

5.4 Discussion

We have explained the process of class-based edge classification and shown its performance on 3 datasets. Before discussing its applications, there are many interesting questions that are worth further discussion. The flip ambiguity is one important point, for which we find one working solution. Admittedly, there may be more principled ways of handling this problem. Some of these are as follows:

1. Ideally, the ambiguity should be automatically handled by the classification process. This seems to yield less optimal results as shown in figure 5.5.

2. Alternatively, we can use the existing and flipped versions of all training features to learn the classifier. This would mean that more regions of the parameter space are considered valid and the decision boundaries may become more complex. Then the onus of learning this ambiguity would lie with the classifier. Since (1) does not work well, this might not do much better.

3. The training annotation can include information about the inside and outside of the object, ensuring correct orientation and eliminating the need for the colour heuristic. However, the ambiguity would still persist during the testing phase.

4. The flip factor could be treated as a latent variable, that could be marginalized out.

5. Alternatively, we could also try to handle this ambiguity at the kernel level. One such experiment was to modify the RBF kernels to internally flip the patches and choose the one which best optimizes the cost. For example, using the kernel:

$$k(x, x') = \max\left(\exp\left(-\gamma\|x - x'\|^2\right), \exp\left(-\gamma\|x - \text{flipud}(x')\|^2\right)\right),$$

   where flipud(x) flips the patch vertically, with the intuition that the correct alignment of the patches (original or flipped) will have lower cost due to greater consistency of the object region. Note that this kernel is no longer a Mercer kernel. However, upon experimentation, we found that this kernel (and similar modifications of linear kernels) performed worse than the standard polynomial and RBF kernels. This difference was heightened if the category has a strong colour model. The inferior performance could simply be a result of not finding the right kernel and SVM parameters. Therefore, the failure of our experiment is not the final word on experimentation with more elegant kernels.

There are also interesting questions about the best methods for edge finding and feature representation and the merits of global *vs.* local methods. Though we use the SVM for classification, other methods such as Adaboost [64], Random forests [25] *etc.* may also be employed. We now discuss these two applications of edge-based classification.

5.5 Application I: Chamfer matching for object detection

Local information in the form of colour, texture and shape is used to classify edges from a class-based perspective. This learning can be used in applications for object detection and object segmentation. Class-based edge validation can help disambiguate between multiple hypotheses and benefit object detection methods such as chamfer matching greatly. In chamfer matching a set of learnt object templates are matched to the detected edges in the image using a distance transform (DT). The position at which the convolution of the template $\mathbf{T}$ with the distance transform of the feature image (the edge-map $\mathbf{I}$) is minimal, determines the match. For scale invariance this can be stated as:

$$D_{chamfer}(\mathbf{T}, \mathbf{I}, \theta) = \frac{1}{|\mathbf{T}|} \sum_{t \in \mathbf{T}} \min_{i \in \mathbf{I}} d(i, t, \theta), \quad |\mathbf{T}| \text{ is the length of template.} \tag{5.4}$$

Here $|\mathbf{T}|$ denotes the length of the template, while θ denotes the set of pose parameters, such as translation on the image, that are applied to the template before evaluating this measure. For each point on the template, the closest point on the image edge map is found, making this measure asymmetric. The symmetric measure would have been defined as follows:

$$D_{chamfer}(\mathbf{T},\mathbf{I},\theta) = \frac{1}{|\mathbf{T}|}\sum_{t\in\mathbf{T}}\min_{i\in\mathbf{I}} d(i,t,\theta) + \frac{1}{|\mathbf{I}|}\sum_{i\in\mathbf{I}}\min_{t\in\mathbf{T}} d(i,t,\theta). \tag{5.5}$$

Consider $\mathbf{T},\mathbf{I}$ to be the edge-maps for two templates being compared. The asymmetric chamfer distance is prone to some degenerate solutions, *e.g.* $\mathbf{T}$ can shrink to a point and match $\mathbf{I}$ perfectly. The symmetric chamfer is a much more robust way of measuring their similarities. However, the second part of this distance must be recomputed every time the template and pose change even the slightest bit, making this very expensive. The asymmetric chamfer distance can be computed efficiently and quickly by the use of Distance Transforms (DTs) and is therefore the preferred choice for object detection here.

Variations of the basic algorithm have been introduced for robustness. The DT can be used to find a precomputed approximation of Euclidean distance using integer arithmetic and simple interpolation. The DT may be truncated at a certain threshold to accommodate occlusion. Also, f^{th} quantile values (such as in the Hausdorff distance) can be used for robustness in handling occlusion. Symmetric chamfer distances may be used in order to reduce mismatches, but they involve more expensive computation. The orientation at edges could be accounted for by using oriented DTs, *i.e.* a different DT is stored for each of a set of discretized gradient directions. This method is effective only when smooth sub-pixel contours are available to compute the DTs for each of the orientation bins.

Matching At a match the chamfer distance must have a low value. Therefore, potential match is found by by minimizing (5.5). Instead of the optimal match, all matches lying within a certain threshold are considered potential matches by Gavrila [67] and subjected to a second verification stage. We will simply consider the single most optimal match (equivalent to having strict thresholds at each prototype node) instead and want this to be accurate without the need of subsequent hypothesis verification. Therefore, the goal is to find $\mathbf{T}, \theta$ such that the DT is minimized for a given image $\mathbf{I}$. Ideally this match should coincide with the actual location of the object of interest.

Usually, the exact specific template corresponding to an image is unknown. Therefore the shape of the object must be generalized from training data. We want to encapsulate the variation across a variety of pose parameters across the possible template space. In practice (see [67]), this is done by simply taking the finite set of training exemplar templates and subjecting them to the set of possible geometric transformations such as rotation, scaling *etc.* to build a vast artificial set of templates. These 'pose' parameters are sampled at discrete intervals to create this data. This is more general than the initial set and computationally easier to handle than the vast continuum of possible shapes. Searching through this vast space of templates which would have otherwise been expensive, is tackled by building a template hierarchy based on structural similarity. This is expensive but can be computed offline. Prototype templates represent the nodes of the tree and distances between templates are represented in order to capture the similarity between prototypes and their child templates (example in figure 5.9). During test time matching can be optimized, by only considering the prototype templates which may yield potential matches thus pruning the search across the tree.

Building the hierarchy involves measuring similarity between each pair of templates. A similarity matrix can be computed from inter-template distances. However in the presence of a large number of templates this can be very expensive. Gavrila [67] proposes a K-means like algorithm which can be faster but less accurate than more comprehensive methods such as agglomerative clustering.

Given a test image, a hierarchical branch-and-bound strategy is used find template matches for the given test image. With the occasional modification, this is a standard algorithm for object detection. In this work, we concern ourselves only with the 'one' best match per image.

The performance of the described hierarchical scheme is rife with some problems:

1. Prototype templates can latch on to various noisy features from occlusion, specularities and random clutter, leading to a small value of $D_{chamfer}$ even when the actual object has not been found (seen in figures 5.9, 5.10, 5.11).

2. The discretization along the parameter space exacerbates this problem. A different template at a different pose can give a smaller truncated chamfer cost by latching on clutter. In contrast, the template that is closest in size, shape and pose to the given image instance may have a higher chamfer cost due to the discretization at which the template dataset is generated.

3. The clustering via the Kmeans like algorithm (see [67, 68]) is neither deterministic, nor globally optimal and also contributes to this problem. This may be improved by choosing a different clustering approach such as agglomerative clustering. However, the untruncated unidirectional chamfer distance is not a metric measure, and choosing the right thresholds at each node, the number of nodes in each level, the number of levels, and the truncation thresholds for chamfer matching, is difficult. Strict thresholds can cause solutions to be missed. On the other hand, lenient thresholds will cause the search to proceed through a large portion of the tree making it expensive.

Originally, Gavrila [67] follows the detection stage with verification where each hypothesis is tested and verified. By using the validated edges from §5.2 for chamfer matching, the process can be sped up without the need for subsequent hypothesis verification (see figure 5.11).

Class-based chamfer matching results: Hierarchical trees are built for each of our datasets *i.e.* orange (50 images), banana (60 images) and bottle (90 images). A few independently segmented masks (training masks) are taken to learn hierarchical template trees. Templates are created from these training masks by subjecting them to the following transformations:

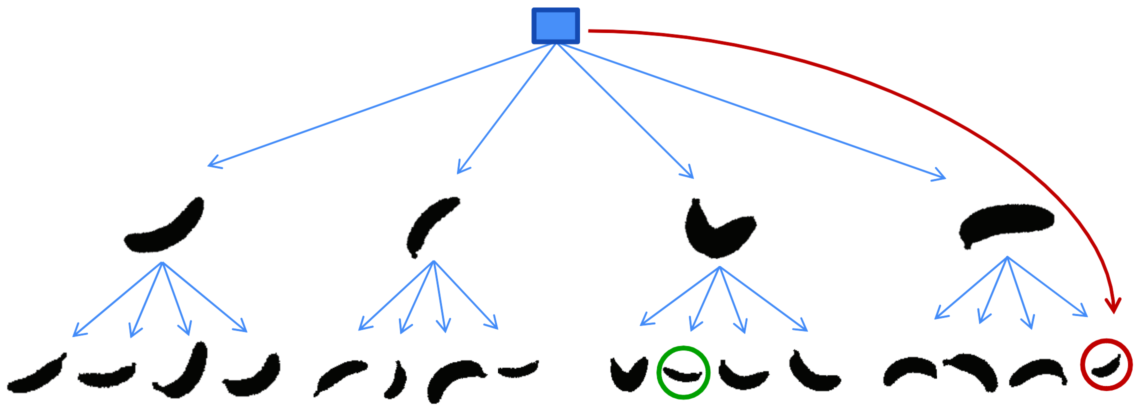


Figure 5.9: A hierarchical template tree showing some branches of the tree of templates and prototypes. The templates are generated at discretely generated samples in the pose space. Therefore for a given image, the template corresponding to its exact pose parameters might not exist in the tree. Perspective distortion which is only approximated by the pose parameters, can also make matching difficult. Therefore, the correct (green) match may be abandoned in favour of an incorrect (red) match. This can also happen if the prototype templates match against clutter in the image edge maps can leading to ambiguity between multiple possible matches.

1. orange: 4 training masks each taken across all combinations of 7 rotations and 11 scales

2. banana: 26 masks at 16 rotations and 16 scales

3. bottles: 7 masks at 10 rotations and 11 scales

The discretization for each dataset is chosen on the basis of the complexity of the class and samples in the training data. The orange is a simple shape and despite variation in bottle shapes, they are observed at relatively simple orientations (mostly upright). Therefore for the orange and bottle classes fewer masks and rotations angles are required. In contrast, the banana is observed at a wide range of orientations and scales and needs a larger set of templates. We first use the original hierarchical template matching approach of Gavrila [67] to find the one best match per image. As shown in figures 5.10(b) and 5.11(a) there are numerous mismatches. We then use the validated edges from our edge classifier (figures 5.6-5.8) to determine the relevant edges for the current object of interest (see 5.10(c)). Chamfer

	Oranges	Bottles	Bananas
Raw edge map	27/50	26/90	14/60
Validated edge map	43/50	41/90	29/60

Table 5.1: Visually correct detections are summarized before and after the edge maps are validated. The validated edge maps clearly give more correct detections. (Also see figures 5.10,5.11)

matching upon this leads to improved detection (figures 5.10(d) and 5.11(c)) compared to the original (figures, 5.10(b) and 5.11(a)). This can be used to improve any algorithm that uses template matching, such as subsequent segmentation using ObjCut as discussed next and illustrated in figure 5.11(d). For this experiment, the entire dataset acts as the test set and the number of visually correct matches are summarized below:

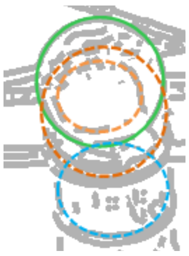
5.6 Application II: Class based segmentation—ObjCut

In this section we illustrate how edge specific classification can be used to improve the performance of an object segmentation algorithm. In particular we modify the ObjCut algorithm of Kumar *et al.* [103]. ObjCut is a Bayesian method for class based binary segmentation using an Object Category Specific Markov Random Field (MRF). In [103], a pictorial structure formulation is used in order to handle multiple part objects. In this work, we deal with one part objects therefore eliminating the need for the use of pictorial structures.

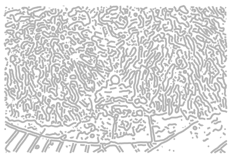
To segment a given image, a two stage approach is followed in [103]: (1) hierarchical chamfer matching is used to first localize the object in the image, and (2) the object characteristics from the detection are used to perform mincut based segmentation. We have shown how to use edge classification to make (1) better in § 5.5. Additionally, we now show that the edge classification results can be used to modify the weights of the image-based graph for refined mincut segmentation in stage (2).

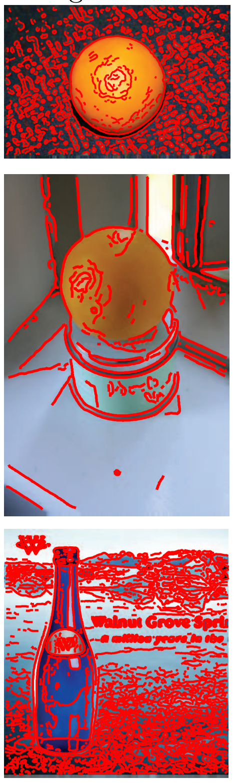
ObjCut formulates the segmentation problem as an object category specific condi-

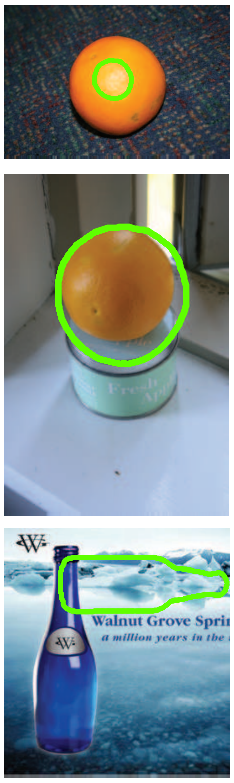
(a) An apparently simple image

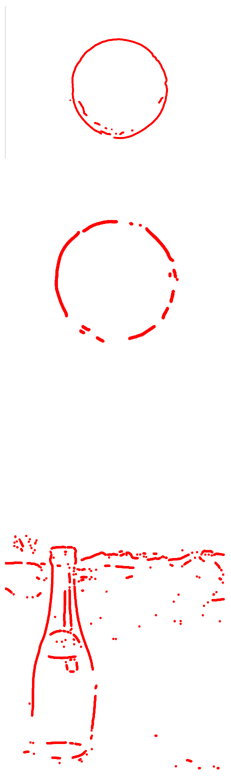

(b) Many plausible matches

(c) A simple but cluttered scene


(d) chaotic edge map (in gray)


(e) Original image with edge map (in red)


(f) Best match


(g) Validated edge map (in red)

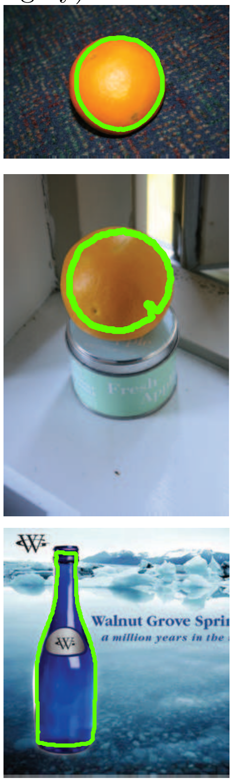
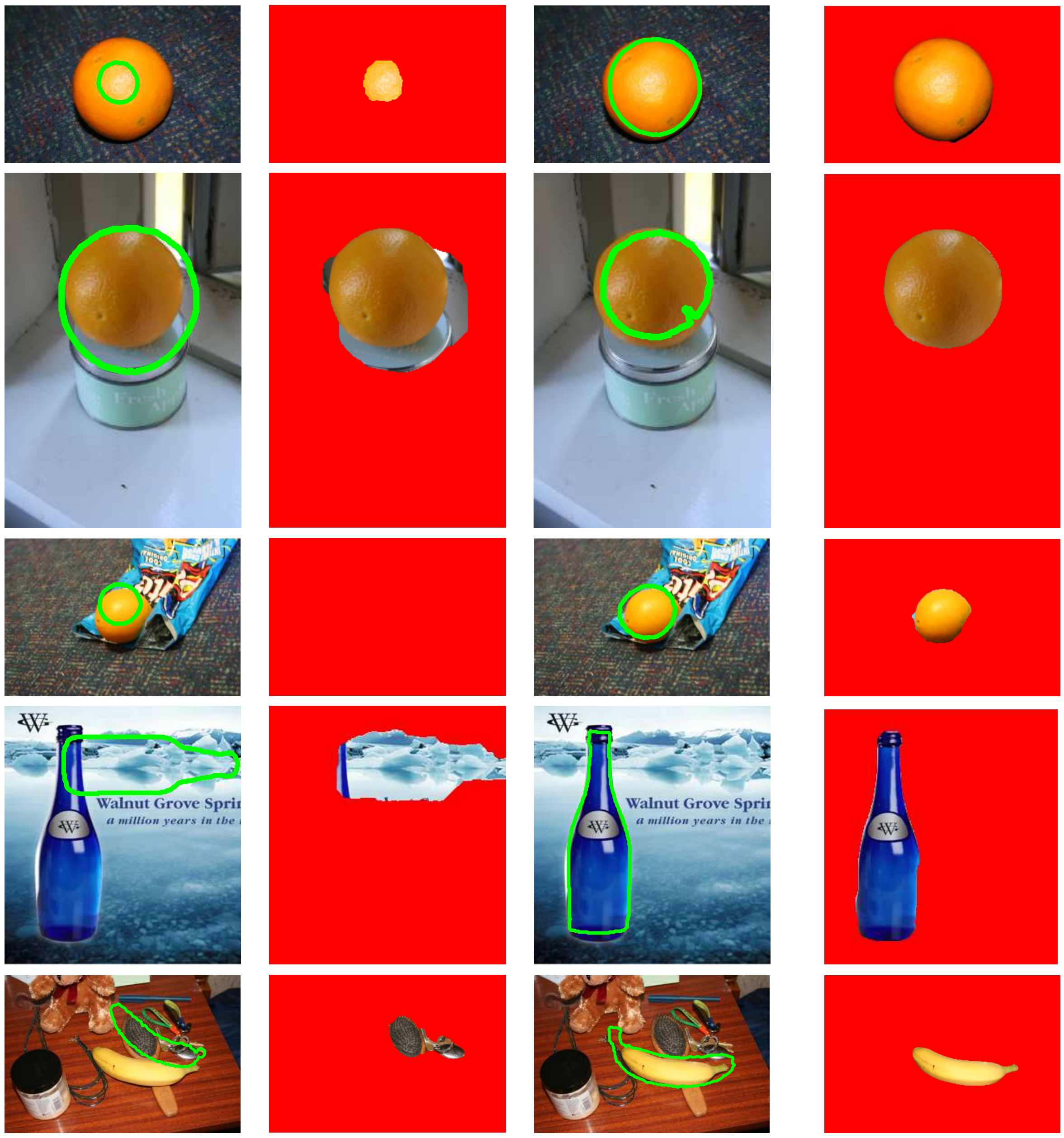

(h) Best match on the validated edge map

Figure 5.10: Edge classification improving Chamfer matching based detection. The first row demonstrates the problem encountered by chamfer matching. (b,d) shows the (gray) edge maps of two apparently simple images (a,b). The correct green match in (b) may be missed due to the presence of other plausible matches (in the other colours), which are latched on to internal specularities and parts of object clutter (tin box). Similarly, one can see how practically any template could be matched in several locations of the edge map shown in (d). In subsequent rows, (e) shows the base image and its edge map. The results of hierarchical chamfer matching are faulty as seen in (f). Replacing the edge map with the validated classification results shown in (g) dramatically improves the 'best chamfer match' as seen in (h).



(a) Basic Chamfer (b) ObjCut (b) Chamfer with class-specific edges (c) Improved ObjCut

Figure 5.11: Class-based chamfer detection for improving ObjCut segmentation. **(a)**Chamfer Matching using all image edges. The matches latch on to irrelevant edges corresponding to internal gradients and specularities (first row), and clutter (circular lid, second and third row). Results in bad segmentation **(b)**. Matching on *class* edges and texture **(c)** leads to better matching – compare with the confusions arising from the problems in **(a)**.**(d)** Modified ObjCut results are much more accurate.

tional random field (CRF). The joint energy for any configuration of the image-based graph is given by the following equation and can be optimized by mincut:

$$\chi(\mathbf{m}, \mathbf{d}; \boldsymbol{\Theta}, \boldsymbol{\tau}) = \sum_i \left(\begin{array}{c} \underbrace{\psi(\mathbf{d}_i; m_i, \boldsymbol{\tau})}_{\text{pixel likelihood}} + \underbrace{\phi(m_i; \boldsymbol{\Theta})}_{\text{shape likelihood}} \\ \\ + \sum_{j \in \mathcal{N}_i} \left(\underbrace{\psi(m_i, m_j; \boldsymbol{\tau})}_{\text{Ising prior}} + \underbrace{\phi(\mathbf{d}_i, \mathbf{d}_j, i, j, m_i, m_j; \boldsymbol{\tau})}_{\text{contrast term}} \right) \end{array} \right). \quad (5.6)$$

The terms in this energy are now defined. The image has N pixels which form a set of observations $\{d_i | i = 1 \dots N\}$. The goal is to find a binary labelling $m_i \in \{0, 1\}$ dividing the image into foreground and background. A set of parameters $\boldsymbol{\tau} = \{H_0, H_1, P, \lambda, \sigma, \mu\}$ is either learnt or set heuristically. The parameters act as knobs to control the different terms of (5.6).

Note: we use $\psi(x)$ to define the negative log likelihood (or energy) for any probability $p(x)$.

$\mathcal{N}_i$ denotes the set of pixels in the neighbourhood of i. The first term in (5.6) is the likelihood term for the observation at a pixel given its labelling. This emission model is given by:

$$p(\mathbf{d}_i | m_i, \boldsymbol{\tau}) = H_{m_i}(\mathbf{d}_i), \quad (5.7)$$

$$\psi(\mathbf{d}_i; m_i, \boldsymbol{\tau}) = -\log(p(\mathbf{d}_i | m_i, \boldsymbol{\tau})), \quad (5.8)$$

where H_1 and H_0 are the foreground and background (normalized) colour (RGB) distributions respectively.

The second term $\phi(m_i | \boldsymbol{\Theta})$ denotes a likelihood term for the shape. MRF-based segmentation techniques which use MINCUT perform very well with manual initialization in the form of strokes, bounding boxes *etc.* Automatic recognition and detection of the object using a shape model can replace the user interventions to yield accurate automatic segmentation. Having learnt a shape model, a set of latent shape variables $\mathbf{\Theta}$ can be used to fit the model to a given image, which will favour segmentations of a detected shape. Here, we use a template tree as described in §5.5 and $\mathbf{\Theta}$ denotes the discretized pose parameters (scale, rotation, translation) involved in the process of hierarchical chamfer matching. The function $\phi(m_i|\mathbf{\Theta})$is chosen so that if we were given an estimate of the location and shape of the object, then pixels falling near that shape would more likely have object (foreground) label and vice versa. It has the form:

$$\phi(m_i;\mathbf{\Theta}) = -\log p(m_i|\mathbf{\Theta}), \tag{5.9}$$

$$\text{where we define: } p(m_i|\mathbf{\Theta}) \propto \frac{1}{1+\exp((-1)^{m_i}\mu * d(i,\mathbf{\Theta}))}. \tag{5.10}$$

$d(i,\mathbf{\Theta})$ is the distance of a pixel i from the shape defined by $\mathbf{\Theta}$ (being negative if inside the shape). The parameter μ determines how much the points outside the shape are penalized compared to the points inside the shape.

The third term is an Ising prior which encourages pixels in a neighbourhood to take similar labels, defined as:

$$p(m_i, m_j|\boldsymbol{\tau}) \propto \exp\left(-(1-\delta_{(m_i,m_j)})P\right), \tag{5.11}$$

$$\psi(m_i, m_j;\boldsymbol{\tau}) = -\log(p(m_i, m_j|\boldsymbol{\tau})). \tag{5.12}$$

In addition, a fourth (contrast) term is commonly used in CRF based segmentation methods. This favours pixels with similar colour having the same label. This is done by reducing the cost within the Ising model for two labels being different in proportion to the difference in intensities of their corresponding pixels. This contrast term of the energy function is given by:

$$\phi(\mathbf{d}_i,\mathbf{d}_j,i,j,m_i,m_j;\boldsymbol{\tau}) = -\lambda\frac{(1-\delta_{m_i,m_j})}{dist(i,j)}\left(\frac{-\|\mathbf{d}_i-\mathbf{d}_j\|^2}{2\sigma^2}\right), \tag{5.13}$$

where $dist(i, j)$ is the Euclidean distance between the two pixels denoted by i and j. While σ controls the sensitivity of the penalty to the distance between the pixels (i, j), λ controls the contribution of (5.13) to the total energy function. Note that (5.13) is not a part of the prior, for the prior term cannot include the data.

5.6.1 Improving segmentation

Initialization

The edges and texture features in the image are used to localize the object in the image by performing hierarchical chamfer matching (as discussed in §5.5). This yields the object's pose parameters $\mathbf{\Theta}$ in the image. We modify this by performing chamfer matching over class-specific edge maps as discussed in § 5.5. This helps to prevent the mismatches illustrated in figure 5.9. The results of incorporating this are summarized in table 5.2 and figures 5.11, 5.10. This change makes a dramatic improvement to the segmentation quality.

Segmentation

Following [19] the CRF used in OBJCUT has a contrast dependent prior. This means that a segmentation which introduces a change of state between pixels (i.e. a change from foreground to background in this case) adds a cost to the energy, *but* this cost is diminished if there is a strong gradient between the pixels (as measured in the image data). The inclusion of data-dependent pairwise terms for pixels in a clique gives a substantial improvement in segmentation quality, and the resulting MRF can still be minimized using graph cuts as described in [19].

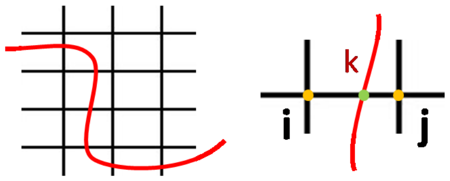


Figure 5.12: Boundary term: Edge classification on Canny edge chains at sub-pixel resolution, is used to set edge weights of the boundary term in the CRF. k denotes the point of intersection of Canny chain with CRF edge between nodes i and j. $edge(i, j) = isedge(\mathbf{k})$

We want the algorithm to be even more specific and consider only those edges that are relevant to the object class. Therefore, only those edges of the MRF which coincide with object boundaries should be weakened. Our edge classification (§5.2) can be interpreted as a probability $isedge(\mathbf{k})$ of a particular sub-pixel image point $\mathbf{k}$ lying on the object boundary as seen in (5.2). This is available along Canny edge chains that are derived at sub-pixel accuracy on an image. For the purpose of CRF segmentation, we need to convert this class-based knowledge to weights on an image graph with the pixels acting as nodes. The edge weight $edge(i, j)$ of the CRF between pixels i, j is represented by the value of the class probability $isedge(\mathbf{k})$ (defined by (5.2), figure 5.12) at the intersection of the Canny edge chain with the edge i, j. A new boundary term is defined, which adds to the category specificity of the CRF:

$$edge(i,j) = \begin{cases} 0 & \text{if no intersection,} \\ isedge(k) & \text{if valid intersection.} \end{cases} \tag{5.14}$$

$$\zeta(m_i, m_j, i, j; \boldsymbol{\tau}) = -\xi(1 - \delta_{m_i,m_j})edge(i,j), \tag{5.15}$$

where ξ is the parameter controlling the influence of this boundary term. Adding (5.14) to (5.6) gives us our new Object Category Specific CRF. This can still be optimized by MINCUT. The results of this change are seen in table 5.2 and figure 5.13.

5.6.2 Implementation

For ObjCut with the modified category specific MRF and shape model, the optimal parameters must be found. A large number of parameters (around 20) can be identified for the model to be learnt. Of these, 6 control the relative weights between the colour likelihoods, shape likelihoods, prior and the boundary terms (Ising, contrast and our class-based boundary terms), and are most crucial for performance and strongly interrelated. Coordinate descent is performed to maximize the accuracy of segmentation over the training data, for this subset of important parameters. Subsequently, the other parameters can be

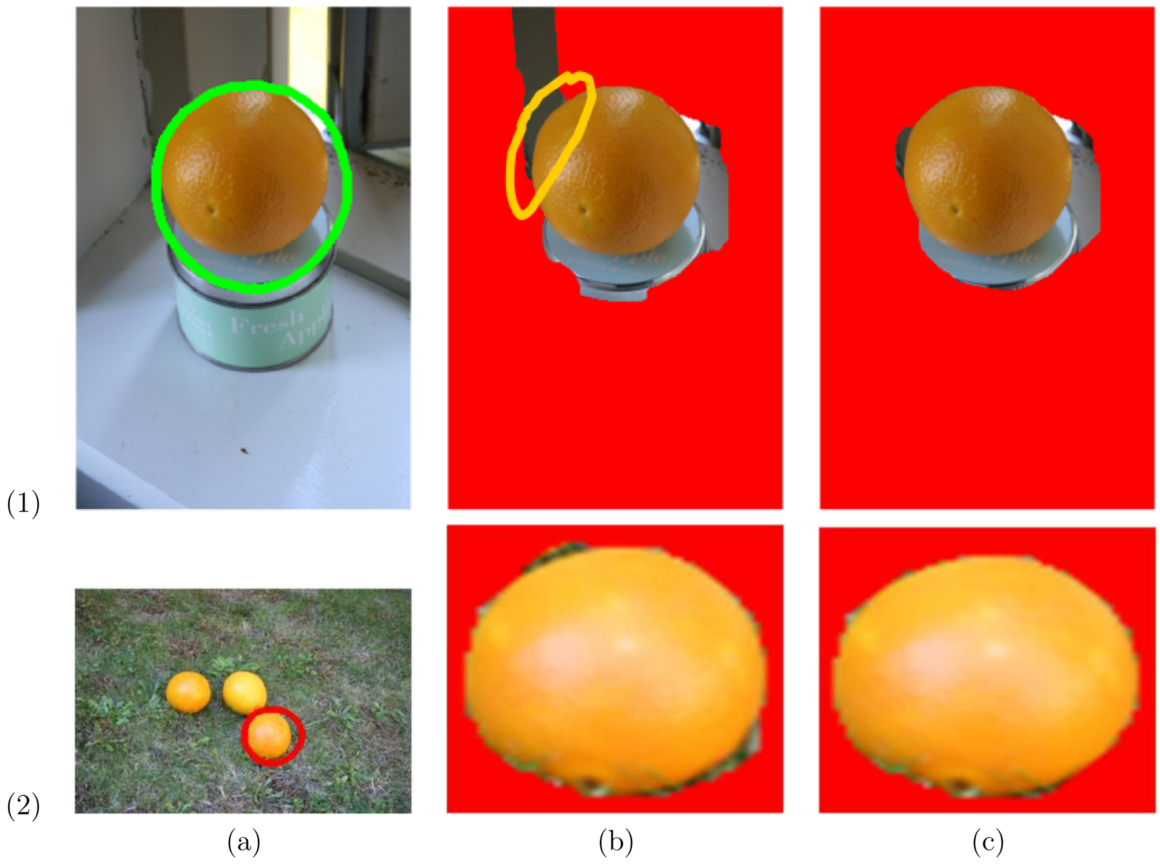


Figure 5.13: Edge classification based boundary term for improving ObjCut based segmentation. Two cases with partially incorrect chamfer matches are seen in (a). The incorrect detection leads to incorrect foreground and background models and faulty segmentation. (1) The boundary term from edge classification leads to a graceful recovery from such a misguided initialization and the leakage in this segmentation can be contained. (2) shows a similar case. The initial segmentation is inaccurate owing to incorrect initialization. But using the boundary term leads to a much more polished segmentation as seen in (2,c).

individually optimized in a similar manner. We start with large step sizes for gradient descent and reduce them as we refine our estimates. This is performed over a grid of sampled parameters and the time taken depends on the size of the search space. However, during testing, performing the modified ObjCut over each image takes only a few seconds.

5.6.3 Results

This test is performed upon the orange and bottle datasets contain 50 (23 training, 27 testing) and 90 (40 training, 50 testing) images respectively. The hierarchical template trees are the same as discussed in the results of §5.5. The parameters of optimization are minimized over the training set.

The performance is measured by the number of misclassified pixels in the test data with respect to the manually segmented ground truth. Table 5.2 summarizes the results for the two object classes. At first look, the improvement to chamfer matching from the edge classification seems to be making the maximum impact in terms of numbers. The subtle yet important contribution of the edge-based boundary term can be visualized in figure 5.13. Note: Each image has $90,000$ pixels on an average. For the orange class, we

Object class	ObjCut	ObjCut + modified CRF	ObjCut + modified CRF + chamfer matching
Orange	2947	2457	256
Bottle	8121	8064	4077

Table 5.2: Classification error: Average number of misclassified pixels per test image are shown. Better object detection through edge-validated chamfer matching makes the maximum reduction to error. However, the modified CRF makes the subtle but important contribution as shown in figure 5.13.

get visually correct segmentations for 47 out of 50 images. For the bottle class, we get 57 correct segmentations out of 90 images. The banana dataset is our most challenging dataset, owing to the wide shape variations and image clutter. We get good segmentations of around 37 out of 60 images. While both our edge based modifications improve ObjCut, the use of relevant edges in chamfer matching makes the more significant difference (see figures 5.11 (c) and 5.14).

5.7 Automation

We have shown how to automatically detect and segment instances of a particular class in a given image. We can use these segmentations to derive object silhouettes and subsequently perform reconstructions for relatively simple views of objects as shown in figure 5.15. In these examples the topology is known due to the knowledge of the class. Given the segmentation, it is also trivial to derive the silhouette for the object. In the case of simple frontal views of objects it is relatively easy to decide on one scheme of inflation. In this case, we assume cylindrical inflation (see §3.7.1.1) for the orange and banana classes. For an orange this kind of inflation automatically produces spherical shapes assuming the correct topology is imposed (see figure 3.10). For simple views, we can continue with the assignment of parameter curves to the silhouette-parameter space mapping as specified in §3.6. Figure 5.15 shows examples of completely automated reconstructions. Of course, in the case of an orange, there are *only* simple views.

5.8 Summary

In this chapter, we show how local image based information can be used to learn a classifier that differentiates object class boundary edges from other image edges such as clutter, specularities *etc.* We then show how class specific edges can be used for Chamfer matching and ObjCut for enhanced object detection and segmentation (algo. 5.1). The implications of such class specific edge labelling are many fold since any algorithm for object classes that uses edges, can now be improved. We show the use of this for automatic class specific single view reconstruction of objects.

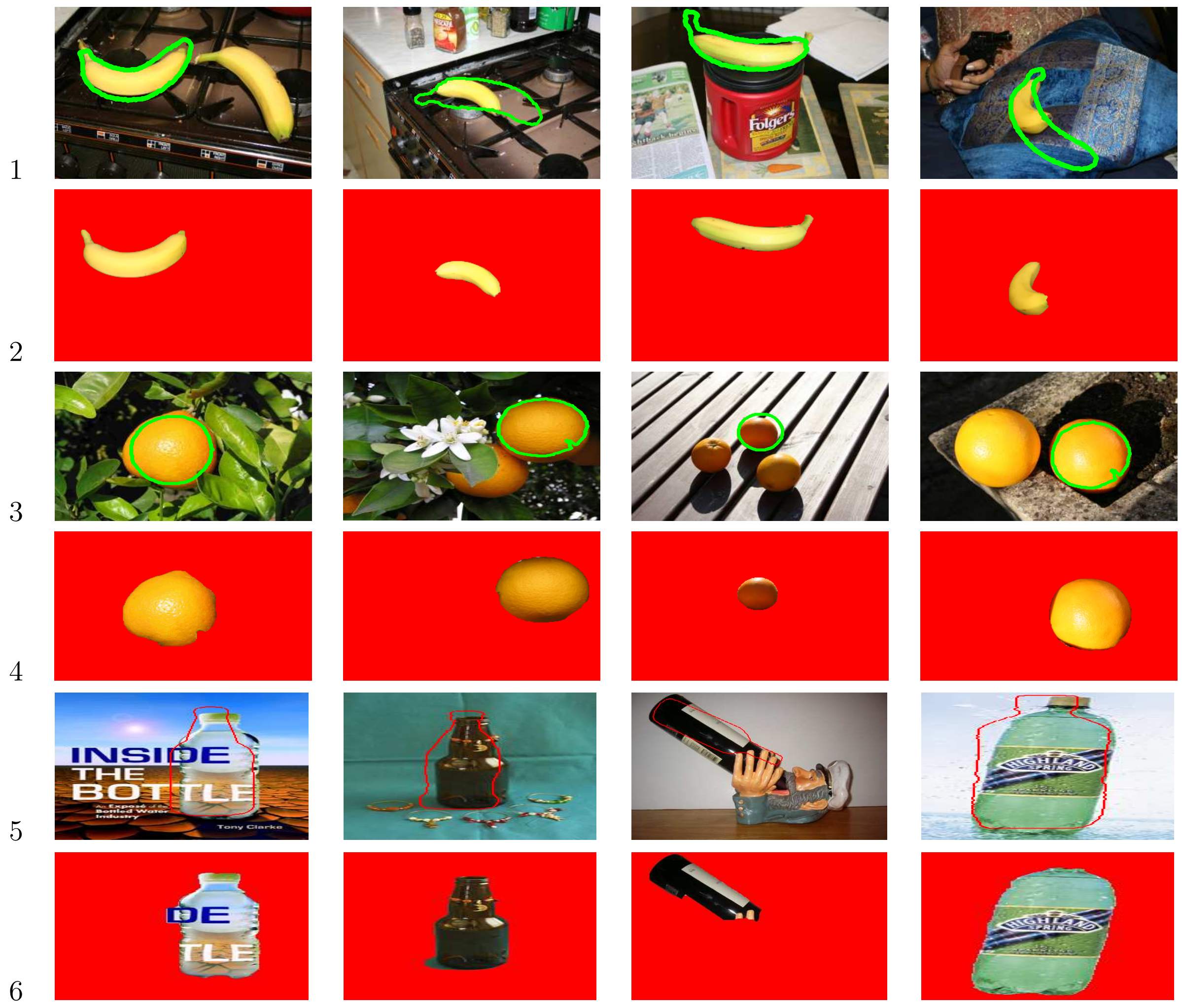


Figure 5.14: More results. The performance of our method on some examples is shown on the banana, orange and bottle datasets. This dataset, has a wide range of challenges from pose, scale, clutter and lighting. In the presence of multiple instances, we use the best Chamfer match as shown in Row 1. The segmentation using the initialization from Row 1, 3 and 5, by the improved ObjCut is shown on Row 2, 4and 6.

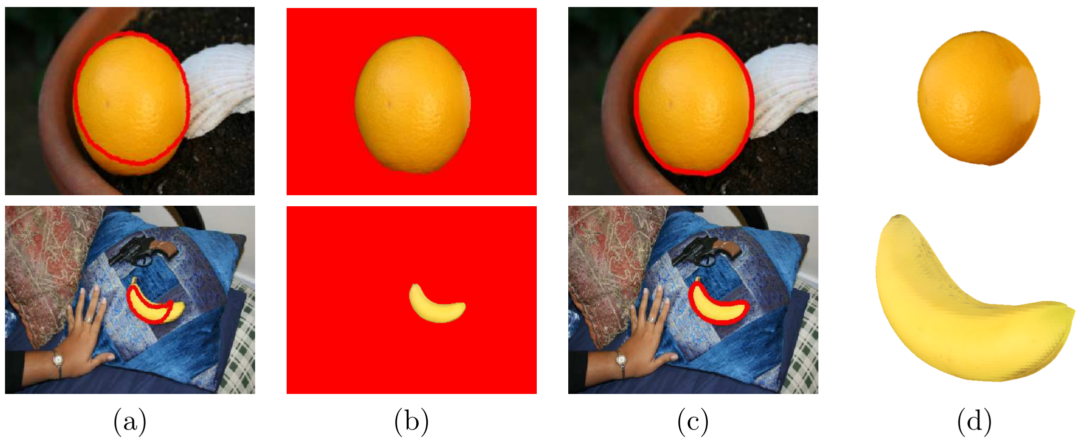


Figure 5.15: Automatic reconstruction of simple objects. (a) Two images superimposed with the initial detections found from the orange and banana models. (b) Segmentation obtained by our method. (c) Contours around the segmentation which form the silhouette input to the reconstruction algorithm. (d) Final 3D reconstruction.

Algorithm 5.1 Summary: Object detection and segmentation

1: Extract Canny edges from the dataset and link these to form edge chains. Annotate training images and extract patch features around the edge chains. Train the edge based classifier as described in § 5.2 and optimize over the parameters by cross-validation. Perform edge classification on the test images.

2: To perform object detection: learn a hierarchical template tree for the object class of interest. Perform object detection over the classified edge map as described in § 5.5.

3: To perform segmentation: Use the classified edge-map to perform the class-specific object detection and also modify the CRF weights. These contribute to the stages of ObjCut. Learn the parameters of ObjCut over the training set by cross-validation using a grid search (see § 5.6).

Chapter 6

Conclusion

We now conclude with a summary of the contributions of this thesis. We will also review recent developments and sketch some avenues for future research.

6.1 Single view reconstruction of curved surfaces

In Chapter 3, we show how readily available image information in the form of object silhouettes can be used for effective reconstruction of objects.

6.1.1 Contributions

- The silhouette is a projection of the 3D contour generator (CG) from the object's surface. The projective properties of the CG are exploited to provide constraints on the 3D surface. By using the freedom in surface parametrization, image-based constraints can be imposed on the actual unknowns (the 3D surface vertices).
- The problem of retrieving the unknown 3D model is framed as a convex problem which can be solved by one single linear operation.
- Within this linear framework, various cues such as normals, gradients and inflation constraints can be embedded.

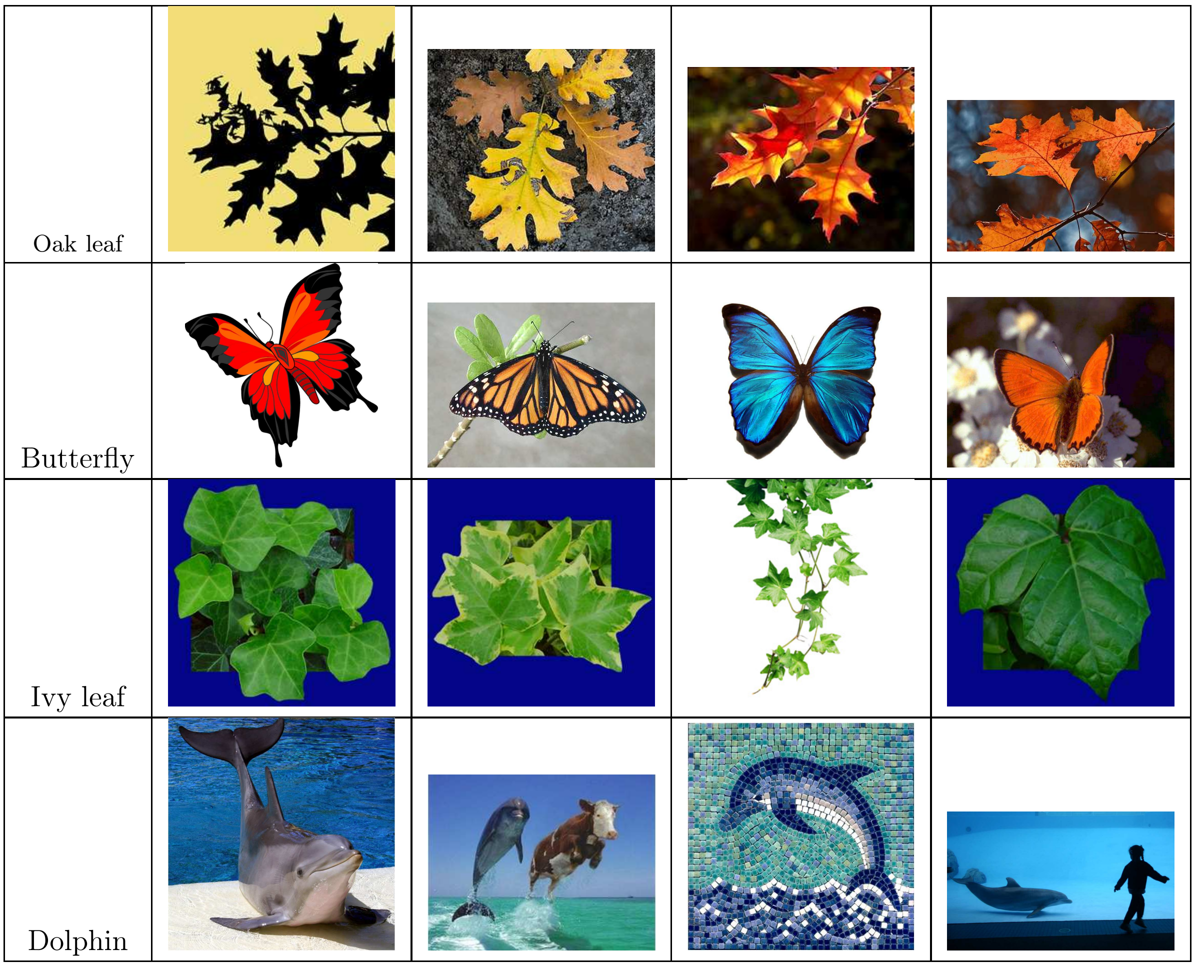


Figure 6.1: More deformable classes to explore: oak leaves, butterflies, ivy. While some features such as number of veins (5 in English ivy), symmetry (of a butterfly across its body) are useful, there are other challenges. Similarly, butterflies have drastically different texture and significantly different wings. The oak leaf has varying number of veins and pointed tips in its leaf depending on the particular instance and characteristics such as age. While the English ivy presents a reliable structure that can be repeatedly identified, the oak leaf presents contrasting challenges. The dolphin seen in completely different views reveals different occluding contours on its surface, which must be correlated. The pose of each instance and issues like occlusion add to the ambiguity in the structure of that 3D object.

- Specific image-based observations such as creases, singularities can also be integrated into the framework and used to more accurately model the 3D surface.
- Our solution allows the representation and building of topologically complex surfaces of virtually any genus (any number of holes).

6.1.2 Directions of future research

We have only explored the tip of the problem of silhouette-based single view reconstruction. There are many interesting and challenging problems that must be addressed. Some of them are noted below:

- We frame the reconstruction problem as a linearly constrained quadratic programming problem. Our smoothness metric is a simple, second-order smoothness. However, several other smoothness metrics such as normal curvature, Gaussian curvature, integrals of smoothness functions over piece-wise domains *etc.* must be considered. Typically, using a complex smoothness prior makes the optimization non-linear and non-convex. The challenge is to retain the simplicity of the solution while using more sophisticated smoothness priors and surface representations. The ideal smoothness prior would also make *ad hoc* inflation redundant by automatically favouring inflated solutions, therefore resulting in a much more elegant framework.
- As long as we still need inflation, it is important to investigate robust ways of inflation other than the *ad hoc*, complex ones used currently. While some automated inflation is derived from silhouettes and distance transforms, there is a need for an investigation of a broader variety of intelligent inflation mechanisms *e.g.* inflationary potential fields as those used in [191].
- In our current method, the mapping of constraints on the parametric surface is approximately set by hand. However this can be made more flexible by allowing reparametrization of the constraint mapping. This also allows the parameter space

to redistribute over the object surface more uniformly. This is equivalent to a joint optimization of the surface coordinates $\mathbf{X}(u, v)$ at vertices along with the parameter assignment (u, v) at those vertices. This is a topic which would be interesting for further exploration.

- Different surface representations have various powers. *E.g.* implicit surfaces can be used easily to represent complex topology, while parametric surfaces (which we use), allow the optimization of complex functionals in a direct fashion. It would be interesting to see which of these representations is suited to specific tasks. In our work in § 3.9, we show how the limitations of parametric surfaces can be overcome so we can represent complex topologies. It would be interesting to see how the limitations of other representations can be overcome.

6.2 Deformable object reconstruction from multiple images of distinct class instances

There are two important aspects to class-based information in reconstruction: (i) learning the class model; (ii) using the class model to predict the 3D model for an image. In chapter 4 we propose a method for learning deformable wireframe-based object class models from photo collections.

6.2.1 Contributions

- Different instances of an object class have widely varying features and sometimes features (such as spots on a lily petal) may not be repetitive making reliable correspondences hard to find. Unlike video there is also no temporal smoothness in the frames of a photo collection. Existing NRSfM techniques cannot be employed for the development of deformable shape models in such cases.

- The ribs of a petal form reliably identifiable curves on the object which *correspond* to each other across images. We show that these can be used to define the higher-level primitive of curve correspondences instead of point correspondences.

- We show that the problem of finding the deformable shape model (in terms of the 3D curves forming the object wireframe) and all the unknown parameters (cameras *etc.*) can be now framed in terms of a unified objective function which uses the image-based curves as observations. This unified objective can be effectively and incrementally solved to retrieve a good deformable model for the class.
- We show how this objective can be solved in its combined analytic form. We also show how by allowing some discretization error, the process can be sped up to yield solutions more quickly.
- Traditionally reconstruction (even for the rigid case) has been framed as a two step process: find correspondences, and then reconstruct the 3D model. Iterative methods toggling between the two have been explored. Our most novel contribution is to elegantly combine the problem of finding dense correspondences and estimating all other variables simultaneously in a principled bundle adjustment framework.

6.2.2 Directions for future research

NRSfM has been an extensively studied problem in recent times. In our work we assume that we know which curves are in correspondence across images, while allowing the corresponding points themselves to vary during optimization. First a method to extend this is discussed below.

Bundle adjustment for deformable object reconstruction with more flexible correspondences

The next natural step to our work is to include the search for corresponding curves in the optimization. The objective function being non-convex, will continue to rely on incremental

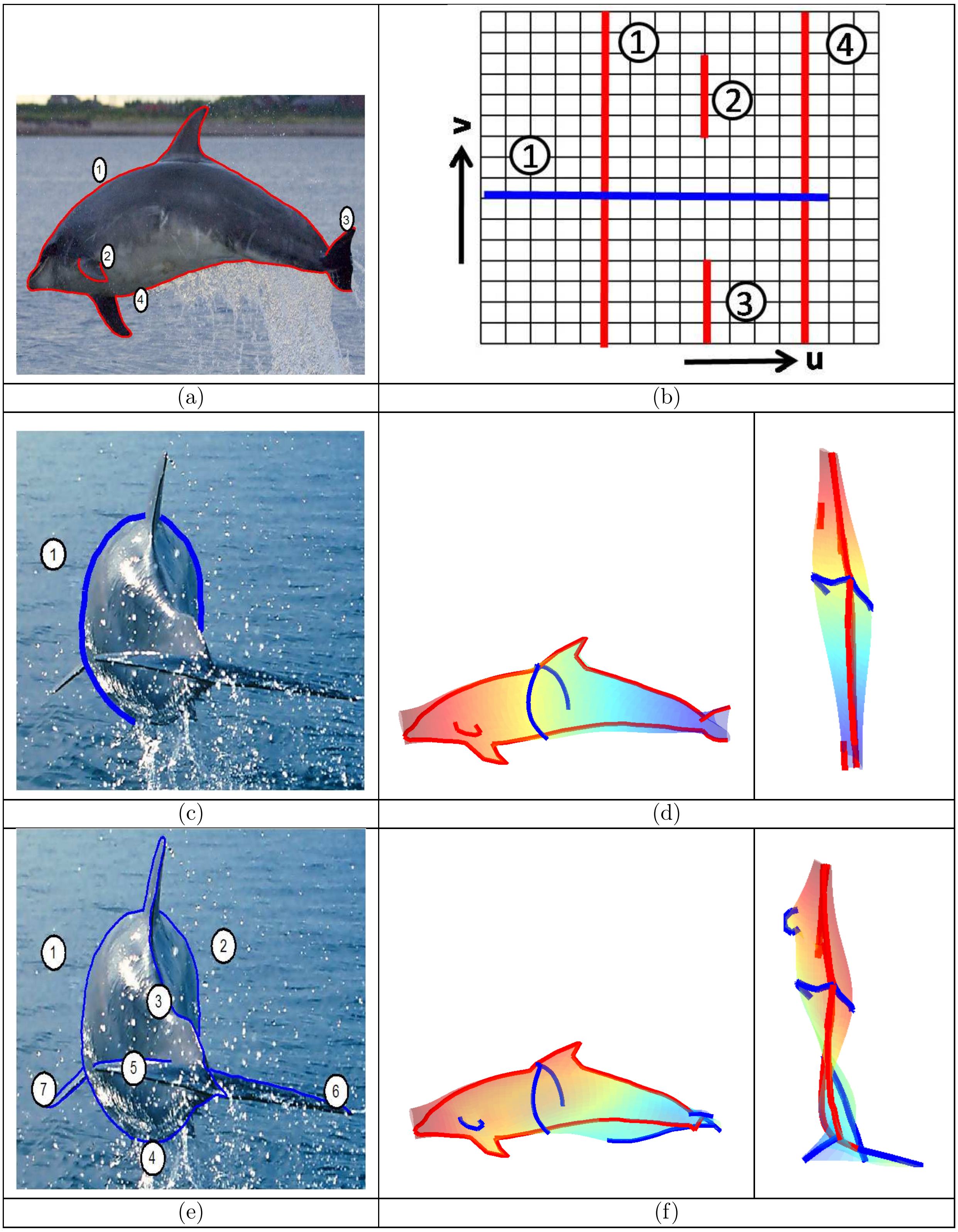


Figure 6.2: Deformable dolphin reconstruction: Methods from chapters 3 and 4 can be extended. (a,c) display different silhouettes of two dolphins, shown on the parametric surface in (b). This information can be related to create the 3D models shown in (d). (e) Many more silhouette parts can be used to create a more constrained model (f).

learning with good initializations, but will allow for automation of the complete procedure, thereby providing the ultimate test.

Another approach is to learn deformable object models from far more generic information available in images. Consider the dolphin photographed from completely different views, with the prominent curves marked out in figures 6.2 (a,c). The silhouettes (and other features) in the two images correspond to different parts of the unknown 3D object surface. While it is difficult to find corresponding curves in the two images, it is easier to identify the relative relationship that the curves in each image have to the unknown 3D deformable dolphin. Note, that different object parts may be occluded in each image (see figure 6.1). Each image (a,c) in figure 6.2 can be used independently to perform silhouette based reconstruction (using the red and silhouette respectively, as marked in figure 6.2 (b)) as described in chapter 3, with simple automated or user-inserted inflation. A contrasting approach is use the silhouette information from one view as inflation constraints for the other image's reconstruction. This involves optimizing jointly over 3D structure and camera position. Preliminary results are seen in figure 6.2 (d,f) for varying number of curves used across the second image(c,e) with respect to the base image (a).

Chapter 3 assumes constant correspondences between the silhouette the object's parametric surface. But the techniques of varying correspondences introduced in chapter 4 can be used to simultaneously allow the location of the silhouette constraints on the object's mesh to be optimized along with the deformable 3D structure and camera parameters. When performed over a large number of images this is a natural extension of our work in deformable object reconstruction.

Other possible extensions are stated below:

- Typically the observations are far fewer than the number of variables required to be found. This necessitates the use of good priors over variables and smoothness regularizers. In our work, smoothness is costly as it adds a lot of terms to our optimization. The investigation of alternative techniques is important. The PPCA based hierarchical regularization which was proposed by Torresani *et al.* [194] regularizes the problem by enforcing priors on the latent variables thus adding few terms to the optimization. It would be interesting to compare our method against an appropriate extension of their work.

- Other structure primitives (other than our wireframes) can be considered in order to constrain the problem meaningfully. One example is to consider surface representation by the more powerful parametric surface representation. Classes such as 'butterflies' (see figure 6.1) are similar to wireframe represented lilies, but introduce the novel challenge of having parts that only show rigid motion. 'Oak leaves' (see figure 6.1) pose new challenges by having an unpredictable number of corners, thereby needing a more flexible object representation than that provided by the linear PDMs (*e.g.* the multi-modal mixture model distributions of [40]) . Exploration of such object categories opens up new challenges and avenues.

- A large part of the work has gone towards writing efficient optimization code, demanding quick and efficient computation of the Jacobian. Currently this code must be optimally written by the programmer. There is a need for higher level programs that write efficient code to compute a Jacobian given a specific problem. This is an extremely hard problem to address and one of the active areas of research in numerical optimization [127, 78].

- Occlusion is one of the major challenges of multiple view reconstruction of objects, especially for deformable models. Typically, this is handled by first finding a set of reliable correspondences to initialize the framework and then to predict the missing features. However, our novel wireframe based object representation allows for new approaches to handling occlusion. Curve subsets can be represented and discovered in the parameter space. This ability to treat visible parts of the curve in the analytic framework presents promising avenues.

6.3 Class-specific object detection and segmentation

Class-based reconstruction from single images is a hard problem and inevitably needs some user-assisted or externally obtained knowledge of the specific problem at hand. The need for automation in this process has often been emphasized. In Chapter 5 we show how object characteristics can be learnt to enable automatic detection and segmentation of an object from an image. Such a step is essential to bypass annotation required for reconstruction.

6.3.1 Contributions

- Image edges provide a compact representation of an image and display some invariance to imaging conditions. They are vital for detection algorithms such as chamfer matching. Edge-like intensity gradient terms are also incorporated in modern graph-based segmentation tools such as GRABCUT, ObjCut *etc.*

- Existing edge-based approaches are generic and do not incorporate class-based information in their use of edges. However, we identify the fact that image edges have differing image intensity distributions in their neighbourhood depending on whether they belong to an object class boundary or not. We show how basic RGB pixel-based features extracted from the neighbourhood of image edges can be used to learn an effective classifier to differentiate object class boundary from other edges.

- We then use this low-level classification with higher level detection algorithms based on hierarchical chamfer matching to improve object detection. This results in fewer false detections, reduced need for hypothesis verification and faster performance.

- Instead of using only intensity and gradient based information from edges, we combine the class-specific edge learning in the pairwise terms of the ObjCut CRF, results in better segmentation.

6.3.2 Directions for future research

- We use a simple patch-based feature and leave the hard work for the classifier. In many other methods such as [128], complex filter responses are used as features for learning transparent glass boundaries in images. It would be interesting to pit our method against this to see the trade-off between the two methods. Also tougher object classes with more difficult texture such as "apples" can be explored.

- We use an SVM based classifier. It would be interesting to explore more sophisticated features, other learning frameworks–Adaboost [65], Random Forests [25] *etc.* Even within the SVM framework, the possibility of using more sophisticated kernels needs to be further explored.

- Detecting and object and estimating its pose is a complex problem. In the continuous domain it is prone to many local minima and subject to initialization. In the discrete domain, possible solutions can be searched more exhaustively. Branch-and-bound methods have been proposed to effectively detect and object and its pose. However, the performance of such methods is limited by the quality of the tree of hierarchically organized templates. Better metrics of template similarity and more effective tree building to ensure effective bounds are much needed.

- In our work, we have dealt with the detection of single instances of inherently rigid objects. The next step would be to extend this to non-rigid objects such as articulated objects with the pictorial structures framework (also used by ObjCut). Levin and Weiss [113] have extended ObjCut to perform learning in one integrated step instead of the different stages. It would be interesting to explore how edge learning, as in our method, can be integrated with their method.

Appendix A

Annotation

Annotation is an important part of gleaning information from data. For effectiveness, the annotation has to be adequately informative and intuitive. A lot of the work in this thesis deals with creating outlines and segmentations for training and verification of performance. We discuss the tools used and developed during the course of this thesis.

A.1 Drawing tools

We often need curve annotations denoting a curve of interest such as an object boundary. The goal can be spelt out as follows:

- To retrieve a curve representation in the form of points, splines etc.
- Maintaining order of extraction *i.e.* we should be able to retrieve points from one end of the curve to the other (as needed in Chapter 3).
- The curves should be automatically drawn towards nature image features such as edges. One such example is a 'snake' [96].
- The user should have adequate and easy control over drawing, and editing the curves.
- There should be ways of trading off smoothness over accuracy and *vice versa*.

There are some tools available from popular image manipulation softwares. The *lasso* completely depends on human input and allows the user to draw a contour around an object of interest. Therefore the output quality depends on the skill of the annotator with the mouse. Another boundary drawing tool is the *Magnetic Lasso (Photoshop, [2])*–lasso 'magnetized' to image edge features. With minor variations these are also known as *Live Wire [50], Lazy Snapping [114], Intelligent Scissors (GIMP, [137]) etc.* This family of algorithms uses a point based representation and a dynamic programming approach to trace the path of a curve from a fixed starting point to the current position of the mouse. As the user draws the contour, control points are inserted in the curve by manual clicking. Each control point becomes a new fixed-starting point. The performance depends on high contrast of foreground against background and can be run in real time. It depends on a good initial seed point and parameters such as width of search window, contrast sensitivity *etc.* Some software such as Photoshop automatically insert control points for the user given an addition parameter–frequency of control points. This method is impressive, but has its share of drawbacks. Though backtracking is allowed, other forms of post-drawing editing are usually not in place. Although this is a boundary drawing method in principle, most applications assume segmentation follows and therefore, do not return the curve itself. The basic algorithm doesn't ensure that the detected curve is close to the trajectory of the mouse motion. Some workarounds have been proposed in recent times. Therefore in difficult regions, the solution is to insert many small segments with many control points, which can be a cumbersome process. Other methods could also be used for editing. Despite the presence of some anti-aliasing, the final output can look a bit jagged as the method is implemented on pixel-based grids with an inadequate incorporation of smoothing.

Another such tool for boundary annotation are snakes. These are energy minimizing curves, so called due to the wriggling motion they undergo while minimizing their energy functions. The energy function is a combination of an internal energy function which determines their elasticity and curvature, and an external energy function based on image information and user interaction. The user must give a good initialization. The classic implementation of snakes by Kass *et al.* [96] allows the problem to be reduced to a matrix form. However this puts constraints on the energy functions. Davison *et al.* [44] propose a less complicated form of the energy functions, and energy minimization is carried out by adjusting individual vertices on the snakes. This allows for a greater range of energy functions, and the addition of internal energy functions like area and symmetry terms without complicating the minimization process. However it gives only a local solution. Also the snake may be prone to contraction without appropriate gradients. Also subsequent editing is not easy in the new approach.

Our tool is snake-like and takes user-drawings in the form of a sequences of points as input (Figure A.1(a)). After initially fitting a piece-wise smooth spline to this set, the image-gradients along the normals to the curves are examined. At each control point, the location of highest gradient in a small interval along the normal is identified. This spline controlled point is moved to this new location. This forms the step of automated latching provided by the system as shown in Figure A.1(b). This ensures that the user does not have to work too hard at getting the initial annotation right. In contrast to snakes, if the end points are unconstrained, this doesn't lead to immediate contraction of the curve. After this auto-correction stage, the user gets control and can change the position of control points deliberately in order to correct or refine estimates of the object boundary. Dragging any control point will automatically adjust the rest of the curve to smoothly follow and automatically adjust the curve resolution to ensure the detail in the image is uniformly captured. An artificial example of such a modification is demonstrated in Figure A.1(c) to demonstrate the flexibility of this piece-wise spline representation. This method depends on some parameters too, such as width of searching (along normals, during the latching phase) and frequency of control points (controlling the maximum undulation achievable by the curve). However, the advantages are as follows. It returns a *smooth sub-pixel, closed-form representation* for the curve. It makes use of the user trajectory during annotation to effective latch on the desired object. The method allows for great ease in editing. Smoothness of boundary is explicitly encoded in the representation, therefore in addition to lying on the boundary, it returns a smooth (naturally anti-aliased) curve that can be adjusted as desired and also subsequently used for segmentation (see figure A.5).

A.2 Edge selection tool

Annotation often involves dividing data into positive and negative examples of a concept. As shown in Chapter 5, we could be interested in differentiating edges that belong to an object boundary from others. Edge features can be reliably retrieved from algorithms such as Canny. A simple algorithm based on proximity of edge points can be used to group them into edge chains. The chains are curves which are encoded with information such as shape and normal at each point in addition to the fact that they're placed on interesting image gradient. Therefore these edge chains can be used for higher level vision tasks. For *e.g.* we may want to delineate object boundaries or mark features such as creases on the object surface, without relying on the user's skill in annotation. The process now involves marking the discovered edge chains. The system detects the current edge chain of interest by selecting the chain closest to the user's mouse at any instant and finalizing this annotation involves only a click. Also with a simple click, the user is given the power to cut edge chains and therefore use parts of edge chains instead of the whole. The chains can also be flipped with a click if directionality is important. With very few clicks the image can be annotated as shown in figure A.2.

A.3 Segmentation

One of the most important pieces of information about data is the location and extent of the image covered by the object, or segmentation. This is commonly wanted in vision tasks in order to learn the location and extent of the object's presence in the image, to learn

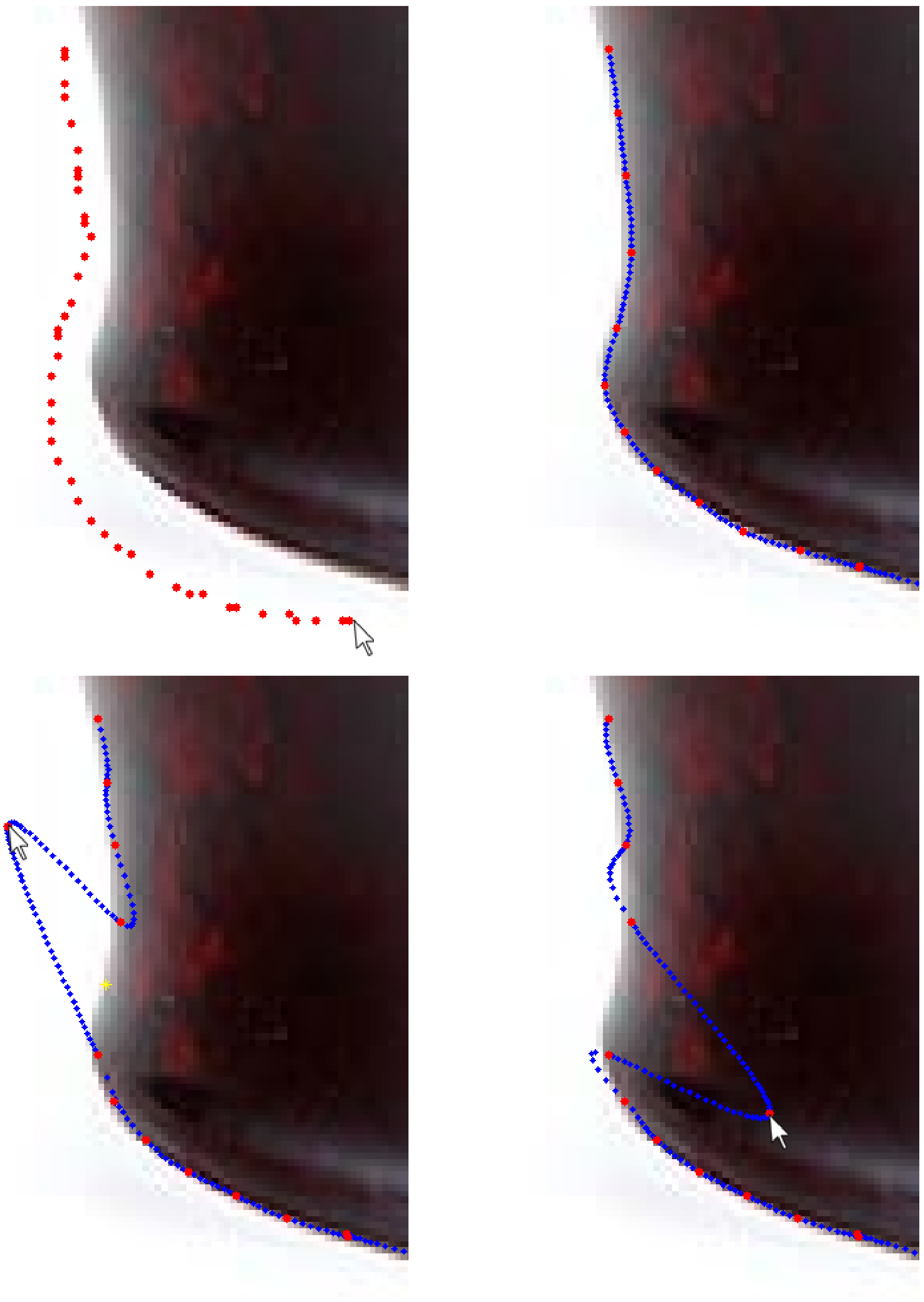
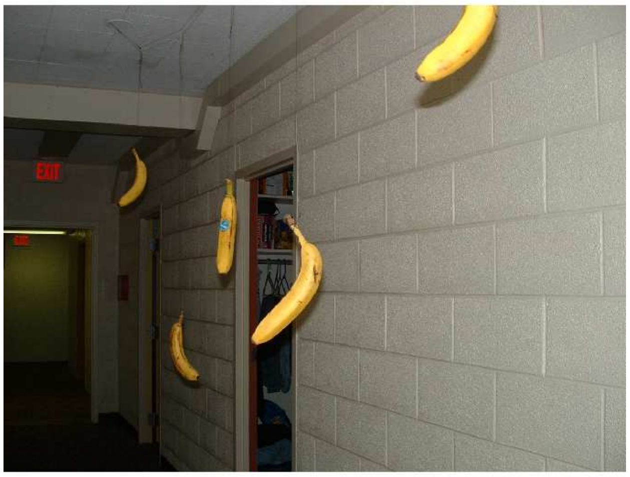
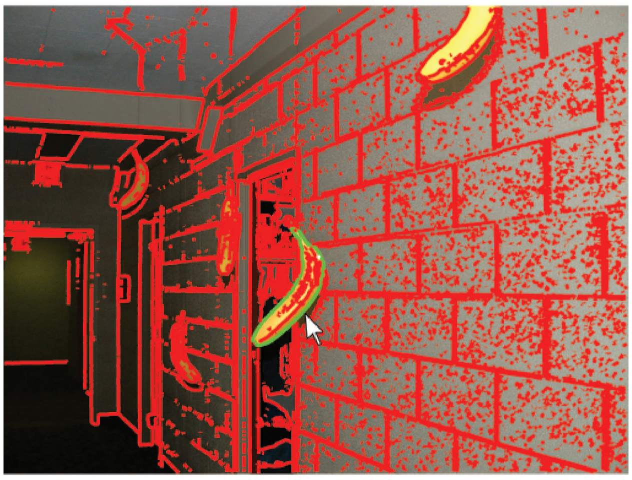

(a) (b)

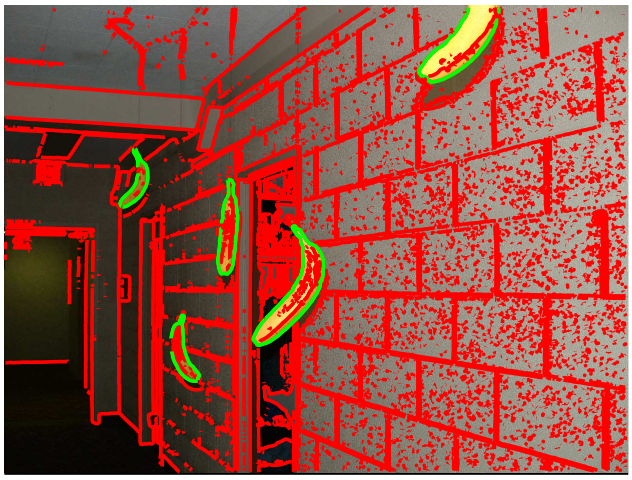
Figure A.1: Using approximate user drawing to draw a precise object boundary. Editing this representation is easy as seen above.



(a) Original image contains a number of instances of object of interest (banana) at various scales and orientations



(b) The map of edges gathered into chains is shown. Mouse-over results in selection of edge chains (current selection shown in green). Upon clicking this chain is finally annotated as a positive boundary examples.



(c) Therefore annotating the boundary of a complex image such as this can be accomplished with a few clicks.

Figure A.2: Figure annotation using edge information. The procedure is shown above.

its shape, texture and colour distribution, or simply to verify the performance of various algorithms. Higher level vision algorithms often go hand and hand with segmentation, such as detection, recognition, matting and other sophisticated image manipulation techniques.

The most basic method of segmentation is thresholding. This doesn't account for noise or intrinsic variation in colour of an object. However many graphics packages still use more sophisticated variations of thresholding. The *Magic Wand* (Photoshop, Digital Image Suite) or *Fuzzy Select* (GIMP), is one such tool. Either the user sets a seed or multiple seed pixels and a tolerance level is retrieved. All pixels are tested against the set tolerance level to get the segmentation. Many selections are required before the object can be selected completely and even then the segmentation may spill out or leave many holes which need further editing. One such result of the method can be seen in figure A.5(d) . In fact the Magnetic Lasso based approaches usually perform much better for segmentation purposes.

One of the most effective methods for segmentation rely on the min cut algorithm on image-based graphs (see [19]). The algorithm directly yields global optima for specific objective function types and results in a pixel level segmentation. Cues about foreground *vs.* background can be incorporated in many ways. One such method GrabCut ([19],[163]) depends on strokes of the user to learn image statistics, which are then used for segmentation. This family of methods form the state of the art in segmentation. However, many of these algorithms are not yet publicly available. The few available ones include *Quick Select* (Photoshop). Unfortunately, though 8 neighbourhood graphs are possible, this software only uses 4 neighbourhoods resulting in jagged contours. For segmentation this thesis sees the use of either the GrabCut based methods, or a simple polygonal fill of the closed curves retrieved by our boundary annotation method (§A.1, Figure A.3). Alternately, our boundary can be used to define weights in a Graph Cut to retrieve an improved segmentation instead of just a fill-in.

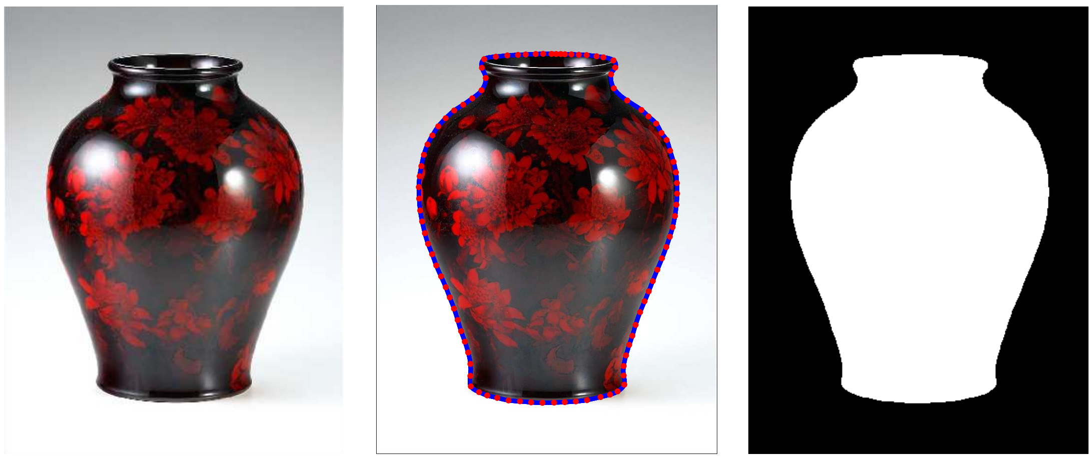
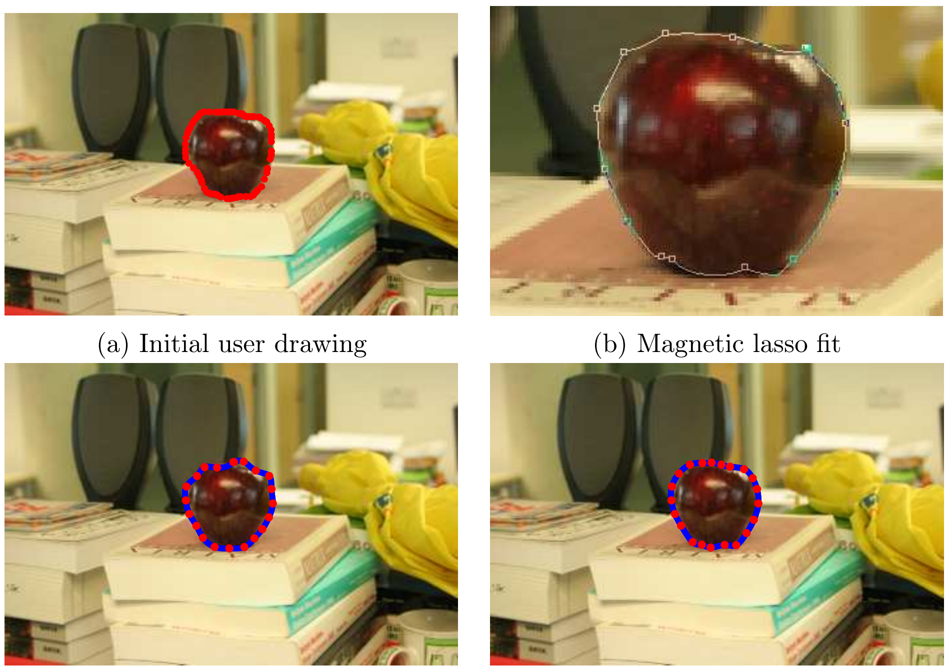


Figure A.3: A vase can be segmented easily and effectively with our method



(a) Initial user drawing

(b) Magnetic lasso fit

(c) Our latched contour from drawing

(d) Final latched contour

Figure A.4: Comparison of magnetic lasso with our method. Drawing as shown in (a) is used in both cases. (b) latches on well, but has sharp corners at control points. (c) our method latches on with a couple of stray matches. Upon correction (d) shows a smooth, near-perfect fit.

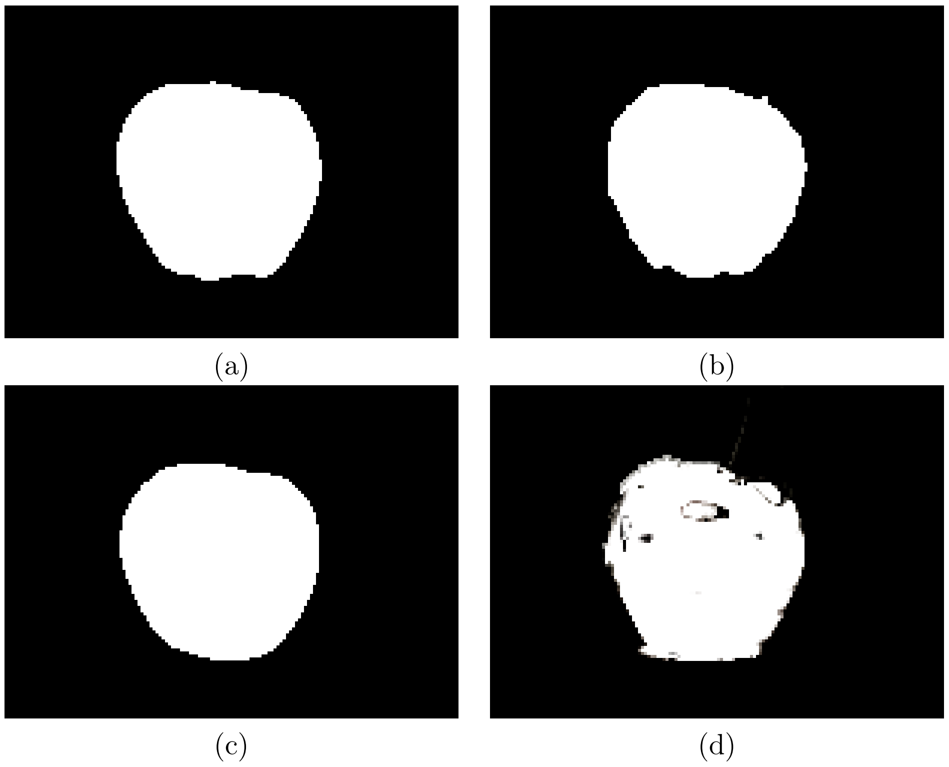


Figure A.5: Segmentations from three methods are seen: (a) Our method with a simple fill-in (b) Magnetic Lasso [2] segmentation (done as well as possible) and (c) Quick Select segmentation [3]. (c) is less jagged, however (a) follows features like the curves at the bottom of the apple better than (b),(c). (d) shows the poor relative performance of the Magic Wand [1] done despite involving twice the effort of (a-c).

Appendix B

Bundle Adjustment

We look at the problem of non-rigid structure from multiple distinct views. A recap of the notation and the variables of the problem follows in Table B.1:

Notation		
symbol	definition	description
$\mathbf{X}$	in bold font	vector
$\mathtt{X}$	typewritten font	matrix
$\pi(\underset{N\times 1}{\mathbf{X}})$	$\frac{X_{1\cdots N-1}}{X_N}$	projection of a vector
$\overset{\circ}{X}$	$\left[\begin{smallmatrix}X\\1\end{smallmatrix}\right]$	homogenizing a vector (appending 1)
$\underset{M\times 1}{\mathbf{P}_i}$	index in subscript	indicates i^{th} column of matrix $\underset{M\times N}{\mathtt{P}}$
$\underset{1\times N}{\mathbf{P}^i}$	index in superscript	indicates i^{th} row of matrix $\underset{M\times N}{\mathtt{P}}$
Problem summary in terms of variables		
variable	range/size	significance
n	$1\cdots N$	image number
p	$1\cdots P$	petal vertex or point
k	$1\cdots K$	index for the basis shape
q	$1\cdots Q$	index into parameter set $\boldsymbol{\theta}$ affecting camera matrices
$\underset{3\times 1}{\mathbf{X}_{np}}$		p^{th} 3D model vertex point for n^{th} image
$\mathbf{B}_{kp}$		p^{th} vertex for the k^{th} basis shape
α_{nk}		Contribution of k^{th} basis towards n^{th} shape
$\underset{3\times 4}{\mathtt{P}_n} = f(\boldsymbol{\theta})_n = \begin{bmatrix}\underset{3\times 3}{\mathtt{A}_n} & \underset{3\times 1}{\mathbf{t}_n}\end{bmatrix}$		Projection matrix for image n (can have many forms)
$\underset{2\times 1}{\mathbf{w}_{np}}$		image correspondence for $\mathbf{X}_{np}$ (if known)
$\underset{2\times 1}{\boldsymbol{\omega}_n(t)}$		point denoted by param t on the analytic silhouette ω_n.
$\underset{2\times 1}{\hat{\mathbf{w}}_{np}}$		projection of current 3D point $\mathbf{X}_{np}$

Table B.1: Problem recap

In our problem statement (refer to § 4.2,4.4) the only information we have available across the images is ω. For each image n, we want to find the unique 3D models made up by points–$\mathbf{X}_{np}$. In order to exploit the common structure across images, some assumptions are needed. The most popular and effective one assumes that the unknown 3D shapes lie in a linear subspace of K bases (see [23, 22, 194]) such that the following relation holds:

$$\underset{3\times P}{\mathtt{X}_n} = \sum_{k=1}^{K} \alpha_{nk} \cdot \underset{3\times P}{\mathtt{B}_k} . \tag{B.1}$$

$$\text{with derivatives: } \frac{\partial \underset{3\times 1}{\mathbf{X}_{np}}}{\partial \underset{1\times 1}{\alpha_{nk}}} = \underset{3\times 1}{\mathbf{B}_{kp}}, \tag{B.2}$$

$$\frac{\partial \underset{3\times 1}{\mathbf{X}_{np}}}{\partial \underset{1\times 3}{\mathbf{B}_{kp}^{\top}}} = \alpha_{nk} \cdot \underset{3\times 3}{\mathtt{I}} . \tag{B.3}$$

The process of projection of a 3D point $\mathbf{X}_{np}$ to form its image estimate can be summed up as follows:

$$\hat{\mathbf{w}}_{np} = \pi\left(\mathtt{P}_n \overset{\circ}{\mathbf{X}}_{np}\right) \qquad = \pi\left(\mathtt{A}_n \mathbf{X}_{np} + \mathbf{t}_{np}\right). \tag{B.4}$$

1. For known correspondence $\mathbf{w}_{np}$ the reprojection error is defined as

$$\underset{2\times 1}{\mathbf{e}_{np}} = (\underset{2\times 1}{\hat{\mathbf{w}}_{np}} - \underset{2\times 1}{\mathbf{w}_{np}}). \tag{B.5}$$

2. For unknown (variable) correspondences, the reprojection error is function of the parameter defining the correspondence too. So the distance of $\hat{\mathbf{w}}_{np}$ to the point corresponding to a parameter t on ω_n is given by:

$$\mathbf{d}_{np}(t) = \left(\hat{\mathbf{w}}_{np} - \underset{2\times 1}{\boldsymbol{\omega}_n(t)}\right). \tag{B.6}$$

The closest point to $\hat{\mathbf{w}}_{np}$ can be found by minimizing the above as follows:

$$\mathbf{d}_{np}^{\min^2} = \min_t \mathbf{d}_{np}^2(t) = \min_t \left(\hat{\mathbf{w}}_{np} - \underset{2\times1}{\boldsymbol{\omega}_n(t)} \right)^2 . \tag{B.7}$$

Note: The only difference between the two equations above is the introduction of extra parameter t. However, this does not affect the function value w.r.t. the other parameters given the same correspondence.

B.1 Objective

The goal is to find 3D shapes for each view of the set of images. The available information is in the form of corresponding image silhouette curves. This means that the most natural objective function is minimization of reprojection error.

$$\min_{\theta,\alpha,\mathtt{B}} \quad E = \min_{\theta,\alpha,\mathtt{B}} \sum_{np} \mathbf{e}_{np}^2 \quad \text{for known correspondences,} \tag{B.8a}$$

$$\text{or} \quad \min_{\theta,\alpha,\mathtt{B}} \sum_{np} \min_t \mathbf{d}_{np}^2(t) \quad \text{for variable correspondences.} \tag{B.8b}$$

The equations (B.8a) and (B.8b) are exactly equivalent given the same correspondences. Also their behaviour w.r.t. variation in the variables $\{\theta, \alpha, \mathtt{B}\}$ is identical.

Let's look at how variable correspondences are represented. In equation (B.5) the correspondences are fixed, but in equation(B.6) they take a functional form $\boldsymbol{\omega}(t)$. Each curve is a set of piecewise smooth cubic splines stored in $\boldsymbol{\omega}^x, \boldsymbol{\omega}^y$. Therefore for P points, we have $P-1$ splines which are smooth at their joins. A given t corresponds to a specific curve segment s and corresponding spline set–$\underset{4\times1}{\boldsymbol{\omega}_s^x}, \underset{4\times1}{\boldsymbol{\omega}_s^y}$. The actual point coordinates corresponding to a parameter t_{np} on curve ω_n are computed as shown.

$$\begin{aligned}
t &\in [0,1] \rightarrow p \in [1,P] \quad |\text{mapping between parameter range and point} \\
s &= \lfloor 1 + (P-1) * t_{np} \rfloor \quad |\text{spline segment for } t_{np} \\
\mathbf{w}_{np} &= \boldsymbol{\omega}(t_{np}), \\
&= \begin{bmatrix} \underset{1\times4}{\boldsymbol{\omega}_s^{x\top}} \\ \underset{1\times4}{\boldsymbol{\omega}_s^{y\top}} \end{bmatrix} \begin{bmatrix} t_{np}^3 \\ t_{np}^2 \\ t_{np} \\ 1 \end{bmatrix}.
\end{aligned} \tag{B.9}$$

This function (B.9) uses pre-computed spline coefficients $\boldsymbol{\omega}_s^x, \boldsymbol{\omega}_s^y$ corresponding to the x and y values of the segment s.

B.2 Optimization

Optimization of the objective (B.8a,B.8b) can be done using the well-known Levenberg-Marquardt algorithm (see [112, 124]). The error function is represented in its vector form (before being squared and summed) and the Jacobian of this residual w.r.t. all the variables can be used to effectively optimize the objective function. The optimization assumes a locally linear surface for the objective function but this has been observed to be a reasonably good assumption for most practical tasks.

Derivatives An important aspect of least-squares techniques such as Levenberg-Marquardt is that, similar to methods like Newton Raphson, Gauss-Newton *etc.*, its performance is greatly enhanced by continuous functions for which derivatives can be spelt out analytically. For a vector residual, this involves finding an accurate Jacobian. In the absence of analytic Jacobians, the solvers resort to finite-difference Jacobians, but this implies more function evaluations and decreased accuracy. Because of our parametrization of the problem we are able to provide analytic derivatives as shown below:

$$\underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} = \left[\underset{2\times3}{\mathtt{A}_n^{1,2}} - \underset{2\times1}{\hat{\mathbf{w}}_{np}} \otimes \underset{1\times3}{\mathtt{A}_n^{3}} \right] \frac{1}{\left(\mathtt{P}_n^3 \mathring{\mathbf{X}}_{np}\right)}, \otimes\text{: Kronecker product} \tag{B.10}$$

$$\underset{2\times1}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \theta_{nq}}} = \left[\underset{2\times4}{\frac{\partial \mathtt{P}_n^{1,2}}{\partial \theta_{nq}}} - \underset{2\times1}{\hat{\mathbf{w}}_{np}} \otimes \underset{2\times4}{\frac{\partial \mathtt{P}_n^{3}}{\partial \theta_{nq}}} \right] \underset{4\times1}{\frac{\mathring{\mathbf{X}}_{np}}{\left(\mathtt{P}_n^3 \mathring{\mathbf{X}}_{np}\right)}}. \tag{B.11}$$

These derivatives can be used for writing the derivatives for the objective function:

$$\begin{aligned}
\underset{2\times3}{\frac{\partial \mathbf{e}_{np}}{\partial \mathbf{X}_{np}}} = \frac{\partial \mathbf{d}_{np}(t)}{\partial \mathbf{X}_{np}} & = \frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}, \\
\underset{2\times1}{\frac{\partial \mathbf{e}_{np}}{\partial \alpha_{nk}}} = \frac{\partial \mathbf{d}_{np}(t)}{\partial \alpha_{nk}} & = \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} \underset{3\times1}{\frac{\partial \mathbf{X}_{np}}{\partial \alpha_{nk}}}, \\
\underset{2\times3}{\frac{\partial \mathbf{e}_{np}}{\partial \mathtt{B}_{kp}}} = \frac{\partial \mathbf{d}_{np}(t)}{\partial \mathtt{B}_{kp}} & = \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} \underset{3\times3}{\frac{\partial \mathbf{X}_{np}}{\partial \mathtt{B}_{kp}}}, \\
\frac{\partial \mathbf{e}_{np}}{\partial \theta_{nq}} = \frac{\partial \mathbf{d}_{np}(t)}{\partial \theta_{nq}} & = \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \theta_{nq}}}.
\end{aligned} \tag{B.12}$$

Variable correspondences, in practice, can be implemented in one of many ways. Each method has its own specific derivative definition and subsequent optimization.

Incorporating varying correspondences and inter-curve distances

For varying correspondences the reprojection error of a point is equivalent to the minimal distance to the image silhouette. We now examine the different ways of computing this.

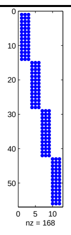
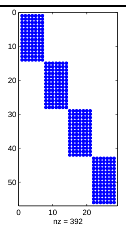
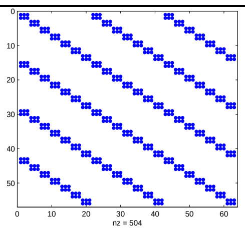
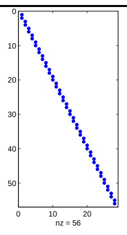
sparsity pattern for reprojection error $\mathbf{e}_{np}$				
size	$2NP \times NK$	$2NP \times NQ$	$2NP \times 3KP$	$2NP \times NP$
non-zero entries	$2NPK$	$2NPQ$	$6NPK$	$2NP$
Variables	(a) α	(b) θ	(c) $\mathtt{B}$	(d) $\mathtt{T}$

Figure B.1: Sparsity pattern for Jacobian for each variable set: The error vector (along vertical axis) composed of $\mathbf{e}_{np}$ is ordered by p first and then stacked for each n. α, θ (along horizontal axis, column (a,b)) are stacked by k, q for each value of n. $\mathtt{B}$ (along horizontal, column(c)) is stacked for each point p for each value of k (basis). $\mathtt{T}$ (column (c)) is stacked according to point index p for each image n. The jacobians here are shown for the toy values of $N = 4, K = 3, P = 7$. For a scaled-orthographic projective matrix $Q = 7$. The sparsity patterns are significant because they represent the data dependencies on the different variables clearly. In order optimize the speed of optimization, it is important to take this sparsity pattern into account.

Exact computation For the parametric correspondences of equation (B.8b) we can use equation(B.9) to provide derivatives w.r.t. the correspondence parameter as:

$$\frac{\partial \mathbf{d}_{np}(t)}{\partial t_{np}} = -\frac{\partial \boldsymbol{\omega}_n(t)}{\partial t_{np}}, \tag{B.13}$$

$$= -\begin{bmatrix} \underset{1\times 4}{\boldsymbol{\omega}_s^{x\top}} \\ \underset{1\times 4}{\boldsymbol{\omega}_s^{y\top}} \end{bmatrix} \begin{bmatrix} 3t_{np}^2 \\ 2t_{np} \\ 1 \\ 0 \end{bmatrix}. \tag{B.14}$$

For each point p in each image n, the reprojection error can be found for equation (B.8b) by performing a Newton Raphson optimization using equations (B.9,B.14). The accuracy in this method comes at the expense of computation.

Distance transform: The need for $N \times \mathrm{P}$ optimizations of equation (B.9) in each function evaluation of the objective (B.8b) is computationally expensive.

$$\mathbf{d}_{np}^{\min^2} = \min_t \mathbf{d}_{np}^2(t) \tag{B.15}$$

By approximating the minimal distance of a point to the silhouette by using distance transforms, the computation can be considerably reduced. Computing the distance transform DT_n of a curve ω_n for a grid of some reasonable resolution is a one-off computation.

$$\underset{2\times1}{\mathrm{DT}_n(\mathbf{w}_{np})} \approx \min_t (\hat{\mathbf{w}}_{np} - \boldsymbol{\omega}_n(t)) \quad (\text{or, } \mathbf{d}_{np}^{\min})$$

The typical DT returns a scalar value, but similar to $d^{\min}$ we store the (x, y) parts of the distance separately. If the resolution of the grid on which the DT is computed is δ, then the approximation error due to DT is at most δ and usually considerably less. Therefore, for any point $\hat{\mathbf{w}}_{np}$ a simple linear interpolation can be performed upon the DT at the given resolution to retrieve $\mathrm{DT}_n(\mathbf{w}_{np})$. Storing the DT at a reasonable resolution to improve accuracy causes memory constraints (especially when storing curves across N different images). Therefore the speed of DT comes at the cost of accuracy (in turn limited by memory). The objective function can be now written as:

$$E = \sum_{np} \mathbf{e}_{np}^2 = \min_{\theta,\alpha,\mathtt{B}} \sum_{np} \mathrm{DT}_n^2(\hat{\mathbf{w}}_{np}). \tag{B.16}$$

Since using DTs changes the objective function express the derivatives and how they're expressed. Once computed the value of the DT is a function of only $\hat{\mathbf{w}}_{np}$. Therefore, finite-difference derivatives can be computed for the DT as shown:

$$\frac{\underset{2\times1}{\partial \mathrm{DT}_n}}{\underset{2\times1}{\partial \hat{\mathbf{w}}_{np}}} \approx \underset{2\times2}{\nabla \mathrm{DT}_n}. \tag{B.17}$$

The derivatives of $\mathbf{X}_{np}$ (B.2,B.3) and $\hat{\mathbf{w}}_{np}$ (B.10,B.11) stay the same. The new derivatives for $\mathbf{e}_{np}$ are given below.

$$\underset{2\times3}{\frac{\partial \mathbf{e}_{np}}{\partial \mathbf{X}_{np}}} = \underset{2\times2}{\frac{\partial \mathrm{DT}_n}{\partial \hat{\mathbf{w}}_{np}}} \cdot \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} \approx \underset{2\times2}{\nabla \mathrm{DT}_n} \cdot \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} \tag{B.18}$$

$$\underset{2\times1}{\frac{\partial \mathbf{e}_{np}}{\partial \alpha_{nk}}} = \underset{2\times2}{\frac{\partial \mathrm{DT}_n}{\partial \hat{\mathbf{w}}_{np}}} \cdot \underset{2\times1}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \alpha_{nk}}} \approx \underset{2\times2}{\nabla \mathrm{DT}_n} \cdot \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} \cdot \underset{3\times1}{\frac{\partial \mathbf{X}_{np}}{\partial \alpha_{nk}}} \tag{B.19}$$

$$\underset{2\times3}{\frac{\partial \mathbf{e}_{np}}{\partial \mathtt{B}_{kp}}} = \underset{2\times2}{\frac{\partial \mathrm{DT}_n}{\partial \hat{\mathbf{w}}_{np}}} \cdot \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathtt{B}_{kp}}} \approx \underset{2\times2}{\nabla \mathrm{DT}_n} \cdot \underset{2\times3}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \mathbf{X}_{np}}} \cdot \underset{3\times3}{\frac{\partial \mathbf{X}_{np}}{\partial \mathtt{B}_{kp}}} \tag{B.20}$$

$$\underset{2\times1}{\frac{\partial \mathbf{e}_{np}}{\partial \theta_{nq}}} = \underset{2\times2}{\frac{\partial \mathrm{DT}_n}{\partial \hat{\mathbf{w}}_{np}}} \cdot \underset{2\times1}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \theta_{nq}}} \approx \underset{2\times2}{\nabla \mathrm{DT}_n} \cdot \underset{2\times1}{\frac{\partial \hat{\mathbf{w}}_{np}}{\partial \theta_{nq}}} \tag{B.21}$$

The derivative is a combination of finite-difference and analytic terms. However the use of a finite-difference derivative for DT−∇DT ensures the derivatives again involve only an interpolated table look-up thus avoiding expensive computation.

Slack approach Let's take another look at equation (B.8b). By pulling the internal minimization into the larger objective function, we can avoid minimization within each function evaluation of E.

$$\begin{aligned} E &= \min_{\theta,\alpha,\mathtt{B}} \sum_{np} \min_t \mathbf{d}^2_{np}(t) \quad \text{original objective,} \\ &\approx \min_{\theta,\alpha,\mathtt{B},\mathtt{T}} \sum_{np} \mathbf{d}^2_{np}(t) \quad \text{new slack objective.} \end{aligned} \tag{B.22}$$

The derivatives for this objective stay the same as shown in equations (B.12, B.14). This means that the variables of the internal minimization– $\underset{N\times P}{\mathtt{T}}$ are added to the set of all variables to be optimized over. Given a 3D point and a projection matrix, the closest point correspondence on an image curve is full determined. However, by introducing extra redundant variables, the optimization is made easier despite the addition of $N \times P$ variables. The added variables introduce a very sparse almost diagonal part to the final Jacobian for optimization.

B.3 Conclusion

This note demonstrates different techniques that can be used in the optimization of reprojection error for construction of deformable object models. In particular, analytic and semi-analytic approaches are demonstrated and it shown how jacobians can be constructed for each case for effective minimization using bundle adjustment. Though reprojection error is used here, the method can be extended to variations of the energy as discussed in §4.4,4.6.1.

Bibliography

[1] Use the magic wand tool. `http://help.adobe.com/en_US/PhotoshopElements/7.0_Win/WSae2ea3b149d0c3591ae939f103860b3d59-7ec1.html`.

[2] Use the magnetic lasso tool. `http://help.adobe.com/en_US/PhotoshopElements/7.0_Win/WSae2ea3b149d0c3591ae939f103860b3d59-7ec3.html`.

[3] Use the quick selection tool. `http://help.adobe.com/en_US/PhotoshopElements/7.0_Win/WS594384B5-E15C-42ee-9EFC-704E01142183.html`.

[4] 2d3 Ltd. Boujou: Automated camera tracking, 2003. http://www.2d3.com.

[5] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. ECCV*, pages 113–130, 2002.

[6] A.H. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[7] H. G. Barrow and J. M. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[8] B. Bascle and A. Blake. Separability of pose and expression in facial tracking and animation. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 323, Washington, DC, USA, 1998. IEEE Computer Society.

[9] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential update of projective and affine structure from motion. *IJCV*, 23(3):235–259, 1997.

[10] A. Beinglass and H.J. Wolfson. Articulated object recognition, or: how to generalize the generalized Hough transform. In *Proc. CVPR*, pages 461–466, Jun 1991.

[11] M.O. Berger. Snake growing. *Proc. ECCV*, 90:570–572, 1990.

[12] R. Berthilsson, K. Åström, and A. Heyden. Reconstruction of curves in $\mathbb{R}^3$, using factorization and bundle adjustment. In *Proc. ICCV*, volume 1, pages 674–679, 1999.

[13] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[14] M. J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. ICCV*, pages 374–381, 1995.

[15] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[16] J. Bloomenthal. Implicit surfaces. In Jack Belzer, editor, *Encyclopedia of Computer Science and Technology*. Marcel Dekker, Inc., New York, NY, USA, 1980.

[17] F. Bookstein. Principal warps : Thin-plate splines and the decomposition of deformations. *IEEE PAMI*, 2(6):567–585, Jun 1989.

[18] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proc. ECCV*, pages 109–124, 2002.

[19] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. ICCV*, volume 2, pages 105–112, 2001.

[20] Y.Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE PAMI*, 26(9):1124–1137, September 2004.

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

[22] M. Brand. Morphable 3D models from video. In *Proc. CVPR*, volume 2, pages 456–463, 2001.

[23] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Proc. CVPR*, volume 2, pages 690–696, 2000.

[24] C. Bregler and S. M. Omohundro. Surface learning with applications to lipreading. In *Advances in Neural Information Processing Systems 6*, pages 43–50. Morgan Kaufmann Publishers, 1994.

[25] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[26] M. J. Brooks and B. K. P. Horn. Shape and source from shading. In *Proc. Intl. Joint Conf. on Artificial Intelligence*, pages 932–936, 1985.

[27] J. F. Canny. Finding edges and lines in images. Master's thesis, MIT, 1983.

[28] J. F. Canny. A computational approach to edge detection. *IEEE PAMI*, 8(6):679–698, 1986.

[29] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, February 1997.

[30] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *IJCV*, 9(2):83–112, 1992.

[31] L.D. Cohen and I. Cohen. Deformable models for 3D medical images using finite elements & balloons. *Proc. CVPR*, 92:592–598, 1992.

[32] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24(5):603–619, 2002.

[33] T. Cootes. Model-based methods in analysis of biomedical images. In E. R. Baldock and J. Graham, editors, *Image Processing and Analysis*, pages 223–248. Oxford University Press, 2000.

[34] T. Cootes and C. Taylor. Data driven refinement of active shape model search. In *Proc. BMVC.*, volume 1, pages 383–392, 1996.

[35] T. F. Cootes and C. J. Taylor. Modelling object appearance using the grey-level surface. In *th British Machine Vison Conference*, pages 479–488. BMVA Press, 1994.

[36] T. F. Cootes and C. J. Taylor. Constrained active appearance models. *Proc. ICCV*, 1:748, 2001.

[37] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models—their training and application. *CVIU*, 61(1):38–59, 1995.

[38] T. F. Cootes, C. J. Taylor, and A. Lanitis. Active shape models: Evaluation of a multi-resolution method for improving image search. In *Proceedings of the British Machine Vision Conference*, pages 327–336. BMVA Press, 1994.

[39] T. F. Cootes, C. J. Twining, and C. J. Taylor. Diffeomorphic statistical shape models. In *Proc. BMVC.*, pages 447–456, 2004.

[40] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, 17(8):567–573, June 1999.

[41] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998.

[42] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 10–17, Washington, DC, USA, 2005. IEEE Computer Society.

[43] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *IJCV*, 40(2):123–148, Nov 2000.

[44] N.E. Davison, H. Eviatar, and R.L. Somorjai. Snakes simplified. *Pattern Recognition*, 33(10):1651–1664, October 2000.

[45] A. Del Bue. A factorization approach to structure from motion with shape priors. In *Proc. CVPR*, 2008.

[46] P. Dollar, T. Zhuowen, and S. Belongie. Supervised learning of edges and object boundaries. In *Proc. CVPR*, 2006.

[47] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proc. ACM SIGGRAPH*, pages 173–182, New York, NY, USA, 1995. ACM.

[48] A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. ACM SIGGRAPH*, pages 341–346, Aug 2001.

[49] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, pages 1039–1046, Sep 1999.

[50] A. X. Falcao, J. K. Udupa, S. Samarasekera, and S. Sharma. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing*, 60(4):233–260, July 1998.

[51] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Boston, 1990.

[52] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[53] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. In *Proc. CVPR*, volume 2, pages 2066–2073, 2000.

[54] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. CVPR*, volume 2, pages 264–271, Jun 2003.

[55] V. Ferrari, T. Tuytelaars, and L.J. Van Gool. Object detection by contour segment networks. In *Proc. ECCV*, pages III: 14–28, 2006.

[56] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, c-22(1):67–92, Jan 1973.

[57] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. In *Proc. BMVC.*, pages 662–670, 2001.

[58] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[59] J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[60] J. L. R. Ford and D. R. Fulkerson. *Flows in networks.* Princeton University Press, Princeton, NJ, USA.

[61] D. A. Forsyth. Shape from texture without boundaries. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 225–239, London, UK, 2002. Springer-Verlag.

[62] D. A. Forsyth and J. Ponce. *Computer Vision: A modern approach.* Prentice Hall, 2002.

[63] R. T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):439–451, 1988.

[64] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[65] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[66] Y. Furukawa, A. Sethi, J. Ponce, and D.J. Kriegman. Robust structure and motion from outlines of smooth curved surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):302–315, Feb. 2006.

[67] D. Gavrila. Pedestrian detection from a moving vehicle. In *Proc. ECCV*, pages II: 37–49, 2000.

[68] D. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *Proc. ICCV*, pages 87–93, 1999.

[69] I. M. Gelfand, S. V. Fomin, and R. A. Silverman. *Calculus of Variations.* Dover Publications, 2000.

[70] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE PAMI*, 6(6):721–741, Nov 1984.

[71] T. Gevers, S. Ghebreab, and A. W. M. Smeulders. Color invariant snakes. In *Proc. BMVC.*, pages 578–588, 1998.

[72] P. Giblin and R. Weiss. Reconstruction of surfaces from profiles. In *Proc. ICCV*, pages 136–144, London, 1987.

[73] J. J. Gibson. *The Perception of the Visual World.* Houghton Mifflin, Boston, 1950.

[74] J. J Gibson. *The Ecological Approach to Visual Perception.* Houghton Mifflin Co., Boston, Massachusetts, 1979.

[75] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.

[76] C. Goodall. Procrustes methods in the statistical analysis of shape. In *Journal of the Royal Statistical Society*, volume 53, pages 285–339, 1991.

[77] Y. Grandvalet, J. Mariéthoz, and S. Bengio. A probabilistic interpretation of SVMs with an application to unbalanced classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 467–474. MIT Press, Cambridge, MA, 2006.

[78] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming. `http://stanford.edu/~boyd/cvx`, June 2009.

[79] A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica.* CRC Press, Inc., Boca Raton, FL, USA, 1996.

[80] W. E. L. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Visual System.* MIT Press, 1981.

[81] W. E. L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE PAMI*, 9(4):469–482, 1987.

[82] X. Gu, S. Gortler, and Hoppe. H. Geometry images. In *Proc. ACM SIGGRAPH*, pages 355–361, 2002.

[83] N.A. Gumerov, A. Zandifar, R. Duraiswami, and L.S. Davis. Structure of applicable surfaces from single views. In *ECCV04*, pages Vol III: 482–496, 2004.

[84] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[85] T. Heap and D. Hogg. Automated pivot location for the Cartesian-polar hybrid point distribution model. In *Proc. BMVC.*, pages 97–106. BMVA Press, 1996.

[86] T. Heap and D. Hogg. Improving specificity in PDMs using a hierarchical approach. In *Proc. BMVC.*, pages 80–89, 1997.

[87] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 24(3):577–584, 2005.

[88] B. K. P. Horn, R. S. Szeliski, and A. L. Yuille. Impossible shaded images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(2):166–170, 1993.

[89] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[90] T. Ijiri, S. Owada, M. Okabe, and T. Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Trans. Graph.*, 24(3):720–726, 2005.

[91] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *In Proc. ICCV*, pages 626–633, 1999.

[92] J. Y. Kaminski and A. Shashua. Multiple view geometry of general algebraic curves. *IJCV*, 56(3):195–219, 2004.

[93] K. Kanatani and T. Chou. Shape from texture: General principle. *AIJ*, 38:1–48, 1989.

[94] O. Karpenko, J. F. Hughes, and Raskar. R. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3):585–594, 2002.

[95] O. A. Karpenko and J. F. Hughes. SmoothSketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics*, 25(3):589–598, July 2006.

[96] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. ICCV*, pages 259–268, 1987.

[97] M. Keller, R. Knothe, and T. Vetter. 3D reconstruction of human faces from occluding contours. In *Model-based Imaging, Rendering, Image Analysis and Graphical special Effects*, volume 4418 of *LNCS*, pages 261–273. Springer, 2007.

[98] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.

[99] J. J. Koenderink. *Solid Shape.* MIT Press, 1990.

[100] P. Kohli, M.P. Kumar, and P.H.S. Torr. P3 and beyond: Solving energies with higher order cliques. In *Proc. CVPR*, pages 1–8, 2007.

[101] P. Kohli and P.H.S. Torr. Dynamic Graph Cuts for efficient inference in Markov Random Fields. *IEEE PAMI*, 29(12):2079–2088, December 2007.

[102] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. ACM SIGGRAPH*, pages 313–324, New York, NY, USA, 1996. ACM.

[103] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proc. CVPR*, 2005.

[104] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[105] X.Y. Lan, S. Roth, D.P. Huttenlocher, and M.J. Black. Efficient Belief Propagation with learned higher-order Markov random fields. In *Proc. ECCV*, pages II: 269–282, 2006.

[106] A. Lanitis, C. J. Taylor, and T. F. Cootes. A unified approach to coding and interpreting face images. In *Proc. ICCV*, pages 368–373, 1995.

[107] A. Lanitis, Chris J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, 1997.

[108] S. Lazebnik and J. Ponce. The local projective shape of smooth surfaces and their outlines. In *Proc. ICCV*, page 83, Washington, DC, USA, 2003. IEEE Computer Society.

[109] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces, 1998.

[110] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC03*, volume 2, pages 264–271, 2003.

[111] V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *Proc. ECCV*, pages IV: 15–29, 2008.

[112] K. Levenberg. A method for the solution of certain problems in least squares. In *Quarterly Applied Mathematics*, volume 2, 1944.

[113] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. *Int. J. Comput. Vision*, 81(1):105–118, 2009.

[114] Y. Li, J. Sun, C. Tang, and H. Shum. Lazy snapping. In *Proc. ACM SIGGRAPH*, pages 303–308, New York, NY, USA, 2004. ACM.

[115] D. Liebowitz. *Camera Calibration and Reconstruction of Geometry from Images.* PhD thesis, University of Oxford, Dept. Engineering Science, Jun 2001.

[116] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. EuroGraphics*, volume 18, pages 39–50, Sep 1999.

[117] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proc. CVPR*, pages 482–488, Jun 1998.

[118] D. Liebowitz and A. Zisserman. Combining scene and auto-calibration constraints. In *Proc. ICCV*, Sep 1999.

[119] J. Liu and Y. Yang. Multiresolution color image segmentation. *IEEE PAMI*, 16(7):689–700, Jul 1994.

[120] A. M. Loh and R. Hartley. Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. In *British Machine Vision Conference*, pages 69–78, 2005.

[121] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Computer Graphocs*, 21(24):163–169, Jul 1987.

[122] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, 1967.

[123] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Proc. ECCV*, 2008.

[124] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of Applied Mathematics*, 11:431–441, 1963.

[125] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, May 2004.

[126] H. Martinsson, F. Gaspard, A. Bartoli, and J.-M. Lavest. Energy-based reconstruction of 3D curves for quality control. In *Proc. EMMCVPR*, 2007.

[127] J. Mattingley and S. Boyd. *Convex Optimization in Signal Processing and Communications*, chapter Automatic Code Generation for Real-Time Convex Optimization, pages <++>. Cambridge University Press, 2009.

[128] K. McHenry, J. Ponce, and D. A. Forsyth. Finding glass. In *Proc. CVPR*, pages 973–979, 2005.

[129] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley series in probability and statistics. John Wiley & Sons, 1997.

[130] G. J. Mclachlan and D. Peel. Mixfit: An algorithm for automatic fitting and testing of normal mixtures. In *In Proceedings of the 14th International Conference on Pattern Recognition*, pages 553–557. IEEE Computer Society, 1998.

[131] D. Meagher. Geometric modelling using octree encoding. *Computer Graphics and Image Processing*, 19:129–147, 1982.

[132] S. Mika, B. Schölkopf, A. Smola, K. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In *NIPS*, pages 536–542, Cambridge, MA, USA, 1999. MIT Press.

[133] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *Proc. BMVC.*, volume 2, pages 779–788, 2003.

[134] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach-A case study in intensive care monitoring. In *ICML*, 1999.

[135] T. Morita and T. Kanade. A sequential factorization method for recovering shape and motion from image streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:858–867, 1997.

[136] B. S. Morse, W. Liu, T. S. Yoo, and K. Subramanian. Active contours using a constraint-based implicit representation. In *ACM SIGGRAPH COURSES*, page 252, New York, NY, USA, 2005. ACM.

[137] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349 – 384, 1998.

[138] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 42:577–684, 1989.

[139] C. Nastar, B. Moghaddam, and A. Pentland. Generalized image matching: Statistical learning of physically-based deformations. In *Proc. ECCV*, volume 1, pages 589–598, Cambridge, UK, 1996.

[140] R. Nevatia and T.O. Binford. Description and recognition of complex curved objects. *Artificial Intelligence Journal*, 8:77–98, 1977.

[141] Y. Ohta and T. Kanade. Stereo by intra- and inter- scanline search. *IEEE PAMI*, 7(2):139–154, 1985.

[142] M. Okabe, S. Owada, and T. Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. In *Computer Graphics Forum (Proceedings of Eurographics 2005)*, 2005.

[143] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *Proc. CVPR*, 2006.

[144] D. Parikh, C.L. Zitnick, and T. Chen. From appearance to context-based recognition: Dense labeling in small images. In *Proc. CVPR*, pages 1–8, June 2008.

[145] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(7):715–729, Jul 1991.

[146] A. P. Pentland. Automatic extraction of deformable part models. *Int. J. Comput. Vision*, 4(2):107–126, 1990.

[147] Alex Pentland. Parts: Structured descriptions of shape. In *AAAI*, pages 695–701, 1986.

[148] A.P. Pentland. Shape information from shading: A theory about human perception. In *ICCV88*, pages 404–413, 1988.

[149] J. Platt. Probabilities for SV learning. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.

[150] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *Proc. ECCV*, volume 2, pages 97–108, 1994.

[151] J. Ponce. On characterizing ribbons and finding skewed symmetries. In *Proc. Robotics and Automation*, volume 1, pages 49–54, 1989.

[152] B. Potetz and T. S. Lee. Efficient belief propagation for higher-order cliques using linear constraint nodes. *Comput. Vis. Image Underst.*, 112(1):39–54, 2008.

[153] M. Prasad, A. Zisserman, and A. W. Fitzgibbon. Fast and controllable 3D modelling from silhouettes. In *Short Paper Proceedings of the 26th Annual Conference of the European Association for Graphics, Dublin*, pages 9–12, Sep 2005.

[154] M. Prasad, A. Zisserman, and A. W. Fitzgibbon. Single view reconstruction of curved surfaces. In *Proc. CVPR*, volume 2, pages 1345–1354, June 2006.

[155] M. Prasad, A. Zisserman, A. W. Fitzgibbon, M. P. Kumar, and P. H. S. Torr. Learning class-specific edges for object detection and segmentation. In *Proc. Indian Conf. on Computer Vision, Graphics and Image Processing*, Dec 2006.

[156] F. Provost. The case against accuracy estimation for comparing induction algorithms. In *In Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann, 1998.

[157] S. Ramalingam, P. Kohli, K. Alahari, and P.H.S. Torr. Exact inference in multi-label CRFs with higher order cliques. In *Proc. CVPR*, pages 1–8, 2008.

[158] X. Ren, C.C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. volume 2, pages 1214–1221 Vol. 2, Oct. 2005.

[159] S. Romdhani, S. Gong, and A. Psarrou. Multi-view nonlinear active shape model using kernel PCA. In *Proc. BMVC.*, pages 13–16, 1999.

[160] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D morphable model. In *Proc. ICCV*, 2003.

[161] A. Rosenfeld. Axial representations of shape. *Computer Vision, Graphics, and Image Processing*, 33:156–173, 1986.

[162] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proc. CVPR*, volume 2, pages 860–867, 2005.

[163] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *Proc. ACM SIGGRAPH*, 23(3):309–314, 2004.

[164] S. Roy and I.J. Cox. A maximum-flow formulation of the n -camera stereo correspondence problem. In *ICCV 98*, pages 492–499, 1998.

[165] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proc. CVPR*, 2006.

[166] M. Salzmann, F. M. Noguer, V. Lepetit, and P. Fua. Closed-form solution to non-rigid 3D surface registration. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *Proc. ECCV*, volume 5305 of *Lecture Notes in Computer Science*, pages 581–594. Springer, 2008.

[167] M. Salzmann, R. Urtasun, and P. Fua. Local deformation models for monocular 3D shape recovery. In *Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, 2008.

[168] G. Sapiro. Vector (self) snakes: A geometric framework for color, texture and multiscale image segmentation. In *Intl. Conf. Image Proc.*, pages I: 817–820, 1996.

[169] Guillermo Sapiro. Color snakes. *Comput. Vis. Image Underst.*, 68(2):247–253, 1997.

[170] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proc. ECCV*, volume 1, pages 414–431. Springer-Verlag, 2002.

[171] C. Schmid and A. Zisserman. The geometry and matching of curves in multiple views. In *Proc. ECCV*, pages 394–409. Springer-Verlag, Jun 1998.

[172] B. Scholkopf and A. Smola. *Learning with Kernels.* MIT Press, 2002.

[173] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *Proceedings of the British Machine Vision Conference*, 2008.

[174] G. L. Scott. The alternative snake–and other animals. In *Proc. 3rd Alvey Vision Conference*, pages 341–347, Cambridge, 1987.

[175] L. Shafarenko, M. Petrou, and J. Kittler. Automatic watershed segmentation of randomly textured color images. *Image Processing, IEEE Transactions on*, 6(11):1530–1544, Nov 1997.

[176] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, 2000.

[177] I. Shimshoni and J. Ponce. Recovering the shape of polyhedra using line-drawing analysis and complex reflectance models. In *Proc. CVPR*, pages 514–519. IEEE Computer Society, 1994.

[178] J. Shotton, A. Blake, and R. Cipolla. Contour-Based Learning for Object Detection. In *Proc. ICCV*, 2005.

[179] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint Appearance, Shape and context Modeling for Multi-Class Object Recognition and Segmentation. *Proc. ECCV*, 2006.

[180] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Trans. Graph.*, 25(3), 2006.

[181] J. E. Solem and F. Kahl. Surface reconstruction using learned shape models. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17.* MIT Press, Cambridge, MA, 2005.

[182] P. D. Sozou, T. F. Cootes, C. J. Taylor, and E. C. Di Mauro. Non-linear point distribution modelling using a multi-layer perceptron. In *Proc. BMVC.*, pages 107–116. BMVA Press, 1995.

[183] P.D. Sozou, T.F. Cootes, C.J. Taylor, and E.C. di Mauro. Nonlinear generalization of point distribution models using polynomial regression. *Image and Vision Computing*, 13(5):451–457, June 1995.

[184] L.H. Staib and J.S. Duncan. Boundary finding with parametrically deformable models. *IEEE PAMI*, 14(11):1061–1075, November 1992.

[185] P. Sturm and S. J. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. In *Proc. BMVC.*, 1999.

[186] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE PAMI*, 12(6):513–528, 1990.

[187] R. Szeliski. Fast shape from shading. *CVGIP: Image Underst.*, 53(2):129–153, 1991.

[188] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using non-linear least squares. Technical Report CRL 93/3, DEC Cambridge Research Lab, Mar 1993.

[189] C.J. Taylor, D.J. Kriegman, and P. Anandan. Structure and motion in two dimensions from multiple images: a least squares approach. In *Visual Motion, Proceedings of the IEEE Workshop on*, pages 242–248, Oct 1991.

[190] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE PAMI*, 13(7):703–714, 1991.

[191] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models for 3D object reconstruction. *Proc. ICCV*, pages 269–276, 1987.

[192] A. Thayananthan, B. Stenger, P.H.S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. CVPR*, pages I: 127–133, 2003.

[193] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *IJCV*, 9(2):137–154, Nov 1992.

[194] L. Torresani, A. Hertzmann, and C. Bregler. Non-rigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE PAMI*, 30(5):878–892, 2008.

[195] L. Torresani, D. B. Yang, E. J. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *Proc. CVPR*, volume 1, pages 493–500, 2001.

[196] K. Toyama and A. Blake. Probabilistic tracking in a metric space. In *Proc. ICCV*, volume II, pages 50–57, 2001.

[197] W. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

[198] M. Turk and A. P. Pentland. Face recognition using eigenfaces. In *CVPR*, pages 586–591, 1991.

[199] T. Uchiyama and M.A. Arbib. Color image segmentation using competitive learning. *IEEE PAMI*, 16(12):1197–1206, Dec 1994.

[200] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proc. CVPR*, volume 2, pages 691–698, Jun 2003.

[201] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on AI*, pages 55–60, 1999.

[202] D. L. Waltz. Understanding line drawings of scenes with shadows. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, 1975.

[203] R. White and D. A. Forsyth. Combining cues: Shape from shading and texture. In *Proc. CVPR*, pages 1809–1816, 2006.

[204] J. Winn and J. Shotton. The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. In *Proc. CVPR*, 2006.

[205] A. P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1–3):17–45, Aug 1981.

[206] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE PAMI*, 15(11):1101–1113, Nov 1993.

[207] J. Xiao, J. Chai, and T. Kanade. A closed-form solution to nonrigid shape and motion recovery. Technical Report CMU-RI-TR-03-16, Robotics Institute, Pittsburgh, PA, June 2003.

[208] C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. *Proc. CVPR*, 0:66–71, 1997.

[209] A.L. Yuille, D.S. Cohen, and P.W. Hallinan. Feature extraction from faces using deformable templates. *IJCV*, 8(2):99–111, August 1992.

[210] R.C. Zeleznik, K.P. Herndon, and J.F. Hughes. SKETCH: an interface for sketching 3D scenes. In *Proc. ACM SIGGRAPH*, pages 163–170, New York, NY, USA, 1996. ACM Press.

[211] L. Zhang, G. Dugas-Phocion, J.S. Samson, and S.M. Seitz. Single view modeling of free-form scenes. In *Proc. CVPR*, pages I:990–997, 2001.

[212] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 13(2):119–152, 1994.

[213] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo and shape from shading. *IEEE PAMI*, 13(7):680–702, 1991.