Interactive 3D Reconstruction From Video

Benjamin Ward

School of Computer Science

The University of Adelaide

August 2012

Contents

Abstract

This thesis explores several approaches to 3D reconstruction from video sequences in which the reconstruction process is aided by information about the scene provided interactively by a user. This user-supplied information may describe, for example, the types of objects in the scene and their positions, the boundaries of faces of an object in one view, geometric properties of an object, or geometry which is not seen in the video sequence.

By incorporating this information into the modelling process, we can reconstruct scenes which would be difficult or impossible to reconstruct with a fully automated process due to elements of the scene having minimal texture, or translucent or reflective surfaces, or due to significant parts of the scene being occluded or poorly visible in all views. These interactive methods allow the user to reconstruct the desired parts of the scene to the desired level of detail, with automated processing minimising the required interaction.

We first propose a method in which individual objects in a video sequence are reconstructed by fitting pre-defined model types selected by a user. A novel fitting process is used to efficiently evaluate and optimise models sampled from the large space of possible models. Models are evaluated using a novel combination of user-supplied information, 2D image information, and 3D point cloud data recovered with a Structure from Motion process.

A method is also presented for polygonal modelling of objects in video sequences. This method allows the user to define the faces of the model in a single frame through an intuitive sketch-based interface. An automated process generates a 3D model from this set of 2D faces. Interactive techniques are also described for generating a complete model from a partial model of an object, for fitting primitive shapes, and for incorporating geometric constraints into the modelling process.

We demonstrate the use of this polygonal modelling method for rapidly generating models in Augmented Reality environments. We then describe an additional method for Augmented Reality applications in which the camera is used as the input device. In this method interaction with the camera is used first to select an object in a scene, and then to provide sufficient views of the object for a complete reconstruction.

Declaration

Publications

In carrying out the research which contributed to this thesis, a number of papers were published.

- A. van den Hengel, A. Dick, T. Thormählen, B. Ward, P. H. S. Torr. Fitting multiple models to multiple images with minimal user interaction. In *Proceedings of the International Workshop on the Representation and Use of Prior Knowledge in Vision (WRUPKV)*, May 2006

- A. van den Hengel, A. Dick, T. Thormählen, B. Ward, P. H. S. Torr. Hierarchical model fitting to 2D and 3D data. In *Proceedings of the 3rd International Conference on Computer Graphics, Imaging and Visualisation*, July 2006

- A. van den Hengel, A. Dick, T. Thormählen, B. Ward, P. H. S. Torr. Building Models of Regular Scenes from Structure-and-Motion. In *Proceedings of the British conference on Machine vision (BMVC '06)*, September 2006

- A. van den Hengel, A. Dick, T. Thormählen, B. Ward, P. H. S. Torr. Rapid Interactive Modelling from Video with Graph Cuts. In *Proceedings of Eurographics 2006*, September 2006

- A. van den Hengel, A. Dick, T. Thormählen, B. Ward, P. H. S. Torr. VideoTrace: rapid interactive scene modelling from video. *ACM Transactions on Graphics*, 26, July 2007

- A. van den Hengel, A. Dick, T. Thormählen, B. Ward, P. H. S. Torr. Interactive 3D Model Completion. In *Proceedings of Digital Image Computing: Techniques and Applications 2007*, December 2007

- A. van den Hengel, R. Hill, B. Ward, and A. Dick. In situ Image Based Modelling. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) 2009*, October 2009

- J. Bastian, B. Ward, R. Hill, A. van den Hengel, and A. Dick. Interactive Modelling for AR Applications. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) 2010*, October 2010

- B. Ward, S. B. Kang, and E. P. Bennett. Depth Director: A System for Adding Depth to Movies, *IEEE Computer Graphics and Applications*, 31, Jan-Feb 2011

Acknowledgements

First and foremost, I wish to thank my supervisors Anton van den Hengel and Anthony Dick for their excellent guidance throughout my candidature. My thanks as well to my colleagues Rhys Hill, John Bastian, Daniel Pooley, and Thorsten Thormählen for all of their help and advice. I am also extremely grateful to my family and friends for all the support they have given me over the years. I would especially like to thank my partner Kate Hoffmann for her endless patience, and for being a constant source of encouragement and inspiration.

List of Figures

Notation

S	Scalar
$\boldsymbol{M}$	Matrix
$\mathbf{X}$	Vector
$(X_1, \ldots, X_n)$	Elements of vector $\mathbf{X}$
$\mathbf{x}$	2D points and vectors
$\Pr(X)$	Probability of X
$\Pr(X\|Y)$	Conditional probability of X given Y
$\mathcal{L}(X, Y)$	Likelihood function

1 Introduction

Interpreting the 3D properties of a scene depicted in 2D images from the viewpoint of one or more cameras is a key problem in computer vision. This problem is challenging, as accurately reconstructing the 3D position and shape of the elements of a scene requires recovering spatial information lost in projecting from a 3D space to a 2D plane. Where scene reconstruction is possible, the results have numerous applications. 3D reconstructions of real world environments are often required in film production, where they allow synthetic elements to be convincingly integrated into live footage. Other applications include generating content for interactive games and training simulations, and modelling structures for architectural visualisation, amongst many others.

Where multiple views of a scene are available, either provided by multiple photographs taken from different locations, or by video shot while moving around the scene, reconstruction can be treated as a geometric problem. Various approaches to this problem exist, including:

- Feature-based approaches, which recover a sparse representation of the 3D structure of a scene by extracting clearly identified features from the images and matching these features across multiple views, and triangulating 2D feature positions to recover positions in 3D space
- Visual hull-based approaches, which aim to recover a 3D surface consistent with the silhouette of an object as seen from each view
- Photoconsistency-based approaches, which aim to recover a surface consistent with the appearance of an object across all views

Several factors limit the accuracy of reconstructions which can be achieved through these approaches. For parts of a scene where features are sparse, such as plain white walls, multiple images may still not provide enough information for accurate reconstruction, as many possible reconstructions of a surface could result in the same appearance in the corresponding regions of the images. The appearance of specular highlights, reflections, and translucent surfaces, common in real world scenes, can violate the assumptions typically used to make automated reconstruction a tractable problem. The resolution of the images may also not be high enough to reconstruct fine detail in some regions of the scene.

Even without such difficulties, an accurate reconstruction of the visible parts of a scene may still only be a partial reconstruction of the complete scene. In a complex environment, much of the scene may be occluded, with objects close to the camera concealing objects behind. Vantage points needed for complete coverage of the scene are often inaccessible or impractical.

Presented with 2D images of a real-world environment, humans intuitively estimate 3D properties of the scene by inferring information from prior knowledge about elements identified in the images. This intuitive understanding of the content of a scene provides a means of resolving the ambiguities present in image information. This human understanding can assist in the task of 3D reconstruction, leading to a range of reconstruction methods where automated processes are aided by information provided by a user on the structure and content of a scene.

In systems for interactive reconstruction from image sequences, a user typically guides the reconstruction process to model the scene as a collection of primitive shapes such as boxes, cylinders, and planar facets. The user provides constraints on the position and shape of these primitives by specifying positions of vertices or edges over the image sequence, shape properties, and relationships between primitives. The automated components of such systems generate an optimal reconstruction given these user-specified constraints.

Reconstructions generated with user interaction can overcome a number of limitations of reconstructions from purely automated methods, to provide reconstructions which are more complete, accurate, and better suited to their application. User interaction can aid in generating structure in regions which are not visible in the sequence, regions with minimal texture, and regions with an inconsistent appearance. Such regions can either be missing or reconstructed highly inaccurately in the results of automated reconstruction methods. By decomposing a scene as a collection of primitive shapes, reconstructions generated interactively can be more compact, and more flexible when used in other applications, than the dense reconstructions typically generated by automated systems. Interactively specified constraints can aid in generating reconstructions which accurately represent geometric properties of the scene as identified by the user, and can overcome ambiguities and deficiencies in the input image set.

Systems for interactive reconstruction from images commonly require the user to provide sufficient constraints to specify all parameters of the model. This can require significant and time-consuming interaction such as specifying corresponding points over the image set [1], outlining faces of the model in multiple images [2], and marking sets of orthogonal lines [3]. Such systems have often been designed primarily for the reconstruction of architectural scenes, exploiting the geometric properties of such scenes to provide constraints on the model and camera parameters [2] [4] [5]. These systems may not be appropriate for modelling more general scenes. Systems in which models are generated from a limited set of primitive types and systems which rely on determining vanishing geometry can place significant restrictions on the range of scenes which can be accurately reconstructed. For some applications, such as constructing models for Augmented Reality environments, the type of highly involved interaction required by these systems may be infeasible.

This thesis explores interactive approaches to 3D reconstruction from video, presenting several related methods aimed at different reconstruction problems. In each of these methods the recovery of scene structure is informed by information provided through user interaction. In contrast to a number of previous methods for interactive reconstruction from images and video, we focus on developing methods which minimise the interaction required from the user while still allowing the flexibility of modelling required to achieve the task, and we do not limit modelling to a specific domain such as architecture.

To minimise interaction, we divide the information needed for reconstruction into information which can be readily determined with automated methods, and information which can more reliably be provided by a user. The user provides information which can be difficult to recover automatically, such as information on the content of the scene and the geometric relationship between elements of the scene. Guided by the user's high-level understanding of the scene, automated processes employed by these methods optimise the fit between models selected or defined by the user and the images.

Rather than requiring the user to provide sufficient constraints to fully define the model, the user initialises the modelling process by providing partial constraints, and the modelling methods use data which can be automatically extracted from the images to determine the missing information. Subsequently the user can provided further constraints if needed to refine the model to the level of detail and accuracy required.

This allows for reconstruction systems which provide the benefits of interactive modelling, producing models which are complete, accurate, and satisfy the user's requirements, while also minimising the interaction required and the complexity of the modelling process. We focus on interfaces which allow models to be specified rapidly, with as much interaction as possible performed in a single frame of a sequence, and which are intuitive for novice users. The methods presented in this thesis are appropriate for modelling a wide range of objects and environments.

1.1 Contribution

In this thesis, we present several novel methods for 3D reconstruction incorporating human interaction, appropriate for different reconstruction scenarios:

- For the problem of modelling an object in a scene where a pre-defined model of that object exists, or the object can be modelled with a primitive shape, we describe a method for fitting models to an image set from simple user interaction in a single frame. This method uses a hierarchical fitting procedure making use of 3D structure extracted from the images, 2D information from the images themselves, and information supplied by the user.

- For the problem of modelling scenes for which no pre-defined model exists, we present a method for polygonal modelling of objects in image sequences through a sketch-based interface. This method applies the fitting process developed for pre-defined models to newly defined geometry, making use of the user's ability to decompose a scene into a collection of planar facets. In this method the majority of the interaction required to define a model can be performed in a single view.

- As knowledge of the shape of elements of the scene may not be sufficient to provide a complete and accurate reconstruction, we present several techniques allowing the user to incorporate information on geometric constraints in the scene. These constraints can be used, for example, to exploit symmetry in the scene to generate a complete model from a partial model of an object.

- We describe the extension of this system for use in generating models for Augmented Reality environments, where modelling with minimal interaction allows for more rapid reconstruction of the scene than was feasible with previous interactive systems. This reconstruction method relies on significant mouse-based interaction, which may not be feasible for all applications. For the problem of modelling in Augmented Reality scenarios in which the camera is also the input device, we present a real-time interactive reconstruction method in which the camera is first used to select an object in a scene, and then used to provide sufficient views for a complete reconstruction of the object.

1.2 Outline

Chapter 2 provides background information on 3D reconstruction from images, and gives examples of existing approaches to this problem. In Chapter 3, a method is developed for fitting models to image sequences, using information provided by the images, Structure from Motion, and user interaction. In Chapter 4, this method is extended to polygonal modelling based on a user-drawn sketch. Techniques for extending and refining the models with user-defined geometric constraints are presented in Chapter 5. Chapter 6 presents two methods for generating models for AR environments. Chapter 7 presents conclusions and possible directions for future research.

2

Background

Recovering 3D structure from 2D images is a widely studied problem in computer vision. Reconstruction methods recover a range of representations of the 3D structure of a scene, from sparse point clouds to dense volumetric models. In this chapter, we provide an overview and examples of some of the more commonly used approaches to reconstruction, rather than an exhaustive survey. In particular, we focus on approaches which relate directly to the reconstruction methods which will be described in later chapters. We begin by describing the geometric basis for 3D reconstruction from images, and some automated approaches to 3D reconstruction. We then introduce a range of methods incorporating user interaction in the reconstruction process, and it is this class of methods that are explored in this thesis. Finally, we describe some of the applications for which 3D reconstructions such as those generated by the methods described in this thesis can be used.

2.1 Automated reconstruction methods

Where it is not required to determine semantic information about the content of the images, reconstruction can be treated as a problem of projective geometry. Feature-based approaches can be used to recover both the parameters of the cameras which captured the images, and a reconstruction of the scene as a sparse set of points. Reconstructed camera parameters are commonly used as a basis for a more detailed automated reconstruction of a scene. Visual hull methods can reconstruct the shape of a convex object from its silhouette in multiple images. Photoconsistency based methods densely

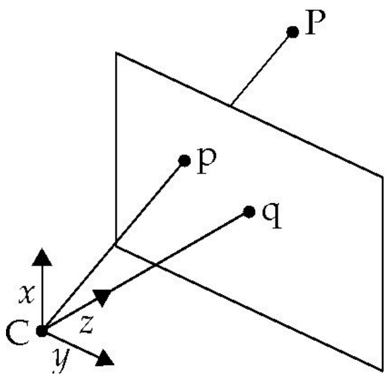


Figure 2.1: Projective camera geometry

reconstruct an object based on the appearance of points on its surface. Reconstruction can often be aided by incorporating prior knowledge about the content of the scene in the form of models.

2.1.1 Feature-based reconstruction

The creation of an image of a scene by a camera is typically modelled by perspective projection. This corresponds to imaging by a pinhole camera. Despite its simplicity, the pinhole camera model (described in detail in [6]) is sufficiently accurate for most cameras used in practice, once the images have been corrected for any lens distortion. For a point in 3D space $\mathbf{P}$ (represented in homogeneous coordinates as a 4 vector), perspective projection gives the corresponding 2D point in an image as $\mathbf{p} = \boldsymbol{A}\mathbf{P}$ where $\boldsymbol{A}$ is the 3x4 projection matrix for the camera. The projection matrix incorporates the camera's orientation, the position of its optical centre in world space, and its intrinsic properties, such as focal length and principal point. This projection process is illustrated in Figure 2.1. $\mathbf{C}$ is the camera's optical centre. $\mathbf{q}$ is the camera's principal point, the point where the image plane is intersected by a line perpendicular to the image plane, and passing through the optical centre. See [6] for more details.

2.1.1.1 Reconstruction from two views

Given two images of the same 3D point, taken by cameras with known parameters in different positions, the 3D location of that point can be recovered by triangulation. This process is illustrated in figure Figure 2.2. The image point $\mathbf{p}$ backprojects to a ray

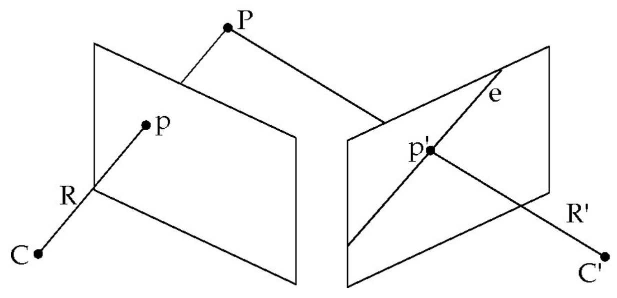


Figure 2.2: Stereo camera geometry

R from **C**, passing through that point. The scene point **P** is found at the intersection of **R** and **R**$'$, the corresponding ray for the second image.

This process is fundamental to feature-based reconstruction techniques, which rely on locating corresponding points in multiple views, and reconstructing their 3D positions. Feature points are points which can be easily distinguished from their surrounding image points, typically due to intensity changes in multiple directions, and so are well-suited for identification in multiple images. These points are located by a feature detector, such as the Harris corner detector [7] for example.

For a feature point $\mathbf{x}$ in one image, the location of the corresponding point $\mathbf{x}'$ in a second image is related by the *epipolar constraint.* The corresponding point in the second image will lie on the epipolar line (labelled **e** in Figure 2.2), the projection of the ray from **C** through $\mathbf{x}$ into the image plane of the second camera.

The epipolar geometry of two views can be represented by the *fundamental matrix.* For the 3×3 fundamental matrix $\boldsymbol{F}$, any pair of points in two images corresponding to the same 3D points satisfy the constraint

$$\mathbf{x}'\boldsymbol{F}\mathbf{x} = 0$$

The fundamental matrix can be determined linearly for 7 or more pairs of matching points. As the point matches between two images are typically expected to include outliers, robust algorithms in the style of RANSAC (Random Sample Consensus) [8] are commonly used in computing the fundamental matrix. In such algorithms, minimal sets of points matches are repeatedly sampled and used to compute a hypothesised fundamental matrix. Each sample is evaluated by determining the percentage of inlier features from the complete set of matches given the hypothesised matrix. Non-linear minimisation is typically applied to refine the matrix computed from point matches. To aid in refining the selected fundamental matrix, the epipolar constraint can be used to find additional point matches.

From the fundamental matrix, the parameters of the camera pair can be determined up to a projective ambiguity. The relative position and orientation of the cameras can be determined from the fundamental matrix if the internal parameters of the cameras are known. If these parameters are unknown, self-calibration methods can be used to recover Euclidean structure from a projective reconstruction. Detailed descriptions of methods of 3D reconstruction from two or more views are provided in [9].

2.1.1.2 Structure from motion

While a minimum of two views of an object are required for a feature-based reconstruction, a moving object or a camera moving through a scene may provide images for a large number of closely spaced views. Feature-based reconstruction applied to a sequence of images of a moving object over time, typically acquired by a video camera, is termed Structure from Motion (SfM). A description of a complete reconstruction system using SfM applied to video from a hand-held camera is provided in [10]. An SfM technique is used to estimate the camera and point cloud data which form the basis for the reconstruction methods described in subsequent chapters. Compared with reconstruction for a smaller set of widely spaced photographs, reconstructing from a video sequence reduces the difficulty of finding corresponding features between frames, as the motion between neighbouring frames is relatively small at standard video frame rates, and the large number of frames can result in a more reliable reconstruction. A feature point tracker, such as the KLT Feature Tracker [11], is applied to find correspondences between feature points in neighbouring images, tracking the motion of each point over the sequence.

In the SfM system described in [10], the reconstruction process for all views is initialised by first performing a two view reconstruction. Features in the remaining views can then be matched against the recovered structure, and used to recover camera pose for additional views. Matches in the newly computed views are then used to reconstruct additional 3D points, which can be matched against further views, with the process iterating until parameters have been recovered for all cameras. The complete set of recovered camera parameters and 3D points can then be refined. Sets of camera and point parameters can be jointly optimised with a *bundle adjustment* [9] method, minimizing the error measure

$$\sum_i \sum_j d(\mathbf{x}_{ij}, \boldsymbol{A}_i \mathbf{X}_j)^2$$

giving the sum of squared distances between each reconstructed 3D point $\mathbf{X}_j$ and the corresponding feature point $\mathbf{x}_{ij}$ in each frame i, where $\boldsymbol{A}_i$ is the projection matrix for that frame.

SfM methods will not be able to recover camera parameters for all video sequences. Typical SfM methods will fail if there is minimal camera translation over the sequence, if the scene consists of a single plane, and if significant parts of the scene are non-static. To determine the camera parameters, feature-based SfM approaches require a scene with a sufficient number of points which can be reliably detected and tracked over the sequence.

The density of the 3D points reconstructed by SfM will vary with the amount of texture present in different regions of the scene. Feature detectors can only localise clearly defined features such as object corners. As such, 3D features will not be found for areas of uniform colour and shading (white walls, for example), or along edges where the image intensity only varies in one direction. Recovering 3D points requires identifying the same point in multiple frames. This may not be possible if the surface containing a point does not have a consistent appearance from different perspectives, as will be the case for translucent and highly reflective surfaces. Recovering 3D points may also not be possible in regions with highly repetitive texture, giving a large number of possible matches for any point.

Errors in the estimated 2D positions of matched features, which are typically modelled as being corrupted by zero-mean Gaussian noise, will result in errors in the recovered camera parameters and 3D points. Errors will be particularly pronounced for points far from the camera, where small differences in the 2D image position will correspond to large differences in the estimated 3D position. This error will also dominate the estimate of the 3D point location when the baseline between the cameras viewing a point is small.

The sparse structure recovered by SfM processes will be insufficient for many applications which require a more complete reconstruction of the scene. SfM is commonly applied as a first stage of a 3D reconstruction process. The recovered camera parameters provide a basis for more detailed reconstruction techniques, which may also be informed by the sparse 3D structure provided by the feature points. Approaches to more detailed reconstruction generally recover either a reconstruction of the surface of the object, represented by a polyhedral mesh or parametric surface, or a volumetric reconstruction, representing the space enclosed by the object by the occupancy of 'voxels' (analogous to 2D pixels) in a 3D grid. The following sections review a range of such approaches.

2.1.2 Visual hull methods

The back projection of an image point gives a ray from the camera's optical centre on which the corresponding 3D point will be found. Likewise, back projecting the silhouette of an object in an image gives the bounds of a 3D region which contains that object. Given a corresponding back projected silhouette from a second view of the object, the object will be found within the intersection of the two back projections. Further silhouettes increase the constraints, and can be used to construct a volume, bounded by the intersection of all back projected silhouettes, termed a visual hull [12].

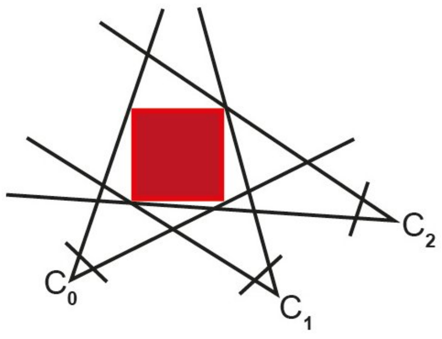
Figure 2.3 illustrates the process of recovering the visual hull. 2.3a shows the back projection of the silhouette of a cube as viewed in three cameras. 2.3b shows the visual hull resulting from intersecting the back projection from each camera. A silhouette carving visual hull method is used to recover models for Augmented Reality applications in the reconstruction method that we describe in Chapter 6.

Silhouette carving methods recover the visual hull, represented as a voxel grid, from a set of silhouettes by removing from the volume all voxels projecting to points outside of the silhouette in one or more views. The carving process assigns each voxel in the grid $\{v^i\}$ a binary label indicating whether it is part of the object. A voxel is labelled as being part of the object if, for a set of n camera matrics $\{\boldsymbol{A}_j\}$,

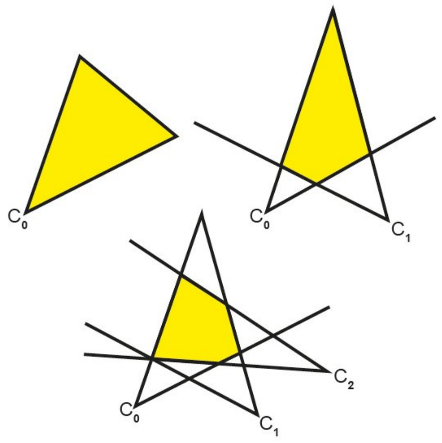
$$\boldsymbol{A}_j v^i \cap \beta_j = \emptyset \, , \, \forall j \in [1, n]$$

where β_j is the set of pixels outside the silhouette in frame j.

An example is given in [13], where an octree representation of the volume is used for efficient carving. To reduce the cost of determining whether each voxel is inside a silhouette, this testing is performed as a coarse-to-fine process. A bounding box



(a)



(b)

Figure 2.3: Reconstructing the visual hull. a: An overhead view of three cameras viewing a cube, showing the back projection of the object silhouette for each frame. b: The visual hull resulting from intersecting each back projected silhouette

containing a set of voxels is first tested against the silhouette. If the box is entirely inside or outside of the silhouette, no further testing is needed to label the voxels. Otherwise, the box is subdivided, and testing continues for the subdivided boxes.

Volumetric methods recover a quantized representation of the visual hull. This can lead to distinct visual artifacts when rendering the resulting model from views where the size in the image of individual voxels if larger than the size of a pixel. Such artifacts can be avoided by methods which perform the reconstruction in image space rather than in 3D space. These methods may be suitable when the goal of reconstruction is to generate novel views of an object, rather than to recover a 3D model of the object. In [14], for example, the projection of the visual hull is determined independantly for each novel view. For each pixel in a novel view, the intersection of the visual hull and a ray back projected through that pixel is computed and used to determine the pixel colour from the projection of the intersection point nearest to the camera into the closest reference view where that point is visible.

The method of [15] allows for visual hull reconstruction and rendering in real time with processing performed on the GPU. The reconstructed model can be used to render novel views of the object with projective texturing. Texture is generated by smoothly blending between the appearance of the object in the source images, weighted by the deviation of the viewing direction for each view from the viewing direction for the novel view.

An example of a method with a similar basis to visual hull techniques, but without requiring explicit computation of silhouettes, is given in [16]. In this method, a volumetric reconstruction is computed from a set of images from different views, using a graph cut to segment voxels belonging to the object from the background. The cost of labelling any voxel as belonging to the object is determined from the difference between the values of the corresponding pixels in the image set and the known background for each image. A penalty on different labels for adjacent voxels is used to encourage spatial smoothness.

While visual hull methods can provide a complete shape enclosing the object, these methods will not reconstruct surface concavities. Within a concave region, any 3D point will project to a 2D point within the object silhouette as seen from any view. Such points will not be removed by a carving algorithm. The accuracy of the reconstruction is also dependent on the range of views available. A complete reconstruction of the visual hull can require views of the object from an exhaustive range of angles. Obtaining sufficient views of an object for an accurate visual hull reconstruction will often be infeasible in unconstrained real-world environments. In addition, visual hull methods require extracting an accurate silhouette of the object for each view. This is itself a challenging problem in unconstrained environments without known backgrounds. The accuracy of the reconstruction will also be dependant on the accuracy of the recovered camera parameters.

2.1.3 Shape from photoconsistency

SfM techniques rely on the similar appearance of corresponding points in multiple views to recover the sparse 3D structure of a scene. Photoconsistency-based approaches to reconstruction apply this consistency constraint to build a dense reconstruction of the scene, recovering a surface which is consistent with each image when projected into that image. We make use of photoconsistency for precise model fitting in Chapters 3, 4, and 5. For volumetric reconstruction, photoconsistency is applied by methods such as voxel colouring [17]. This method computes a volumetric reconstruction in which the colour of each voxel is consistent with the image set. To handle occlusions in the scene, voxels are traversed in a fixed order allowing occlusion for each voxel to be determined from the set of voxels already included in the reconstruction.

Similarly, the space carving [18] method extends the concept of a visual hull, recovering a photo hull. The photo hull is defined as the shape containing all shapes which are photoconsistent with the images, and is computed by iteratively removing portions of a voxel space where the appearance of the voxels is inconsistent over multiple views. To measure consistency, this method evaluates the variance of the colour at the projection of each voxel over the image set. A voxel v^i is evaluated as photoconsistent if

$$\mathrm{var}\{F_j(\boldsymbol{A}_j v_i)\} < \tau$$

where $\{\boldsymbol{A}_j\}$ is the set of projection matrices for a set of frames where the voxel is visible, $F_j(\mathbf{a})$ gives the pixel colour for point $\mathbf{a}$ in frame j, and τ is a threshold chosen to give tolerance to small errors due to quantization, sensor noise, and other effects.

In the method of [19], a volumetric reconstruction is performed with a graph cut on a volume defined within a base surface. This base surface can be generated either through silhouette carving or by triangulating a set of reconstructed points. The base surface allows for a graph structure in which the voxels on the exterior and interior of the volume are connected to the source and sink of the graph respectively. The graph cut finds an optimal separation between points inside and outside the object on the basis of photoconsistency. The base surface is used for occlusion reasoning in the photoconsistency computation.

Photoconsistency is evaluated in this method with a measure based on Normalised Cross-Correlation (NCC) [20], comparing patches around the projection of the point into each image. NCC between two image patches $F_i(\mathbf{a}, x, y)$, giving colour in frame i at an offset determined by (x, y) from an image point $\mathbf{a}$, and $F_j(\mathbf{b}, x, y)$ is evaluated as

$$\sum_{x,y} \frac{(F_i(\mathbf{a}, x, y) - \bar{F}_i(\mathbf{a}, x, y))(F_j(\mathbf{b}, x, y) - \bar{F}_j(\mathbf{b}, x, y)))}{\sigma_{F_i(\mathbf{a},x,y)} \sigma_{F_j(\mathbf{b},x,y)}},$$

where $\bar{F}$ and σ_F are mean and standard deviation over the patch. Considering the appearance of the neighbourhoods around a pair of pixels gives stronger cues as to whether those pixels correspond to the same point than comparing those pixels in isolation. By normalising the patches, this measure provides some tolerance for the effect of variation in lighting of the scene and exposure settings of the cameras on the appearance of the image regions.

A complete reconstruction process is described in [10], beginning with a set of uncalibrated images, and performing SfM, dense surface estimation using photoconsistency, and model construction and texture mapping. An example of a reconstruction algorithm designed for a particular application is [21], which presents a method for real-time reconstruction of urban environments viewed from street level. The method uses a plane-sweeping stereo algorithm to recover a depth map. Plane-sweeping methods recover depth by evaluating the photoconsistency of patches aligned with a set of planes swept through a volume of space. This method uses the assumption that points in an urban scene tend to lie on planes with a limited set of orientations. Each pixel in a reference image is tested for a set of hypothesised planes to determine for which plane that pixel is most consistent with the image set. To allow for real-time performance, consistency tests are performed on the GPU, and plane sweeping is restricted to a set of likely planes determined from the sparse features returned by SfM.

For common applications requiring 3D models, a polygonal mesh representation of an object is often more desirable than a volumetric representation. Mesh representations can be stored compactly, and rendered extremely efficiently on modern graphics hardware. Triangle mesh model representations are used by the reconstruction methods described in Chapters 3, 4, and 5. A mesh representation can be generated from a representation of the object surface as a set of points by surface reconstruction methods such as [22]. Some reconstruction methods produce a mesh representation of the object directly. In the method of [23] a visual hull is computed and used to generate a polygonal mesh, which is then deformed to optimise a photoconsistency metric.

Unlike visual hull methods, photoconsistency-based methods are capable of reconstructing concave surfaces, and can reconstruct a surface from a smaller range of views. In suitable conditions, such methods can produce highly detailed and accurate reconstructions. However, an object must satisfy a number of assumptions for a complete and accurate reconstruction to be possible using photoconsistency. The appearance of a point on the object surface in a given view is dependant on the lighting of the scene, the reflectance properties of the surface, and properties of the camera. While using measures such as NCC for photoconsistency can provide tolerance to lighting and exposure variations, accurately matching points between images is still challenging for surfaces which are translucent or reflective. It can also be difficult to determine the true surface in regions with consistent colour and texture, where photoconsistency constraints may be satisfied by many possible surfaces. The resolution of the images will limit the level of fine detail which can be reconstructed. A complete reconstruction using photoconsistency also requires unoccluded views of each object surface, which is often infeasible in real-world environments.

2.1.4 Model based reconstruction

The techniques described above provide various means for recovering 3D properties of a scene purely from optical information provided by images of the scene. While these techniques can provide highly accurate reconstructions in suitable situations, real-world applications often present restrictions on the available views of a scene, lighting variation, image noise, challenging surfaces and other factors which may make complete reconstructions based solely on optical information infeasible. However, a reconstruction suitable for a particular application may still be possible if information about the content of the scene can be provided by a user or can be automatically recovered. Prior information can be incorporated into the reconstruction process through the use of 3D models capable of representing elements of the scene. These models could fully specify the structure of part of the scene if a predefined 3D model exists for that element, or be a parameterised model capable of representing instances of a particular object type. More generic models may represent primitive shapes that are expected to be found in the scene. The following section describes some approaches employing models for automated reconstruction.

2.1.4.1 Generic primitives

Generic primitives provide a high-level approach to modelling. We use generic primitive types for modelling objects in Chapters 3, 4 and 5. In [24] objects such as the roofs of houses from aerial imagery are reconstructed as sets of planar faces. A set of 3D lines is reconstructed by matching edges over the image set. From this set, hypothesised faces are generated and evaluated for the images. In contrast to polygonal representations which may provide a partial description of the surface of an object, but not necessarily a complete boundary for that object, 3D shape primitives can describe a complete 3D volume matching the object. In the object recognition system of [25] objects in single images are modelled by a set of ten such primitives including cylinders, ellipsoids, and blocks. Combinations of generic primitives can provide accurate representations of a wide range of objects, and an approach to reconstruction where the possible domain of applications is not excessively restricted. However, without applying any knowledge of the object other than how its shape appears in the image, decomposing objects in images into primitives is a highly challenging problem.

In the method of [26], scenes are reconstructed as a combination of triangle meshes and 3D primitive types. Segmentation is applied to an initial sparse mesh, and a Jump-Diffusion based method is used to reconstruct the segmented surfaces as either dense mesh patches or basic geometric types. By representing parts of the scene with primitive types, a significantly more compact model can be generated than would be possible with each element being represented by a dense mesh.

A method for automated 3D reconstruction from a single image is presented in [27]. This method assumes that the scene can be modelled by a ground plane and a set of vertical planes. In this method, the image is clustered into superpixels, and a decision tree algorithm is applied to classify superpixels as belonging to the ground, the sky, or a vertical surface. Lines are fitted to the intersection of the superpixels labelled as ground and the superpixels labelled as vertical surfaces. A 3D model is then constructed consisting of a ground plane and a set of vertical planar surfaces.

2.1.4.2 Predefined models

In contrast to generic primitives, which can require little knowledge of the structure of the modelled object, are pre-defined models such as CAD models providing an exact description of the structure of a known 3D object. In Chapter 3 we present an interactive method which can be used for fitting such a pre-defined model. A technique for fitting between models represented as sets of corresponding 3D points is described in [28]. A system for matching models acquired with a range scanner to pre-existing CAD models is described in [29]. The matching is based on similarity between the model surface and clustered surface patches in the range data. The standard algorithm for aligning two 3D mesh or point cloud models is Iterative Closest Points. This algorithm optimises the alignment of models from an initial estimate by iteratively selecting a set of point correspondences on the two models, and optimising an error metric based on the distance between the corresponding points. In [30], a range of variants on the stages of the algorithm are described and evaluated. The detection and fitting of 3D models to point data can be aided by representations of the model which are invariant to rotation [31] [32] [33].

The image based method of [34] tracks the pose of an object using edge locations in one or more views, by matching these locations with the visible edges of a 3D model of the object. Edge features have the advantage of being relatively robust to lighting effects, and can be used for matching and tracking objects with minimal surface features, such as the mechanical parts tracked in this paper. This is possible for an object which is sufficiently complex that its position and orientation can be determined from its visible boundary. The object is tracked over time by adjusting the pose of the model to minimise the distance between points along the model edges and edge points in the current image. Such a technique is suitable for complex structures with a fixed shape, such as manufactured parts. However, shape parameters of a CAD model are fixed, and as such can be suitable for recognition and tracking of known objects, but not more general reconstruction.

2.1.4.3 Parameterised models

A middle ground between generic shapes and rigidly detailed models, parameterised models are designed for a specific class of object, but allow for the possible variation between objects in that class. In addition to the pose and scale parameters used to map pre-defined models into a scene, further parameters govern object shape and appearance. 3D Morphable Models [35], for example, apply a parameterised model to the reconstruction of human faces. From a data set of faces providing 3D shape and appearance information, an average face and its main modes of variation are determined. Presented with a new face from outside the initial data set, model parameters are optimised to minimise the difference between this new face and its reconstruction by the morphable model.

In [36] a parameterised car model is developed which can be aligned with an image using edge matching. Principal Component Analysis is performed on a training data set, and used to determine a parameterisation of the model which captures the main variation between training examples while having a small enough number of parameters to make the alignment process tractable. In the method of [37] a RANSAC framework is used to fit a deformable car model to single images. An appearance model is constructed for a set of landmark features, trained on a large set of labelled images. This method is robust to outliers and capable of fitting the model from a partial set of matches. In [38] a parameterised model, described by a scene graph, is fitted to a set of images using photoconsistency, with photoconsistency costs computed on a graphics card. The method of [39] gives an example of automated interpretation of images combined with 3D reconstruction for the architectural domain. Building models, incorporating components drawn from a library of pre-defined elements, are sampled from a prior distribution and matched against a set of images. Initially, parameters are determined for a set of base planes modelling the walls of a building. Parameters are then determined for a set of primitive types representing, for example, doors, windows, or columns, aligned with these base planes. To find the parameters of the primitive set which best models the images, sets of primitives are sampled and evaluated based on the image set and prior information on properties of architecture and the appearance of each primitive type.

2.2 Interactive reconstruction methods

Performing reconstruction automatically presents a range of challenges which may be overcome by incorporating user interaction into a system. User interaction may assist in both the problem of recovering camera parameters for a scene, and in reconstructing the scene given the recovered cameras. In scenarios where only one view of a scene is available, the user may be able to indicate geometric constraints in the scene which provide sufficient information to enable reconstruction. Where matching between views is difficult due to widely separated cameras, the user may be able to manually indicate matching points in the images. The user may be able to provide additional information on structure in regions where surface properties of objects in the scene present problems for automated reconstruction methods. The user may also be able to provide information on structure in parts of the scene which are not visible in the sequence. The reconstruction methods described in subsequent chapters are designed to be guided by user interaction.

2.2.1 Image-based modelling

In [40] techniques are demonstrated for making 3D measurements from 2D images using vanishing point and line constraints specified by the user, and it is shown that these techniques can be applied for 3D modelling from a single image. The method of [41] creates reconstructions from a single view, requiring significant labelling by the user. The user specifies a model as a set of planar faces, and indicates constraints such as parallel and perpendicular lines. These constraints are used to calibrate the camera, reconstruct the vanishing geometry of the scene, and reconstruct a set of points and planes. Visual information is not used in the fitting process.

A number of systems exist designed for interactive reconstruction of architectural scenes, from both single and multiple images. Architectural scenes typically feature structural elements such as groups of parallel or orthogonal edges, which can be readily identified by a user and provide valuable information for a 3D reconstruction process.

Facade [2] is a system for interactive reconstruction of architectural scenes from multiple images. In this system, the user models the geometry of the scene with a set of polyhedral primitive objects such as boxes and wedges. The user specifies the position of edges of the primitive objects in multiple views. The user can also specify relationships between primitive objects, such as that one object sits on top of another, or that two objects have the same orientation. Once the user has specified at least as many constraints on edge position as there are parameters required to specify the cameras and the model, these parameters can be optimised. The optimisation minimises the distance between the edges of the model and the edges specified by the user. To obtain an initial estimate of the camera parameters, camera rotation is first estimated using lines with known orientation, then camera positions and model parameters are estimated for these rotations. Minimisation is then applied to the complete parameter set. To add detail to the model, depth maps can be computed for model surfaces by minimising the disparity between the appearance of a surface in one view and a projection into that view of the appearance of the surface in another view.

In the PhotoBuilder system [3], camera parameters for an image sequence are recovered from manually-specified lines and point matches. To calibrate the cameras, the user first identifies sets of parallel lines in an image corresponding to orthogonal directions in the world. Vanishing points are determined from these sets of lines, providing constraints which can be used to recover the camera's intrinsic parameters. Projection matrices are then recovered by performing bundle adjustment on a set of feature correspondences provided by the user. The system then automatically constructs a wireframe model from sets of 3D points. Candidate triangles for the model are tested and selected by comparing the appearance of the projection of the triangle into different views. An initial model can be used to improve the reconstruction with additional feature matches. Additional matches can be added automatically using the model to limit the search space for matched features by providing an estimate of the location of a feature in other frames.

An interactive reconstruction technique in which the user draws polygonal geometry in multiple images is presented in [42]. The user can define correspondences between 2D primitives in different images and geometric constraints such as perpendicularity and coplanarity. This set of constraints is used to recover the cameras and 3D coordinates of the model points. The work in [43] presents a method for camera calibration and 3D reconstruction from one or more panoramas based on users defining geometry such as points, planes, and lines with a known direction in the images. These are used to construct a linear system including both hard and soft constraints. In [44] parallelepipeds are used as primitives for both camera calibration and 3D reconstruction, also requiring the user to mark the vertices of primitives in multiple images.

ICARUS [1] combines automated recovery of camera parameters with an interactive reconstruction technique again based on a user specifying vertex locations of primitives in multiple images. An SfM technique is used to recover camera parameters for a video sequence. The user then specifies constraints on the vertices of primitive shapes such as polygons, boxes, and cylinders. The layout of the various primitives is described in a scene graph. The user can also apply hierarchical constraints to primitives, restricting their parameters by the parameters of their parents. An optimisation procedure is applied to the model parameters using the specified constraints. While hierarchical constraints are applied exactly in optimisation, vertex location constraints are not. The optimisation can be performed in real-time as the user adjusts the constraints.

The system described in [4] is designed for interactive modelling of architecure from a large set of unordered photos. Camera parameters for the set of photos are recovered automatically. In this system, plane fitting is performed on planar faces drawn in the images, without requiring marking in multiple views. The fitting process uses both 3D features and vanishing point constraints. Snapping of edges to vanishing lines is used to assist the user in drawing the model geometry. The texture mapping process uses a graph cut technique to combine texture from multiple images. Specifically designed for reconstructing building facades, [45] presents a method in which a single rectified image of a building facade is subdivided into tiles using detection of repeated elements. The 2D shape of these tiles is matched against a library of 3D models. User interaction is then used to provide depth information. A semi-automated method for facade reconstruction using multiple images is presented in [46]. An automated reconstruction process generates results which can be corrected with interactive refinement.

In the interactive reconstruction method presented in [47] parameterised surfaces are fitted to a set of seed points, manually specified in multiple images. Components of a model can be replicated in other parts of the scene by specifying sufficient corresponding points to define a transform mapping the component to the new location. In the method presented in [48], a user manually labels corresponding regions in a set of calibrated images. A carving method is used to recover a polygonal model which is consistent with these labels. Applying different labels to separate objects allows the system to resolve occlusions.

An example of image-based modelling for another specific domain, [49] describes a semi-automated method for reconstructing plants from image sequences. The user manually assists in segmenting leaves of a plant with a graph cut technique. One leaf is selected as an exemplar and modelled. This model is then fitted to each segmented leaf using 3D points from an SfM process and the segmentation boundaries. An interactive process can then be used to reconstruct branches of the plant with reference to the images.

In [50] a method is presented for interactive reconstruction from an image set based on painting interactions performed in single views. Dense depth is recovered for mesh patches with vertices corresponding to the marked pixels. The reconstruction is performed as the user paints on the image using a hierarchical fitting procedure. From an initial guess of the depth of each vertex, depth values are iteratively refined to optimise photoconsistency. This refinement is first performed on an initial coarse mesh, with the results used to initialise depths for progressively more detailed meshes.

In [51] a range of techniques are presented for 3D reconstruction and editing from a single image. Reconstruction is based on a user manually segmenting elements of an image and assigning depths. Depths can be assigned through painting interactions, or through fitting planes and other primitive object types. Image editing operations such as copying and pasting can then be applied, aided by the assigned depth information.

In the approach of [52] a set of orthographic images of an object are generated from an image sequence. These images can then be used as a basis for modelling in a traditional 3D modelling package. If sufficient views of the object are available, pixel colour in the ortho images can be determined directly from the image set by finding for each pixel in an ortho image the view where a ray from the camera centre to that pixel is closest to orthogonal. If insufficient images exist for this method, a volumetric graph cut is used to reconstruct the visible object surface in each ortho view.

In the system presented in [53], also developed by the author of this thesis, interactive techniques are used to assign depth to frames of a video sequence for the purpose of converting 2D footage for stereo viewing. In this system, a graph cut technique is used to segment elements of a video sequence. The segmentation is initialised with coarse user markup. The user can then manually assign depths to selected elements in a 3D view of the scene. A further graph cut technique is used to assign depth to all pixels in each frame given a partial set of manually-assigned depths. Primitive shape types can be fitted to assign depths to elements of the scene with a known geometric structure. For scenes which are suitable for SfM reconstruction, depth in each frame can be initialised from the 3D structure generated by SfM processing.

2.2.2 Sketch-based modelling

Constructing models with a CAD package or other traditional 3D modelling program such as Autodesk 3ds Max [54] is typically a complex process, and unintuitive for a novice user. To enable faster and more intuitive modelling based on familiar drawing techniques, a range of modelling techniques have been developed which use sketch-based interfaces. The sketch-based modelling system described in [55] is based on defining primitive shapes with gestural interactions. The Teddy system [56] allows the user to construct a model from a set of freehand strokes. 3D polygonal surfaces are generated from 2D shapes by an inflation algorithm. The system presented in [57] interprets freehand contours, including inferring hidden contours from the set of visible contours, and generates plausible smooth 3D shapes matching the drawn outline. In [58] models are likewise constructed from 2D sketches in a single view. Users sketch primitives which can be modified with various handles, and with annotations specifying relationships between components.

Google SketchUp [59] provides a sketch-based interface for architectural modelling. Models may be constructed over a series of photographs used as a visual reference. The method of [60] provides sketch-based modelling assisted by a single image. The image is used as a guide for the construction of the sketch, with sketch strokes being snapped to edges in the image. A 3D model is constructed from the sketch either with an inflation or extrusion operation. The image is also used to texture the final model. This method assumes that the sketched object is symmetric about a plane parallel to the image plane. Sketching is used in the search engine described in [61] as a way of retrieving 3D models from a database. Rotation invariant descriptors are generated for a set of 2D views of each object, and used to match against one or more sketched views provided by a user. A sketch-based interface for specifying a model is used in the modelling method described in Chapter 4.

Figure 2.4: A 3D model in Google Earth

2.3 Applications of 3D reconstruction

3D models of real-world objects are used in a wide variety of applications. Image-based modelling can be useful for increasing the speed with which such models can be generated, and increasing the accuracy of the final models. This is desirable in applications such as architectural modelling. In these applications, the resulting model is often intended to be used separately to the images or video from which it was generated, while still using the images or video to construct a realistic model, and often using these as a source for realistic texture mapping. In other applications, models will be useful specifically because they provide 3D information on a particular element of a video sequence. This will frequently be the case in modelling for visual effects applications, where 3D modelling can assist in realistically incorporating rendered elements into live-action footage. Examples of these application types are given below.

2.3.1 Uses for architecture

A common application of image-based modelling is the creation of models of architectural structures. Such models are seen in applications such as Google Earth [70], which can display user-generated architectural models overlayed on top of aerial photographs of the corresponding location. These models vary in detail, but can typically be classified as low-polygon models. The texture of these models is often drawn from photographs of the buildings themselves. An example of a texture mapping algorithm is given in [71], which describes an algorithm incorporating reduction of visible distortion at triangle boundaries and synthesis of texture for triangles which are not visible in any image. The high number of buildings in any urban centre makes techniques which

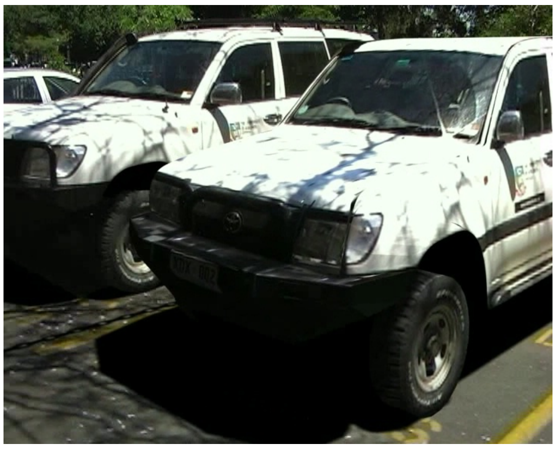


Figure 2.5: Frame of a video containing a synthetic copy of an object alongside the real object

allow rapid generation of realistic models highly desirable. Figure 2.4 shows an example of a 3D model, generated with techniques developed in this thesis, incorporated into Google Earth.

Architectural models can also be utilised in Augmented Reality applications, which can overlay information about the user's environment onto live video of that environment. Other applications include architectural visualisation, where models of proposed structures must be shown side by side with models of existing structures. Architectural models can also be used in simulation applications, for example in simulations of potentially dangerous environments, such as mining sites, where these models can be used in training applications before exposing workers to the real environment.

2.3.2 Uses for visual effects

Structure from Motion is commonly used in films and advertising featuring computer-generated visual effects. Recovering camera parameters for a sequence makes it possible to render synthetic elements over the frames of that sequence, with the perspective from which a synthetic element is rendered correctly matching the perspective of the camera for each frame.

However, this alone is often not sufficient to convincingly integrate a synthetic element into a sequence. If the synthetic object is positioned behind foreground objects, those objects will also need to be modelled, to a sufficient degree that the silhouettes of those models can be used as masks where the synthetic object will not be rendered. This is commonly achieved using rotoscoping techniques. For example, [72] describes a method in which a user specifies parameters for a curve around an object in multiple frames of an image sequence. An optimisation process is then applied to interpolate the curve in the intervening frames. This process involves minimising an energy function which encourages the curve to follow the motion of contours and features in the image, while penalising rapid changes in the curve.

In addition to extracting silhouettes, modelling of background geometry may also be required, allowing the synthetic object to cast shadows, and enabling other visual effects which help to create a convincing combination of real and synthetic elements.

Modelling of background geometry can also allow for convincing interactions between real and synthetic elements. Physical simulation can be used to model convincing collisions between fixed models of real geometry and moving models of synthetic geometry. As such, modelling systems which enable the rapid reconstruction of scene geometry are of significant benefit in the creation of visual effects.

As well as the addition of synthetic objects, another common task in visual effects is the removal of existing objects. Objects required for film production, but not desired in the final frames, are often unavoidably present in the original footage. Such objects commonly include cameras and lighting rigs. Removing these elements again requires them to be modelled in some fashion, so that they can be masked out, and a new background synthesised to fill the masked space.

Visual effects will also frequently require synthetic models of real world objects, so that these objects can be used in ways that would be either physically impossible or prohibitively expensive using the objects themselves. Frequently, both original and synthetic versions of an object will be used, varying with the specific situation, and as such the synthetic version will need to be a convincing analogue of the real object for the transition to be seamless. In these cases, modelling from video provides one solution for the generation of the required synthetic models. Figure 2.5 shows a frame of a video sequence in which a 3D model of a car is placed side-by-side with the real car.

2.3.3 Uses for augmented reality

In Augmented Reality (AR), computer generated imagery is incorporated into live video footage of real environments. Convincingly incorporating synthetic elements into a real environment can require modelling geometry of that real environment, so that this geometry can be used to construct occlusion masks, be used to model interactions between real and simulated objects, and be used for graphical effects such as lighting and shadows. Interactive modelling provides a means of rapidly constructing this geometry. Two interactive modelling methods for AR applications are presented in Chapter 6. The method of [62] performs camera tracking and constructs a point cloud in real time, by performing real time frame-to-frame tracking in parallel with a process performing bundle adjustment on a set of key frames. The sparse point cloud produced by this process can form the basis for further reconstruction. In [63] a toolkit is described for constructing a model of an environment using a wearable computing platform. Visual information is not used. Rather, it relies on accurate tracking and a user specifying point locations in multiple views to generate vertices.

The AR system described in [64] makes use of user interaction and an aerial image of the scene for tracking and reconstruction. The user clicks on corresponding points on the aerial image and an image from the camera's video stream. This initialises an automated process for fitting a model of a building corner to the video frame and aerial view, making use of GPS and an inertial sensor to estimate the camera position and orientation. Feature and edge tracking is applied to fit the building model to subsequent frames, and this fitted model is used to recover camera pose.

In [65] camera tracking is performed online, and the user constructs model vertices by clicking on a point in a single view to specify a ray, then setting the position of the vertex on that ray in a second view. Once a model has been defined, it can be used in the tracking process. While visual information is used to reconstruct camera pose, this information is not used in the modelling itself. In [66] models are interactively constructed using a contact probe (tracked by IR LEDs) to manually specify points on the object surface. The model is overlaid on the object in an AR head up display. The method of [67] does not perform a reconstruction, but does perform 2D tracking of an object with a graph cut. The cut is performed in a band of pixels around a predicted silhouette in a new frame, with this prediction being the silhouette from the previous frame offset by the average optical flow of a set of tracked feature points. The costs used in the cut are based on edge detection and the predicted silhouette, without considering colour.

A semi-automated online reconstruction technique is presented in [68], in which a model is generated from an object being manually rotated in front of a fixed camera. The problem of segmenting the object from the background is solved by only reconstructing the moving geometry of the scene. The moving object is first reconstructed as a point cloud. An initial model is generated from this point cloud by Delaunay tetrahedralistion. A probabilistic carving step is used to remove spurious faces and obtain the final model. The system guides the user to rotate the object to views needed to complete the model. In the live dense reconstruction method of [69], a base mesh is first fitted to the set of features returned by a real time SfM process. For dense reconstruction, this mesh is then deformed using the dense optical flow computed between a reference frame and a set of adjacent frames. Flow estimates are constrained to encourage local consistency. This method relies on the assumption of photoconsistency and a relatively accurate estimate of the base mesh.

3

Fitting models to images

Viewing a video sequence of a real-world environment, humans can easily recognise objects in the scene and their relationships. For some applications requiring 3D models, such a high-level understanding of a scene may be necessary to obtain a suitable 3D reconstruction. By recognising structures in the scene as either corresponding to primitive model types such as boxes or cylinders, or as corresponding to a specific object for which a 3D model already exists, we can obtain reconstructions giving a more complete and accurate model of the scene than is possible from only reconstructing the parts which are visible in the sequence. This high-level understanding will also be needed where a purely geometric reconstruction is insufficient, and a model incorporating semantic information on the content of the scene is required. Such high-level information is challenging to recover with an automated process, particularly if insufficient examples of similar scenes are available to apply a machine-learning approach to scene understanding. Reconstructing a scene with a manual process such as CAD modelling will allow a user to model the scene to the level of detail required, but can be extremely time-consuming, and unintuitive for novice users.

In this chapter we present an approach to the problem of recovering 3D structure from a video of a scene using user-supplied information, while minimising the amount of user interaction required. The particular problem that we are concerned with in this chapter is 3D reconstruction of elements in a video sequence for which a pre-defined 3D model exists, but the parameters required to map the model into the scene are unknown. This can either be a detailed model representing a specific object, or a model representing a generic primitive type. We do not assume that texture information is available for the model. While we assume that point cloud data can be recovered for the object in the video sequence, we expect that this data will be sparse for objects with regions of minimal texture. Subsequent chapters will present methods for cases where no such pre-defined model exists.

We describe a novel method for fitting pre-defined 3D models to a set of images from a video sequence, exploiting high-level information supplied by a user. This information is used to guide a fitting method using both appearance information from 2D images and sparse 3D information from Structure from Motion. The method allows the user to initialise the fitting process with minimal interaction, and to use further interaction to refine fitting results to the desired level of accuracy.

The model fitting process is initialised by a user selecting a model type found in a scene, and indicating the location of the object in one frame of a video of the scene for which Structure from Motion data is available. Two methods are provided for indicating the location of the object, the first using a single click, and the second using coarse marking of an image. A set of hypothesised models is generated, spanning the range of possible parameter values given the user-supplied information. Fitting proceeds in a coarse-to-fine manner, using 3D point cloud data to select and refine a set of hypothesised models which may be close to the optimal parameter values, and using 2D image information to select and refine the model which will be presented to the user.

In comparison with other interactive modelling techniques, the main novelty of this method is that it allows for fitting with minimal interaction where possible. Systems which allow scenes to be modelled as a collection of primitive objects typically require the user to specify as many constraints as there are free model and camera parameters. For example, in the Facade system [2] the user must mark out sufficient model edges in one or more views to recover the parameters for the cameras and the primitive blocks used to model the scene. In the system described in [44], which uses parallelepiped primitives to model the scene, the user must provide enough point correspondences to define the cameras and primitive parameters. While the ICARUS system [1] uses SfM to recover camera parameters automatically, it still requires the user to specify sufficient vertex locations for the primitives in multiple frames to fully define the model parameters. Automated methods for fitted models to images, such as [37], typically require a large set of labelled training images. Automated methods for matching models to 3D data, such as [32], typically assume that dense data such as that recovered by laser scanning is available.

Rather than requiring the user to supply sufficient constraints to fully define the model parameters, which requires either detailed interaction in multiple frames or specifying a number of constraints in a single frame, we only require the user to interactively provide some initial partial constraints. This initial interaction is performed in a single frame of the sequence, and does not require precision. Given these constraints, our method then determines the complete likely model parameters from the point cloud and image data. Although simple, the initial interaction solves the difficult task of identifying the appropriate model, significantly constrains the space of possible model parameters, and simplifies the task of identifying feature points belonging to the object.

After fitting has been performed, the user can select the hypothesised model with the best fit, and refine the fitting results if required. While we use well-known techniques for fitting geometry to 3D point and 2D image information, we combine these techniques with user-supplied information in a novel hierarchical framework. From the initial interaction, a set of hypothesised models is generated spanning the possible model parameters, and evaluated and optimised using the point cloud and image information. For efficient evaluation, tests based on the 3D data are performed first, and used to select the best candidate models, to which more computationally expensive testing and optimisation using the image data is applied. Finally, the user can add additional vertex position constraints to the selected candidate model to improve the quality of the fit if required.

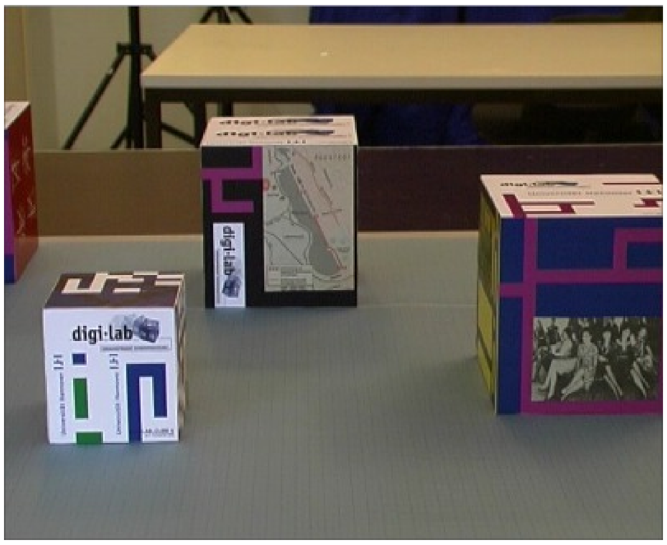
In Section 3.1 we describe the video data used as input to the fitting method, how the video data is acquired, and the process used to recover camera and point cloud data. In Section 3.2 we give a definition for the models fitted by this method and their possible parameters. Section 3.3 provides a description of the hierarchical fitting method, followed by specific examples of likelihood functions used by the method. In Section 3.4 we then describe the application of the complete fitting process. The chapter closes with an experimental evaluation of this process for a set of model types and sequences in Section 3.5. Work presented in this chapter was originally introduced in [73], [74], and [75]. Further development of this method is described in [76]. The method described in this chapter was developed by the author with input from his thesis supervisors, and implemented and tested by the author.

3.1 Input to the method

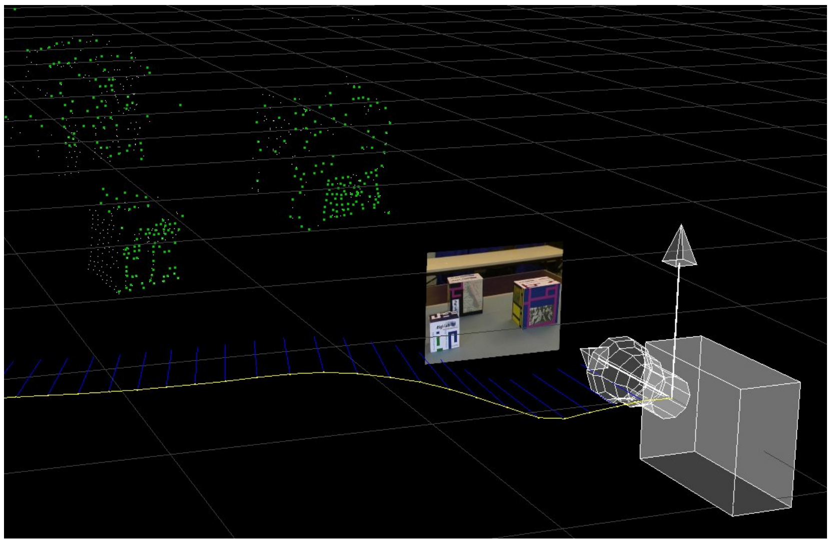
As input, the fitting method takes a set of images of a scene, typically sourced from a video, and a set of camera parameters and feature points reconstructed from those images by a Structure from Motion process (described in 2.1.1.2). The reconstruction methods described in this thesis were developed for use on video sequences, as for suitable sequences the parameters of the cameras can be reliably recovered automatically. This avoids the need for the user to manually provide sufficient constraints to recover the cameras, as employed in methods such as those presented in [2] and [3]. Automatic extraction of camera parameters is similarly used as the first stage of an interactive reconstruction process in the ICARUS system [1].

The use of SfM restricts these methods to sequences for which SfM is suitable. SfM processes typically assume that the scene in the video is largely static, free from moving objects such as people walking through the view, and that the scene includes enough texture to provide sufficient point correspondences in each frame. SfM is suitable for video shot with standard off-the-shelf digital camcorders, and sufficiently robust to work with video shot by a moving person with a handheld camera, assuming that the camera shaking is not too severe. As successful SfM reconstruction typically requires both camera translation and rotation, videos used in this thesis were captured by moving in an arc around the object or scene to be reconstructed, while keeping the camera roughly centred on any object of interest in the scene. Methods described in this thesis could also be applied to image sets from sources other than video, provided that accurate camera parameters, images from a sufficient range of views, and a sufficiently detailed point cloud are available.

The set of feature points recovered by SfM includes both a set of 3D points, represented in homogeneous coordinates as $\mathbf{P} = (P_1, P_2, P_3, 1)$, and a set of corresponding 2D image points, represented in homogeneous coordinates as $\mathbf{p} = (p_1, p_2, 1)$. Each 3D point $\mathbf{P}_i$ has an associated set of 2D points, where each $\mathbf{p}_{i,k}$ is the projection of $\mathbf{P}_i$ into the kth image of the sequence. Each 3D point will only have 2D points for those frames in which it was observed. For each image in the sequence, there is an associated camera matrix $\boldsymbol{A}_k$. The projection of a point $\mathbf{P}_i$ into the kth image is given by $\mathbf{p}_{i,k} = \boldsymbol{A}_k\mathbf{P}_i$. Figure 3.1 shows the results of SfM reconstruction when applied to a test sequence. These results were generated with the Voodoo camera tracker [77].



(a)



(b)

Figure 3.1: Results of SfM reconstruction. a: A frame from a tracked sequence. b: The camera and image plane for the frame, recovered 3D feature points, and camera path over the sequence (shown in yellow)

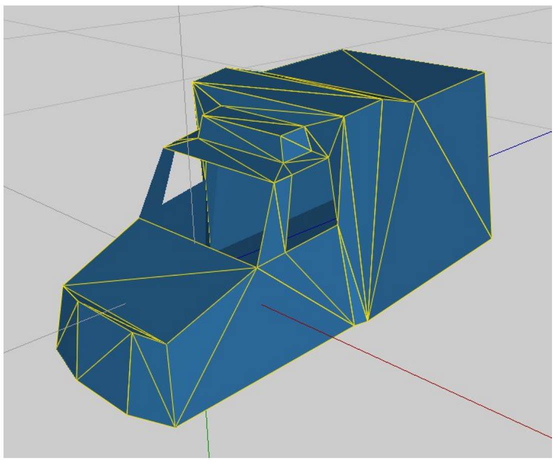


Figure 3.2: Mesh model for a toy ambulance.

3.2 Model definition

Models encode the 3D structure of an object, and can additionally encode semantic information about the object type. The properties of each model instance are specified by a parameter vector $\mathbf{M}$, which will vary depending on model type. For model types with a fixed shape the parameter vector describes the position of the model's centroid $\mathbf{C} = (C_1, C_2, C_3)$, its orientation $\mathbf{R} = (R_1, R_2, R_3)$ in Euler angles, and scale S. Orientation parameters are used to define a rotation matrix $\boldsymbol{R_M}$. These parameters are used to define a model transform matrix $\boldsymbol{T_M}$, mapping points on the model surface into the coordinate system of the scene by applying the transform $\mathbf{P_S} = \boldsymbol{R_M} S\mathbf{P} + \mathbf{C}$, and the inverse transform $\boldsymbol{T_M}^{-1}$, mapping points in the scene to the model coordinate system. Other models may require alternate parameters. For example, a cylinder primitive may use a scale vector $\mathbf{S} = (S_1, S_2)$ to define the length and radius of the cylinder. The definition of a model is kept quite general, so that the described method will be applicable to a wide range of model types, including both abstract geometric types, and more detailed models representing specific objects.

3.2.1 Model representation

In implementing this fitting process, a triangle mesh representation was used for the model types. Triangle mesh models were chosen for their flexibility in representing a wide range of possible model types, and for the efficiency with which the likelihood functions used by this fitting process could be evaluated for this model representation. This representation is commonly used in applications requiring 3D models as it can be stored compactly and rendered efficiently on modern graphics hardware. The fitting process could equally be applied to other model representations, such as parameterised surface representations, with the appropriate implementation of the likelihood function evaluations. Figure 3.2 shows an example mesh representation for one model type.

3.3 Fitting process

The goal of the model fitting process is to find the most likely set of parameters for a model, specified by parameter vector $\mathbf{M}$, given the data extracted from the image set D, and available prior information I, including the information supplied by the user. We formulate this problem in terms of Bayesian inference, aiming to maximise the probability $\Pr(\mathbf{M}|DI)$. By Bayes' Theorem [78],

$$\Pr(\mathbf{M}|DI) \propto \Pr(D|\mathbf{M}I)\Pr(\mathbf{M}|I), \tag{3.1}$$

where $\Pr(D|\mathbf{M}I)$ is the likelihood of the image set data given the model parameters and prior information, and $\Pr(\mathbf{M}|I)$ is the probability of the parameters themselves.

The information supplied by the user specifies a model type, and a rough location or selection of the model in one image. Data extracted from the image set consists of 2D data extracted from the images directly, and 3D data extracted from the SfM results. The 3D data, consisting of a sparse point cloud, gives 3D positions for discriminative features in the scene. It should be noted that, as the SfM results are derived from the images, this information is not independent. Rather, the 3D data is treated as an approximation to the 2D data.

Model fitting is initialised by a user selecting a model type to fit, and roughly indicating the location of the object to model in an image from the sequence, or roughly selecting that object in one image. We refer to this image as the *initial frame*. This was chosen as the initialisation step to minimise the amount of interaction required from the user. Details of the interactive initialisation are given in Section 3.4.1. The model fitting process considers both the likelihood for the 3D information $\Pr(D_3|\mathbf{M}I)$ and for the 2D information $\Pr(D_2|\mathbf{M}I)$. After the fitting process has been applied, model fitting results can be refined using the likelihood $\Pr(D_U|\mathbf{M}I)$ considering additional information provided by the user.

Evaluating $\Pr(D|\mathbf{M}I)$ for an exhaustive set of samples is infeasible for an interactive system where the user will expect fast feedback. To efficiently explore the parameter space, we employ a coarse-to-fine strategy, exploiting the particular properties of $\Pr(D_3|\mathbf{M}I)$ and $\Pr(D_2|\mathbf{M}I)$. A set of hypothesised models is generated, with parameters spanning the set of expected parameter values given the constraints supplied by the user. The set is subjected to progressively more accurate, but also more computationally expensive likelihood tests, with hypotheses with low likelihood being removed at each stage. Parameters of the hypothesised models can be optimised at each stage. This process of performing a series of testing stages, and retaining the best hypotheses at each stage, is similar to the Preemptive RANSAC scheme employed in systems such as [79]. Details on applying the fitting process are given in Section 3.4. Before describing the details of this process, we discuss the likelihood functions.

$\Pr(D_3|\mathbf{M}I)$ determines the likelihood of the 3D data associated with a sequence given a particular instance of a model. 3D likelihood is determined based on how close the surface of the model is to nearby 3D points, giving a likelihood based on the sparse description of the 3D structure of the scene provided by the point cloud. This function is described in Section 3.3.1.

$\Pr(D_2|\mathbf{M}I)$ gives likelihood based on the dense information contained in the image set, determining likelihood from the appearance of the model when projected into the images. This likelihood is well suited to precise object localisation, but only for models close to the true object location, due to the presence of local minima. Detail on this function is given in Section 3.3.2.

The 3D likelihood function is typically computationally cheaper to evaluate than the 2D likelihood, as the 3D data is significantly sparser than the image data. For the same order of computational cost, the 3D likelihood allows a more thorough exploration of the parameter space. As such, this likelihood is suited to the initial coarse localisation of objects in the scene. 2D likelihood is then used to refine this initial estimate. After fitting has been performed, the user can further refine the results if necessary with manual input. For this, $\Pr(D_U|\mathbf{M}I)$ is used, giving likelihood based on model position information directly supplied by the user. Detail on this function is given in Section 3.3.3.

Algorithm 1 describes the fitting process for n hypothesised models, given a set of m functions $\{\mathcal{L}_i(D_i, \mathbf{M})\}$ each used to evaluate the likelihood of data D_i given model $\mathbf{M}$. $\{f_i\}$ gives the fraction of the hypothesised models remaining at test stage i.

Algorithm 1 Model fitting

```
Generate a set of hypothesised models $\{\mathbf{M}_j\}$ spanning the parameter space given
user supplied information $I$
for $i = 1$ to $m$ do
  for $j = 1$ to $f_i n$ do
    Evaluate $\mathcal{L}_i(D_i, \mathbf{M}_j)$, optimising if required
  end for
  Sort $\{\mathbf{M}_j\}, j = 1 \dots f_i n$ by $\mathcal{L}_i(D_i, \mathbf{M}_j)$, in ascending order
end for
Present selected model $\mathbf{M}_s = \mathbf{M}_1$ to the user
The user can select another model from $\{\mathbf{M}_j\}$, if required
Optimise $\mathcal{L}_U(D_U, \mathbf{M}_s)$ for additional user-supplied data $D_U$, if required
```

3.3.1 3D likelihood functions

We first introduce a likelihood function suitable for the typical case, where the available 3D data D_3 consists of a cloud of points $\{\mathbf{P}_i\}$. For objects with sufficient feature detail, this cloud will include points giving a sparse representation of the surface of the object. The likelihood function is based on the distance to the surface of a model for points in its vicinity. The function favours models with strong support from the data, in this case meaning models with many points lying close to or on the model surface. The function $d_3(\mathbf{P}_i, \mathbf{M})$ measures the absolute distance from a point to the closest point on the model surface. We evaluate the distance from each point to the model in 3D space.

For a model $\mathbf{M}$ defined by a mesh consisting of a set of vertices $\mathbf{V}$, faces $\mathbf{F}$ and edges $\mathbf{E}$, this distance is defined as the minimum distance from the point to any face of the model. The minimum distance from a point to a face is given by

$$d_F(\mathbf{P}_i, \mathbf{F}_j) = \begin{cases} |\mathbf{P}_i \mathbf{N}_j + o_j| & \text{if the closest point on the plane is inside the face} \\ \min_k d_E(\mathbf{P}_i, \mathbf{E}_{jk}) & \text{otherwise} \end{cases} \tag{3.2}$$

where $\mathbf{N}_j$ is the normal for the plane containing face $\mathbf{F}_j$, o_j is the distance from that plane to the origin, $d_E(\mathbf{P}_i, \mathbf{E})$ measures the minimum distance from the point to a line

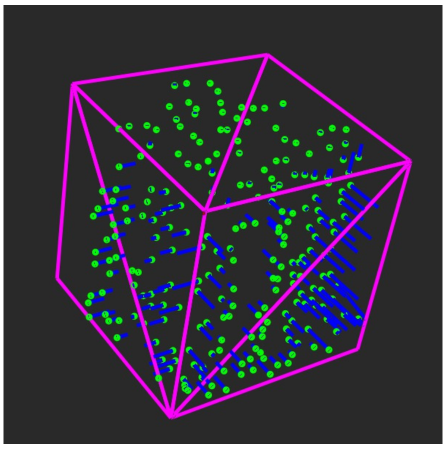


Figure 3.3: Distances from points to a cube model. Blue lines show the distances from feature points to the nearest point on the cube surface.

segment $\mathbf{E}$, and $\{\mathbf{E}_{jk}\}$ is the set of edges belonging to the face. Figure 3.3 shows the distances between a set of points and the surface of a cube model. Similar measures of 3D distance are used in common model fitting approaches based on the Iterative Closest Points algorithm [30].

This distance could alternately be evaluated as the 2D distance in image space between a point and the nearest point on the model surface. This, however, would mean the distance was dependant on the relative orientation of the camera and the surface, with the measured error for an equivalent 3D distance being reduced for surfaces closer to parallel with the image plane. This problem could be reduced by measuring 2D distance in a number of images. However, this would require multiple projections and distance calculations, increasing the cost of evaluating the likelihood.

Assuming that all 3D points are independent, the likelihood $\Pr(D_3|\mathbf{M}I) = \Pi_i \Pr(\mathbf{P}_i|\mathbf{M}I)$. In optimising and selecting from hypotheses, we can evaluate the average negative log likelihood for a set of n points $\{\mathbf{P}_i\}$ with the function $\mathcal{L}_3(\{\mathbf{P}_i\}, \mathbf{M}) = \frac{1}{n}\sum_{i=1}^{n} -\log(\Pr(\mathbf{P}_i|\mathbf{M}I))$. We take the average as the number of points will vary between hypotheses. Assuming that the error in the points recovered with SfM is normally distributed,

$$\mathcal{L}_3(\{\mathbf{P}_i\}, \mathbf{M}) = \frac{f_3}{n}\sum_{i=1}^{n} d(\mathbf{P}_i, \mathbf{M})^2, \tag{3.3}$$

where f_3 is a constant scale factor. Such scale factors are used to weight the contribution of each likelihood function when evaluating a model using multiple likelihood measures.

This assumes that all points in the set will correspond to points on the surface of the object being modelled. However, in the vicinity of an object, points may be contributed by features of the surface that the object is resting on, and by features of other nearby objects. At this stage of the fitting process, we do not expect a hypothesised model to be close enough to the true model parameters that outlier points can be reliably determined and excluded from the likelihood evaluation. Rather, we use a robust measure to reduce the influence of outliers on the likelihood function. For robustness to points not belonging to the object, a Huber function [80] $\psi(d_3(\mathbf{P}_i, \mathbf{M}))$ is applied to the distance measure.

The Huber function is defined as

$$\psi(x) = \begin{cases} \frac{x^2}{2},\ |x| < \epsilon \\ \epsilon|x| - \frac{\epsilon^2}{2},\ \epsilon \leq |x| \end{cases} \tag{3.4}$$

and limits the influence of points beyond a threshold distance ϵ, to prevent the cost being dominated by outliers. The Huber distance transitions between a squared loss and absolute loss at the threshold distance ϵ.

The Huber function requires specifying an appropriate value for the threshold ϵ. This threshold is set to a robust standard deviation estimate, given by $1.4826\,\mathbf{MAD}$, where $\mathbf{MAD}$ is the median absolute deviation of x. An explanation of this robust standard deviation is provided in [81].

With the Huber function applied, the complete likelihood function is defined as

$$\mathcal{L}_3(\{\mathbf{P}_i\}, \mathbf{M}) = \frac{f_3}{n} \sum_{i=1}^{n} \psi(d(\mathbf{P}_i, \mathbf{M})) \tag{3.5}$$

3.3.1.1 Evaluating distance to the model surface

To evaluate the distance from a 3D point to a mesh model surface, we determine the minimum distance from the point to any face of the model. Determining this distance is potentially computationally expensive for models with a large number of faces. For efficient evaluation, we use the fact that the distance from a point to a plane containing a face gives a lower bound on the minimum distance to that face to reduce the number of faces for which we must perform tests involving the plane boundary. We first determine the distance from the point to the planes containing each face. The distance from the point to each face is evaluated in increasing order of this distance. If the closest point on the corresponding plane is within the boundary of the current face, this distance is taken as the distance to the model surface, and evaluation terminates. Otherwise, the distance to the face is taken as the minimum distance to an edge of the face. Subsequently, the distance to the remaining faces will only be evaluated if the distance to the corresponding plane is less than the minimum distance to an edge of the current face. To evaluate whether the nearest point on the plane is within the bounds of the face, we project the point onto the plane containing the face, and determine whether this point is within the corresponding 2D projection of the face. This is significantly less expensive than perform this testing in a 3D space.

To evaluate $\mathcal{L}_3$, a set of features and model faces must be selected. The set of features is selected as the set of all features with a corresponding measurement in the initial frame. This will exclude features belonging to objects occluded by the selected object in the initial frame, reducing the possibility of fitting to features that do not belong to the selected object. The selected set of model faces contains all faces which are visible in the initial frame, as this is the set from which features with a correspondence in the initial frame will be drawn.

3.3.1.2 3D template function

To significantly reduce the time that the user spends waiting for a fitting result, we employ a coarse-to-fine strategy, computing the relatively inexpensive likelihood given 3D data first, and using this to eliminate hypothesised models with low likelihood before evaluating 2D likelihood. A coarse-to-fine strategy can also be employed in evaluating the individual likelihood functions, with a coarse approximation of the likelihood function used for initial localisation, and a more accurate likelihood function used for refinement. We initially apply a template function $T(\{\mathbf{P}_i\}, \mathbf{M})$ for the given model and point set, which determines an approximate likelihood given the 3D data. This function approximates the model based on distances from points on its surface to its centroid.

As the centroids of the candidate models are informed by the initialisation information provided by the user, we assume that models will be generated close to the true object centroid. However, as orientation is coarsely sampled, we assume that there will be a significant difference between the orientation of hypothesised models and the true object orientation. For this reason, we use a rotationally invariant template.

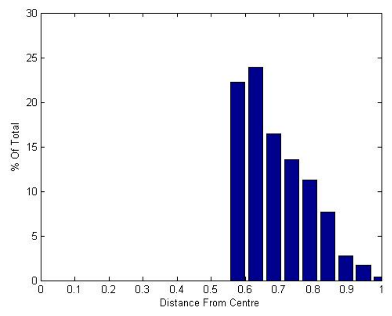


Figure 3.4: Histogram of distances from the centre for points on the surface of a cube model

The template for a model is generated by sampling over the surface of the model and building a histogram H_M of distances from the points on the surface to the model's centroid $\mathbf{C}$. Figure 3.4 shows the histogram generated for a cube model. A similar representation of a 3D model is constructed for the shell model shape histogram used for model searching and classification in [31].

This template is compared to a histogram H_P of distances from 3D points to the model centroid. This distance is measured in the coordinate system of the model, to remove the effect of scaling on the distances. As we do not expect to have sampled the precise centre and scale of the object, we apply Gaussian smoothing to the histogram to reduce the effect of the distance between the centroid and scale of a hypothesised model and those parameters for the true object. Comparison is performed using Kullback-Liebler divergence [82], a measure of the difference between two statistical populations, used to compare a set of samples with the true distribution of points on the surface:

$$T(\{\mathbf{P}_i\}, \mathbf{M}) = \sum_i H_{Pi} \log \frac{H_{Pi}}{H_{Mi}} \tag{3.6}$$

We chose this function primarily for the speed with which it could be evaluated. As the function only requires determining the distance from feature points to the object centre and comparison between 1D histograms, the cost of computing this function is significantly lower than the cost of evaluating distance to the object surface, and is not dependant on the complexity of that surface. Although this test is simple, it is sufficiently discriminative to discard a large number of hypothesised models in regions which are unlikely to contain the object.

A more sophisticated method for rotation-invariant matching between models and 3D point data is provided by the *spin image* method of [32]. A spin image is a 2D histogram representing the distribution of points on the surface of an object in a coordinate system defined with respect to an oriented point on that surface. A set of spin image descriptors can be used to efficiently match a pre-defined model to 3D scene data. However, computation of a spin image requires point data which is sufficiently dense that the orientation of the surface can be determined at a selected point. As the SfM process will typically recover only sparse point data for much of the scene, such a method is not suitable for our fitting process in the general case, but could be useful for environments where point data is sufficiently dense.

An alternative rotation invariant representation is constructed in [33]. In this method, a frequency domain decomposition is performed on a spherical function representing the shape, and a descriptor is computed from the magnitudes of the function's frequency components. A spherical representation of a voxel grid is generated by computing the intersection of the grid with a set of concentric spheres, and a descriptor is computed independently at each radius. This method similarly assumes that a dense representation of the object is available.

3.3.2 2D likelihood function

2D likelihood functions determine the likelihood of a set of images given a particular model instance. The functions favour models for which some selected aspects of their appearance are strongly supported by the data. 2D likelihood functions can be based on different appearance properties, such as colour and edge information. Different functions would be appropriate for objects with different surface properties. Here we describe a 2D likelihood function based on photoconsistency. Photoconsistency is ideal for objects with regions of significant surface detail, and a surface with a consistent appearance over the sequence. A likelihood function based on edge information could be more appropriate for objects with strong edges but minimal surface detail, or inconsistent appearance due to a translucent or reflective surface. 2D likelihoods are computed for a subset $\{F_j\}$ of all frames of the sequence, given the large amount of redundancy expected between nearby frames.

3.3.2.1 Photoconsistency likelihood

The 2D likelihood function is based on evaluating the photoconsistency of points on the model surface when viewed from multiple cameras. A range of reconstruction approaches using photoconsistency were described in Section 2.1.3. Here we use the assumption that the object can be modelled as a Lambertian surface. If the illumination of the scene does not change significantly between frames, we expect that the appearance of points on the surface of an objects will not vary significantly from image to image.

Under this assumption, and assuming that the object is not occluded by other objects in the scene, the likelihood of a given model can be determined from the variance in pixel colour values for points on its surface over the sequence. Assuming normally distributed error, the average negative log likelihood is given as

$$\mathcal{L}_{2_p}(\{F_j\}, \mathbf{M}) = \frac{f_2}{n} \sum_{i=1}^{n} \operatorname{var} \{F_j(\boldsymbol{A}_j \mathbf{S}_i)\} \tag{3.7}$$

where $\{\mathbf{S}_i\}$ is a set of n point samples from the object surface, $\boldsymbol{A}_j$ is the projection matrix for frame j, $F_j(\mathbf{s})$ gives the pixel colour for point $\mathbf{s}$ in frame j, and f_2 is a constant scale factor. Variance for a sample is determined over frames where the sample is visible.

This function assumes that the colour of each point on the surface will be consistent in all views. In environments where this is not the case due to variation in lighting or exposure in different views, another measure may be more appropriate. While more expensive to evaluate, using Normalised Cross Correlation [20] to measure consistency would provide robustness to intensity variation between different views of the same point. A description of NCC was given in section 2.1.3.

3.3.2.2 Evaluating photoconsistency likelihood

To generate the set of surface samples, a set of model faces is first selected from which samples will be drawn. This set includes all faces which are visible in more than 1 frame of the sequence. A face is considered visible in a frame if it is within the bounds of the frame and facing the camera. From this set of faces, the set of samples is generated. Samples are generated evenly within triangles, with the number of samples per triangle determined according to its area relative to the largest triangle in the model, to avoid

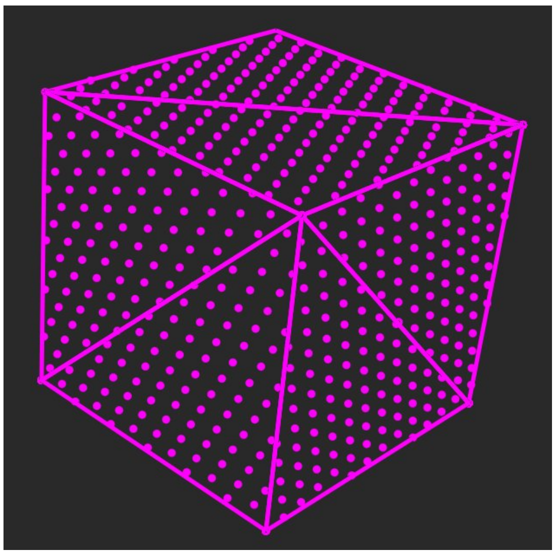


Figure 3.5: Sample points generated for the cube model

biasing the fitting towards faces with smaller area. Each sample point is represented by its Barycentric coordinates for that triangular 3D face, so that for a given set of model parameters, a 3D location for each sample point can be generated from the coordinates of the vertices of the face containing that point. This allows for simple recomputation of the sample point locations when model parameters change. Figure 3.5 shows a set of sample points generated for the surface of a cube model.

For each face, a set of frames is selected over which samples for that face are considered visible. As in selecting visible faces, a frame is added to the set for each face if the face is within the frame bounds and facing the camera. As it may be prohibitively expensive to evaluate variance over all frames of a sequence, and the difference between adjacent frames is typically small, a set of at most 5 frames is used to evaluate the variance for each face. The set is selected to maximise the difference between each frame. This set is selected with a procedure iteratively selecting the frame from the set of all possible frames with the largest minimum angle between the ray from the face centroid to the camera's optical centre and the corresponding ray for any frame already selected, beginning with the initial frame.

3.3.3 Interactive refinement

Interactive adjustment of a selected model is performed by the user specifying locations of individual model vertices in one or more frames. Given a set of model parameters $\mathbf{M}$, each user specified vertex position $\mathbf{u}_{ij}$ corresponds to the desired position for vertex $\mathbf{V}_i$, projected into frame j. The projection of $\mathbf{V}_i$ into frame j is given by $\boldsymbol{A}_j\mathbf{V}_i$, where $\boldsymbol{A}_j$ is the camera matrix for the image. The function $d_2(\mathbf{u}_{ij}, \boldsymbol{A}_j\mathbf{V}_i)$ gives the distance between the reprojected vertex and the user specified position in image space. Again assuming normally distributed error, we evaluate the negative log likelihood of a set of user-specified vertex positions $\{\mathbf{u}_{ij}\}$ given a parameter vector $\mathbf{M}$ as

$$\mathcal{L}_U(\{\mathbf{u}_{ij}\}, \mathbf{M}) = f_U \sum_{ij} d_2(\mathbf{u}_{ij}, \boldsymbol{A}_j\mathbf{V}_i)^2 \tag{3.8}$$

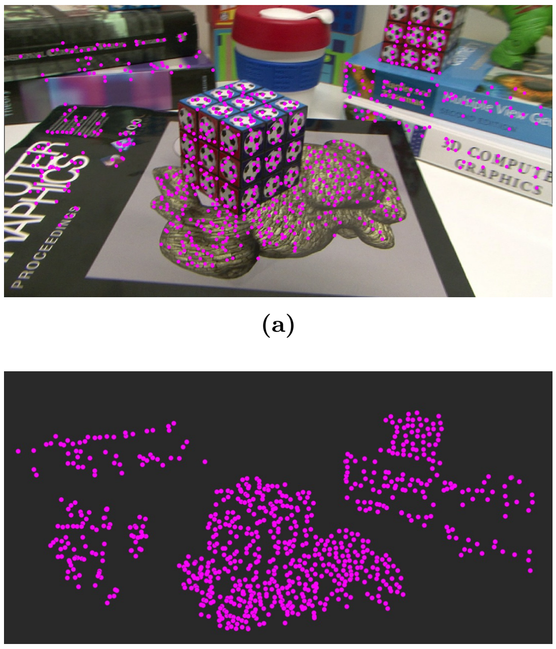
where f_U is a constant scale factor. This measure is the same as that used for optimising models given user-specified vertices in systems such as ICARUS [1].

3.4 Fitting process

The following section describes the process for fitting a model to the image and point data. Figure 3.6 shows an example of a selected initial frame from a sequence, displaying the feature point correspondences for that frame, and a view of this cloud of features.

3.4.1 Initialisation

The fitting process begins with the user selecting a model type, and indicating the object to fit to in a frame of the sequence. Two interfaces are provided for selecting the object. The object can be selected by clicking near to its centre in the image. We refer to this as a *single click* initialisation. Alternately, the object can be selected by the user drawing an outline around the object. This outline can be drawn quite coarsely. An example of a user-drawn outline is shown in Figure 3.7. This outline is used to specify the set of features used in the fitting process. We refer to this as an *outline* initialisation. Although this initialisation is slightly more involved than a single click, it can give better results in cluttered scenes where isolating features belonging to the object is difficult, or where good fitting results (as determined by the likelihood functions) can be obtained for a model fitted to only a subset of the object features.



(a)

(b)

Figure 3.6: Cube test sequence a: A frame from the sequence showing feature point locations. b: The cloud of 3D features with correspondences in that frame.

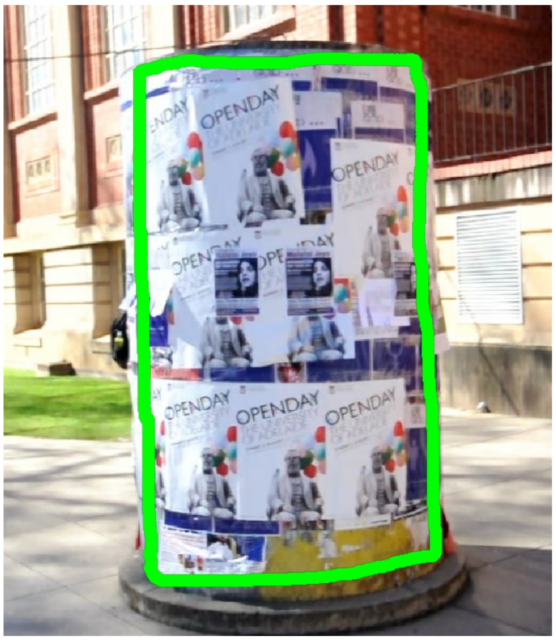


Figure 3.7: Manually selecting an object

3.4.2 Sampling model parameters

As the likelihood functions are subject to local minima, we sample model parameters with the goal of generating at least one model with parameters closer to the global minimum than any other local minimum. Centroid, scale, and orientation parameters are sampled for a set of hypothesised models. We generate samples spanning the expected range of parameters, given the user interaction. For some model types, such as a model of a 2D planar region, hypothesised models could alternately be generated by sampling sets of feature points and using these points to define the model parameters. However, this would be challenging for models with a complex surface. As we do not know in advance the corresponding point on the model for a given feature point, for any selected set of feature points there would potentially be many sets of model parameters for which all selected points would lie on the surface.

The centroid parameter C_1 specifies the distance of a model from the camera along a ray through a centre point specified by the user, with C_2 and C_3 specifying distance from the ray in two orthogonal directions, orthogonal to the ray. As it is assumed that the image point specified by the user is close to the centre of the object, C_2 and C_3 are set to 0 for the initial set of hypothesised models. For the outline initialisation, this point is set to the centroid of the drawn outline.

The range of distances over which C_1 is sampled is set to the range between the closest and the furthest feature point from the camera in the initial frame. Here, we consider only points which were observed in that frame, which can significantly reduce the range of space over which samples are generated, compared with generating from all feature points. Due to the effects of perspective projection, the effect of variation in the distance of an object from the camera on the appearance of that object in the image is greater the smaller its distance from the camera is. C_1 is therefore sampled on a log scale, so that distances are sampled more densely closer to the camera. For the single click initialisation, the minimum scale sampled at each distance from the camera is selected on the assumption that the object will occupy at least 10% of the image. In selecting the maximum scale, we assume that the object is within the image bounds. The maximum scale at each depth is set to the largest scale at which a bounding sphere around the model is within the bounds of the frame. Scale parameters are evenly sampled within this range. For the outline initialisation, the maximum scale at

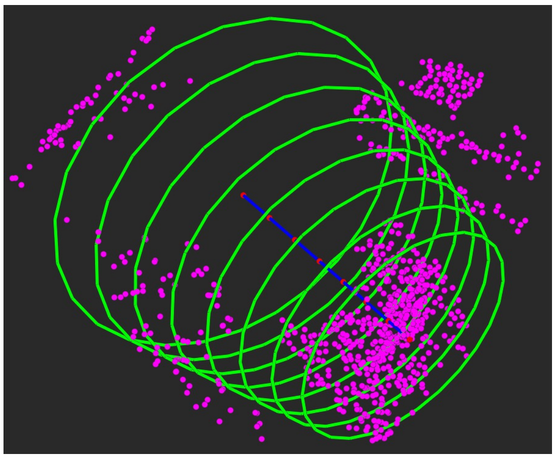


Figure 3.8: Positions and scales for hypothesised models. Centre positions are shown as red dots. The maximum scale at each centre position is shown as a green circle

each depth is determined from the bounds of the outline. **R** is evenly sampled over the full range of possible orientations. Figure 3.8 shows the range of positions and maximum scales generated in fitting a cube model to the scene shown in Figure 3.1.

3.4.3 3D likelihood evaluation

Evaluating 3D likelihood requires selecting a subset of SfM points and determining their distance to the model. If an outline initialisation is used, we select all points in the initial frame with a projection within the selected region, assuming that points in this region will correspond to points on the surface of the object. In the case of single click initialisation, the parameters of each model hypothesis are used to define a bounding box around the object. The bounding box is used as an approximation of the complete object, and provides an inexpensive means of selecting points in its vicinity. The scale of the bounding box is set to 1.3 times the scale of the sampled model, to account for the difference between the parameters of the sampled model and the true model parameters. We perform a count of the number of points within each bounding box. We then reject any model with a bounding box containing fewer points than the model has parameters, for which the optimisation problem is under-constrained. Although this threshold is low, it still results in a significant reduction of the search space, as a scene will typically contain more empty space than occupied space. A slightly simpler option for selecting points in the vicinity of the model would be to use a bounding sphere

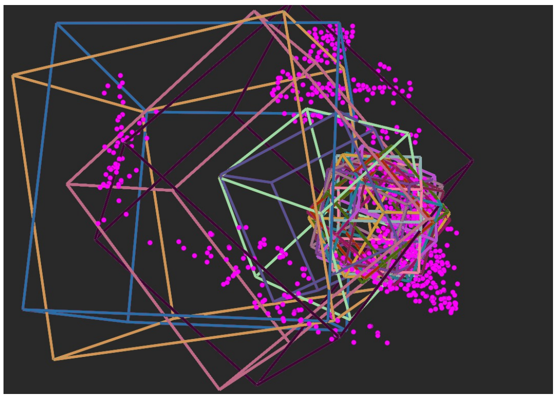


Figure 3.9: The top ≈ 100 results for the template likelihood on the cube sequence

instead of a bounding box. However, this would be expected to include an excessive number of outlier points for models with a significant difference in the maximum extent of the model in different dimensions.

To reject further locations and scales unlikely to correspond to the true parameters, the template function $T(\{\mathbf{P}_i\}, \mathbf{M})$ is evaluated for each hypothesis, and the top 50% of results are retained. We use this initial 3D likelihood approximation to indicate regions which are likely to contain objects with a similar shape to the selected model type. Figure 3.9 shows a selected set of models with high likelihood as determined by the template function.

For each remaining hypothesised model, we evaluate the 3D likelihood function $\mathcal{L}_3(\{\mathbf{P}_i\}, \mathbf{M})$. On the basis of this evaluation, most of the hypothesised models are rejected. We retain the top 10% of results. A Levenberg-Marquardt [83] optimisation procedure is applied to optimise this likelihood function for the remaining hypotheses.

3.4.3.1 Optimising for 3D likelihood

Levenberg-Marquardt (LM) is a standard algorithm for solving non-linear least squares optimisation problems. LM interpolates two other common fitting techniques, namely gradient descent and Gauss-Newton. Gradient descent will converge to the local minimum for a given function. However, the method may take a long time to converge. Gauss-Newton will converge more rapidly if initialised close to the minimum, but otherwise may oscillate around this minimum. LM combines the desirable properties of both

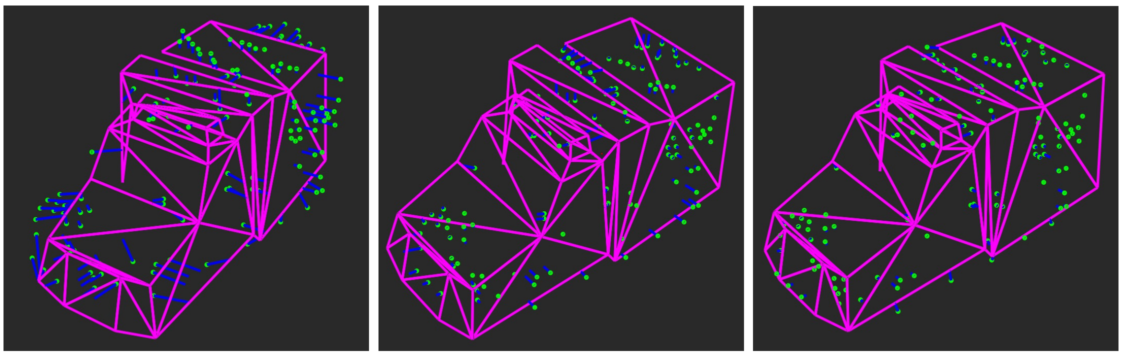


Figure 3.10: Optimising the 3D likelihood for an ambulance model. Blue lines show the distance from feature points to the model surface as the minimisation progresses.

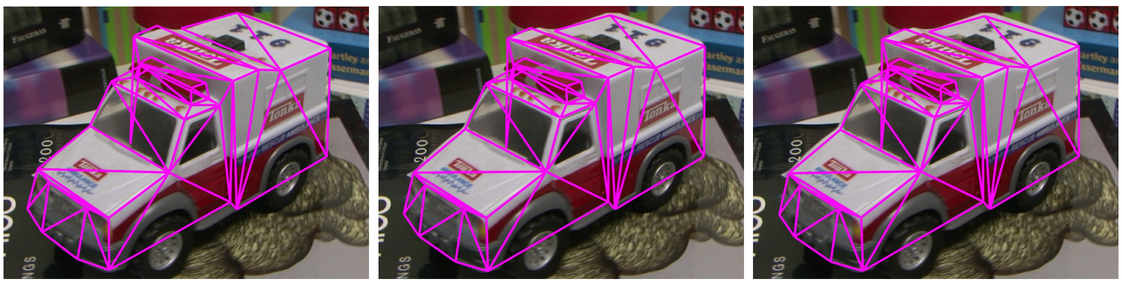


Figure 3.11: Optimising the 2D likelihood for the ambulance model

methods, by behaving like Gauss-Newton when close to the minimum, and behaving like gradient descent when farther away.

By applying optimisation to the likelihood function, we can find modes of $\Pr(D_3|MI)$. By generating a set of models spanning the parameter space, we aim to generate at least one model closer to the global minimum than any local minimum. After minimisation we remove models with parameters not significantly different from another hypothesised model. Models are rejected if the difference between all parameters and the parameters of a model with a greater likelihood score is $< 5\%$. This reduces the cost of evaluating multiple hypotheses that are likely to give the same final result. Figure 3.10 shows the result of optimising the 3D likelihood for the ambulance model.

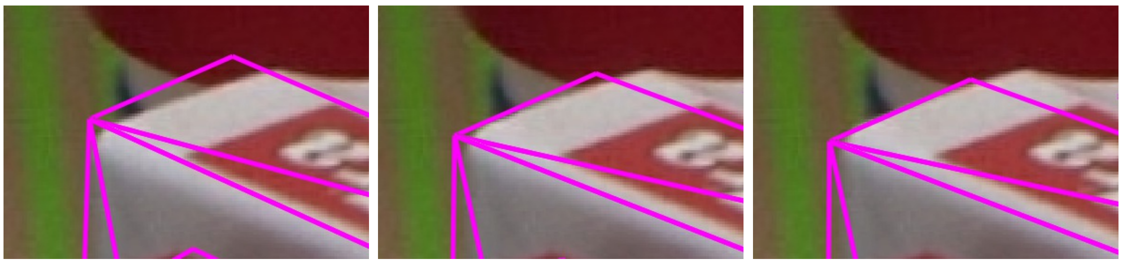


Figure 3.12: 2D optimisation results for one section of the model

3.4.4 2D likelihood evaluation

We now evaluate $\Pr(D_2|\mathbf{M}I)$ for the remaining hypotheses, using the likelihood function $\mathcal{L}_{2_p}(\{F_i\}, \mathbf{M})$. We retain the top 25% of these hypotheses. LM optimisation is then applied to optimise this function for the remaining hypotheses. LM optimisation is only capable of finding the local minimum of a likelihood function. The goal of the 3D likelihood evaluation and optimisation is to find a set of hypothesised models including at least one model for which the local minimum for the 2D likelihood function is also the global minimum. As the images provide stronger cues for the precise object location than the sparse point cloud does, $\Pr(D_2|\mathbf{M}I)$ is used to select the final model to present to the user. Figure 3.11 shows the result of optimising the 2D likelihood for the ambulance model. Figure 3.12 gives a close-up view of one region of the model as this likelihood is minimised, showing the effect of this minimisation on fine localisation as the photoconsistency of the model over the sequence is improved.

3.4.5 Hypothesis selection

At the completion of the fitting process, we again find models within the set with very similar parameters, and remove these duplicates from the set. The remaining models are sorted by their 2D likelihood, with the model with the greatest likelihood presented to the user. The user can view the other hypotheses by scrolling through the set with the mouse wheel. Here, we rely on the user to resolve the ambiguity in model fitting if more than one likely set of model parameters is found.

Typically, the top resulting model will closely fit the selected object in the image. However, erroneous results are possible, particularly if the object is significantly

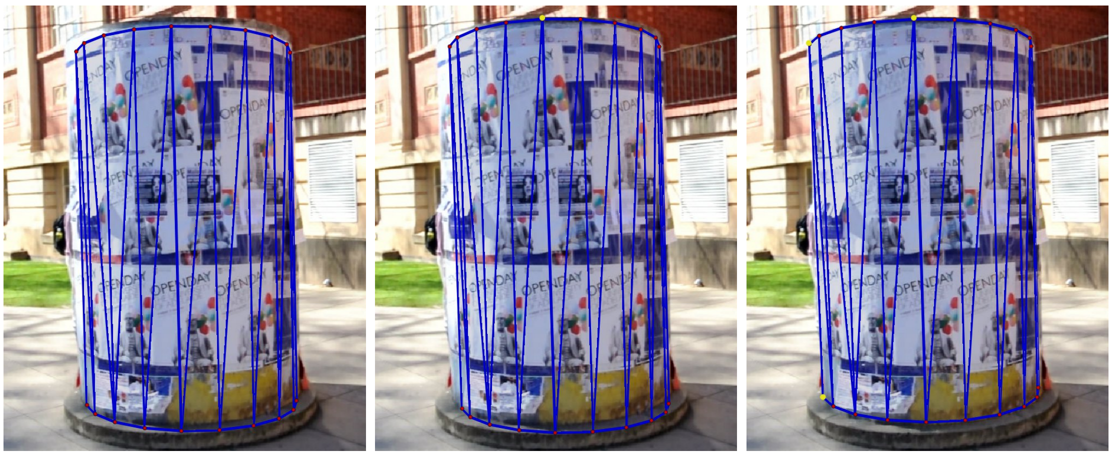


Figure 3.13: Manually refining the fit for the cylinder model. On the left, the initial fitting result. In the centre, the fitting result with one vertex specified. On the right, the fitting result with three specified vertices.

occluded in the images, if the image data is particularly noisy, or if appearance measurements are unreliable due to light changes, specular reflections, or other effects. Even with a close fit, the accuracy of this fit might not be sufficient for a particular application. To correct the result in such cases, the user is given the ability to further refine the model.

3.4.6 Interactive refinement

In the final stage of the fitting process, user interaction can be used to refine the fitting results. Interactive refinement allows the user to correct the result to the desired level of accuracy, if this has not been achieved by the automated process. The interactive process is intended to minimise the interaction required to achieve the desired result. Wherever possible the user interaction is performed directly on the frames of the sequence, which enables both interactive operations themselves and inspection of the results to be performed in the same space. As the images are the main reference that the user has for the correctness of a given model fit, directly correcting misalignment in the images is an intuitive interaction.

In the interactive refinement process, the user adjusts the fit of the model chosen in the hypothesis selection stage. To adjust the fit, the user selects an image from the set where misalignment is visible, and selects and drags a vertex of the model into the correct position. The model fit can be further refined by adjusting the vertex position in further images, and by adjusting positions for additional vertices. The user can continue refining vertex positions until the desired accuracy of fit has been achieved. During the interactive process, we optimise the model parameters to align the model with the user-specified vertex positions.

We combine the likelihood given user specified vertices with the 2D likelihood $\mathcal{L}_{2_p}(\{F_i\}, \mathbf{M})$, and optimise the summed likelihood by again applying LM optimisation. We give a high weighting to costs from $\mathcal{L}_U(\{\mathbf{u}_{ij}\}, \mathbf{M})$, with the user specified positions acting as soft constraints on the final result. If the user has specified at least as many vertex positions as there are model parameters, only $\mathcal{L}_U(\{\mathbf{u}_{ij}\}, \mathbf{M})$ is minimised. Once sufficient vertex positions have been specified to fully constrain the model, the process is similar to the procedure used for interactively specifying model parameters in systems such as ICARUS [1]. Figure 3.13 shows an example of manually refining results for the cylinder model. The image on the left of this figure shows the initial fitting result. The image in the centre shows the fitting result after a single vertex position has been specified by the user. The image on the right shows the fitting result with three vertex positions manually specified.

3.5 Experiments

Table 3.1 shows test results for 5 runs of fitting for four model types, giving the average error for vertices of the final selected models. This error, given in pixels, is measured as the ℓ^2 norm of the vector between a vertex of the fitted model, projected into a frame of the sequence, and the projection of the corresponding vertex for a manually refined model. We describe this measure as *image error.* Image error was averaged over each vertex and each frame of the sequences. The sequences were captured at a resolution of 1920×1080. The fitted and manually refined models are seen in Figure 3.14. The model types include cube and cylinder primitives, and two more detailed models. The cube and ambulance models were fitted using the single click initialisation, while the house and cylinder were fitted using outline initialisation. The ambulance includes large regions of uniform texture, giving rise to sparse features in the point cloud recovered by SfM, and is an example of an object which would be difficult to identify automatically from the 3D data. The third run of fitting the house model is an example where the best hypothesis as measured by the photoconsistency test has a relatively high image error, but a hypothesis with a better fit could be selected by the user from the sorted set of hypothesised models. Results for fitting the cylinder show an example where image error is relatively high due to limited views of the object resulting in high photoconsistency scores for a range of model parameters. In this case, further user interaction (shown in Figure 4.5) can be used to resolve the ambiguity.

Table 3.2 shows test results for the stages of the fitting process for one run of fitting the cube model, showing the improvement in image error at each stage in fitting the final selected model. *Initial* gives the error for the model with the original generated parameters, while *Optimised 3D* and *Optimised 2D* give the error after optimising the 3D and 2D likelihood functions. The fitted model at each stage is seen in Figure 3.15.

	Trial				
Cube	7.734107499	9.594787093	8.683456797	4.598463015	6.481200504
Ambulance	12.57431313	10.38260774	9.199748952	9.355773317	9.591450155
House	8.031003501	4.698100065	25.56713002	5.723756937	9.688623257
Cylinder	29.173147635	25.409166735	23.27648898	27.13983996	35.07249279

Table 3.1: Image error for a set of models

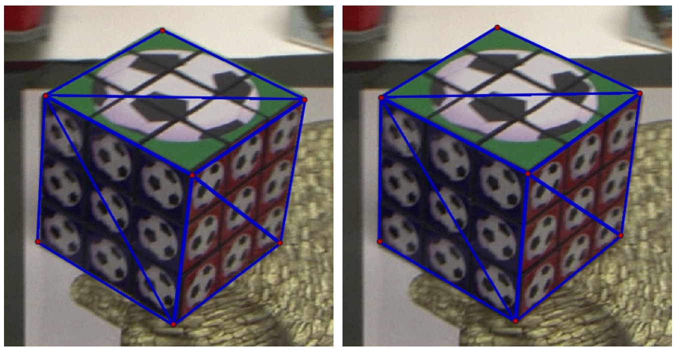
	Initial	Optimised 3D likelihood	Optimised photoconsistency likelihood
Cube	61.86901557	20.10094239	5.745042435

Table 3.2: Image error for stages of fitting the cube model

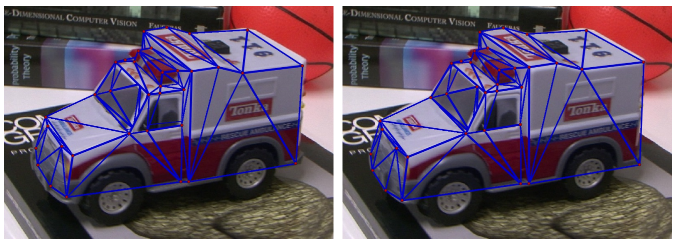
3.6 Summary

In this chapter, we have presented a method for fitting a model to images from a video sequence with user interaction. This method uses 3D point cloud and 2D image information to minimise the amount of user interaction required. We rely on a user to perform the task of identifying objects in the scene and roughly indicating their position or shape. While simple for a user, this task is difficult to perform automatically, particularly if features on the object are sparse, or if few training images are available.

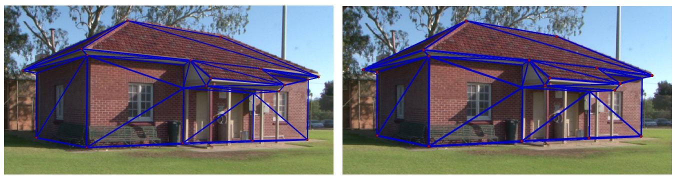
We use an automated process to align models with the images, which would require significant user interaction to perform manually. We then allow the user to refine fitting results to the level of accuracy required. We employ a hierarchical fitting procedure, only performing more computationally expensive image-based testing on hypothesised models with high scores from less expensive tests on point data. In contrast to other interactive methods using primitive models, the fitting process does not require the user to specify sufficient constraints to determine all model parameters. Our method does require existing models representing the geometry in the scene, and in the following chapters we describe a method for constructing such models from video sequences.



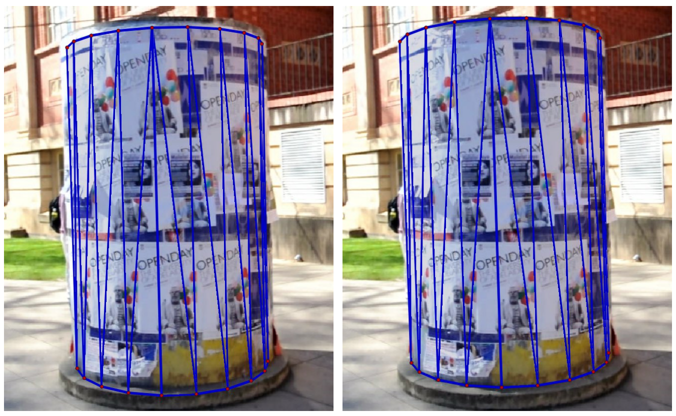
(a)



(b)



(c)



(d)

Figure 3.14: Test results for model fitting. Images on the left show fitting results. Images on the right show manually refined results.

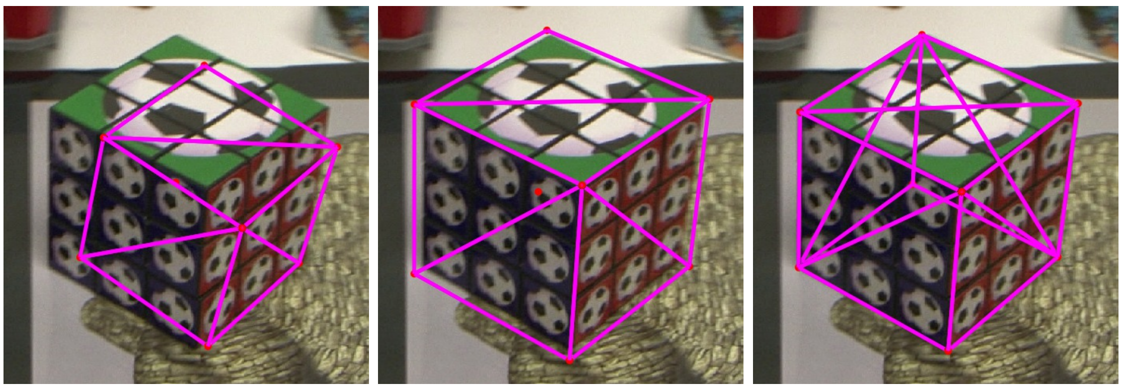


Figure 3.15: Test results for stages of fitting the cube model

4

Modelling with User-Defined Geometry

In the previous chapter, we presented a method for fitting pre-defined 3D models to image and point data with user interaction. These models can include both primitives and more detailed shape types. While primitives are suitable for modelling much of the basic geometry of man-made environments, these environments will also typically include elements which cannot be realistically represented by simple geometric types. For example, in architectural scenes, the basic geometry of many buildings can be captured by cuboid types. However, specific details of building facades will often include more complex geometry.

A general-purpose approach to modelling requires a means of modelling geometry which cannot be easily represented by pre-defined primitive shape types. One solution is to provide a large set of more detailed model types. However, to model all geometry that could be found in a typical scene, a very large set of models would be required. This also requires a method of obtaining the original models.

An alternative approach to providing a large library of model types is to allow the user to define the geometry. This approach allows elements of a scene to be modelled regardless of whether a corresponding pre-defined type exists. User-defined geometry could also be used to create new pre-defined shape types, which could then be applied to modelling further scenes.

In the following chapter, we describe an approach to interactively modelling objects in a video sequence. To make the modelling process rapid and intuitive, geometry is defined using a sketch-based interface. To minimise the amount of interaction required, where possible interaction is only performed in a single frame of the video. We focus on the problem of recovering 3D structure from a set of 2D lines traced over an object in one image.

Sketch-based interfaces for 3D modelling, such as those described in [55] and [57], allow users to rapidly generate 3D models from simple 2D interactions, without requiring the specialised knowledge of the modelling system typically needed to successfully construct models with traditional systems such as 3ds Max [54]. To infer 3D structure from 2D interactions, these systems either map 2D gestures to a pre-defined 3D axis [55], or infer a model from a 2D contour, using assumptions such as that the desired model can be generated by inflation of a flat surface matching the contour [56] [57]. We resolve the ambiguity of a 2D sketch by using point cloud and image data to provide the missing depth information.

By modelling from user-defined geometry, we can create models which satisfy the user's requirements, modelling the required geometry to the desired level of detail. From the sketch describing an object, we generate a set of planar facets. Knowing the facets that an object is composed of, and the projection of those facets into one frame, we can recover structure which is difficult or impossible to reconstruct with an automated technique. In particular, we can model structure in regions for which texture detail is sparse. With the boundary of a plane defined in one view, determining depth for all points on that plane only requires knowing the depth of sufficient points to define the plane parameters. We can aid in fitting facets to regions where features are extremely sparse or noisy by using the constraints provided by adjacent facets fitted to regions with greater feature detail to reduce the number of points required to define the parameters of the plane. User-defined geometry can also be used to model regions of the scene which are partly occluded, and regions where the user is aware of fine detail which cannot be recovered automatically due to the limited resolution of the sequence. Where needed, fitting results can be corrected and improved by the user specifying additional constraints in multiple frames.

The model fitting method described in the following sections builds on the method described in Chapter 3, and can be used as a way of constructing initial model instances which can then be fitted to subsequent scenes using the previous method. We again make use of camera parameters and point data recovered with SfM. From boundaries defined with a sketch-based interface, we use the point cloud and any existing model geometry to generate a set of hypothesised models, and select the model which best fits the data with a hierarchical fitting procedure using both point cloud and image information.

The primary novelty of this method is in allowing the user to define and fit a model with intuitive sketch-based interactions performed in a single frame. Systems such as PhotoBuilder [3] allow the user to construct a mesh model of the scene, but require the user to specify vertex locations in multiple frames, or rely on feature points being automatically matched at the desired corner locations. The method of [41] can be used to build models from a single image, but requires the specification of additional constraints such as parallelism and perpendicularity. The modelling method of [4] allows for model construction with single frame interaction, but is designed for reconstruction of architectural scenes, and assumes that significant vanishing line geometry can be found in the scene. The authors of [4] believe that their system is better suited to reconstructing architectural scenes than the VideoTrace [84] system which forms the basis for the work in this chapter, while VideoTrace is better suited to reconstruction of free-form shapes.

The method of [50] allows for reconstruction with painting interactions in a single frame. While this method has the advantage of recovering fine surface detail, it does not allow the user to define the structure of the resulting model, which consists of a uniformly dense mesh. This method also assumes that the surface can be fully reconstructed on the basis of photoconsistency, and does not allow the user to modify the model beyond adjusting the position of an initial surface patch.

In Section 4.1 we describe our interactively specified model type. Section 4.2 describes the method for fitting a single specified face, the likelihood functions used in the fitting procedure, and the method of selecting frames used by the appearance-based likelihood function. Experimental results for this fitting method are presented. We describe a method for jointly fitting sets of connected faces in Section 4.5. As the density of an SfM reconstruction will vary with the amount of texture present, we expect that in some regions of the scene the point cloud will be too sparse for accurate fitting using only point data. Applying fitting to a group of connected faces can allow fitting results for regions with dense features to inform the fitting for regions with sparse features. To fit single faces in regions of the scene with sparse features, Section 4.6 gives a further method of generating and evaluating hypothesised faces, based on sampling planes at a range of orientations and with depth sampled based on the range of depths of nearby features.

Finally, Section 4.7 gives detail on the techniques we use for interactively specifying the boundaries of model faces in frames of the sequence. This includes a sketch-based interface for intuitive interaction, and tools for refining existing faces. Some material in this chapter on the model fitting process and the interface for specifying models was originally presented in [84]. The methods described in this chapter were developed and implemented by the author with input from his thesis supervisors.

4.1 Specifying 2D shape

The interface described in Chapter 3 was conceived with the goal of obtaining the maximum information from the minimum user input. While modelling with user-defined geometry requires greater input from the user, we similarly aim to allow the user to model an object without providing more information on the structure of the model than is necessary. Towards this goal, we again restrict user interaction to 2D operations performed on frames of the sequence, with the user specifying the appearance of the geometry in one frame, and the corresponding 3D shape being determined from the 2D and 3D data for the sequence. The interaction used to specify a face is designed to be intuitive for novice users. A polygonal mesh representation is again used for the modelled geometry. This representation is well suited to modelling much of the geometry typically encountered in manmade environments.

To construct 2D polygonal representations of objects in the scene, the user specifies one or more faces, consisting of vertices and edges. This specification can either be performed directly, by the user individually specifying each element of each face, or by the user providing a freehand sketch representation, which requires further processing to convert it to a polygon mesh form. Figure 4.2 shows a face of the model being constructed from a set of user-specified vertex locations. Details of the interactive process for specifying faces are given in Section 4.7. We first introduce the bounded plane model type representing user-specified faces, and the process for fitting such models.

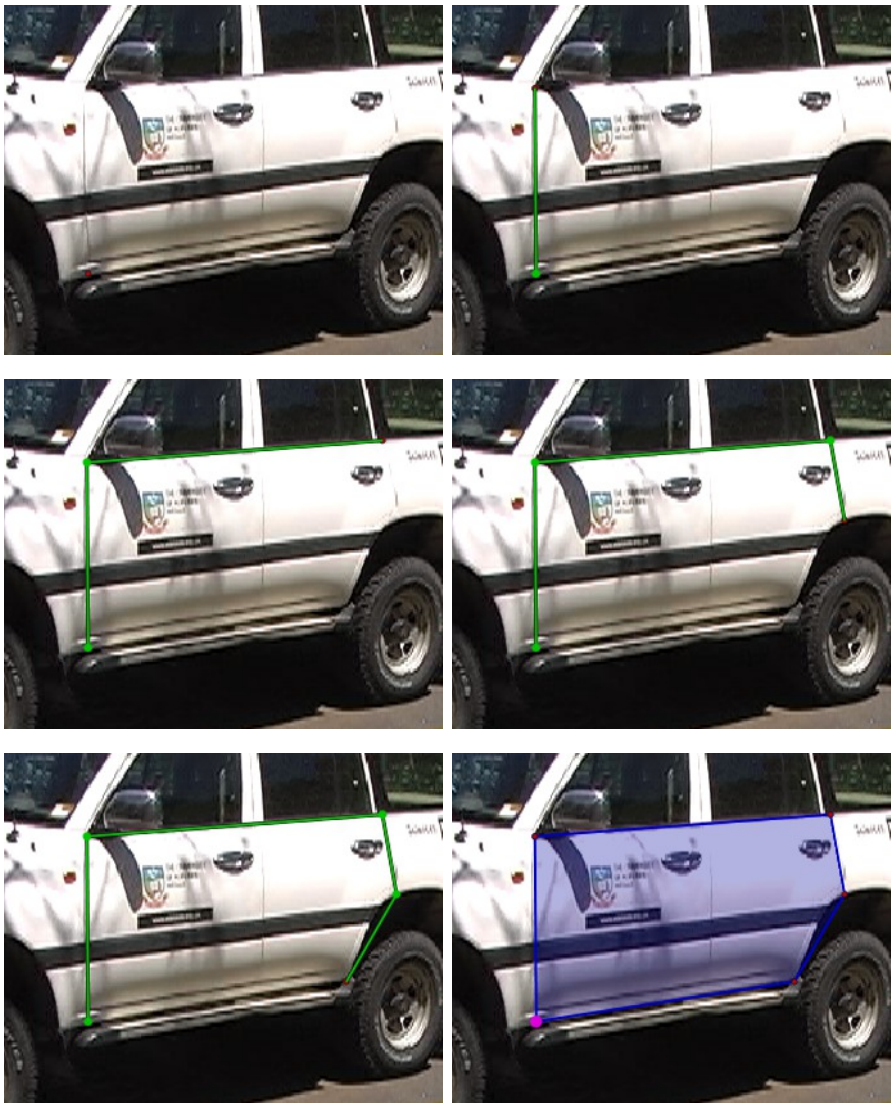


Figure 4.1: Specifying shape with a drawing interaction

4.1.1 Bounded plane model

The bounded plane model is a flexible model type which can be used to represent planar 3D shapes. An instance of the bounded plane model is defined by rotation parameters $\mathbf{R} = (R_0, R_1)$, used to specify a plane normal vector $\mathbf{N}$, and by a position parameter C, specifying the distance from the camera to the point on the plane corresponding to the centroid of the 2D polygon. Each plane model has a boundary defined by a set of 2D vertices $\{\mathbf{u}_i\}$, specified in an image from the sequence. For a given plane model, a set of 3D boundary vertices $\{\mathbf{U}_i\}$ is generated by back projecting the vertices of $\{\mathbf{u}_i\}$ onto the plane.

4.2 From 2D sketching to 3D models

Converting a 2D polygonal model to a 3D polygonal model requires determining a 3D position for each 2D vertex. Under the assumption that each face of the model corresponds to a planar face in the world, vertex locations are given by projecting each vertex onto the plane corresponding to the face containing that vertex. The problem, therefore, is to determine the plane parameters for each face of the model. If a model consists of multiple connected faces, the parameters of the plane defined by a face will be dependent on the parameters of the planes defined by neighbouring faces.

The model fitting method in Chapter 3 was designed to be general enough to be applicable to a range of model types. While the model types considered in the previous chapter were pre-defined, the method described is also applicable for models defined by the user. By treating user-defined planes as another model type, this method can be applied for modelling with user-defined geometry.

To apply the previously described model fitting process to a new model type (in this case, the bounded plane), we determine appropriate 2D and 3D likelihood functions for this model type. As before, these functions take advantage of both 3D point and 2D image information, and information supplied by the user. In this case, the user supplied information consists of one or more 2D faces, with edges corresponding to the boundaries of planar faces of the object. The fitting process first evaluates and optimises a set of hypothesised models on the basis of relatively inexpensive tests on the 3D data. The best models from this stage are then tested and optimised using 2D image information, with the best model given the image information added to the set of face models generated so far.

Algorithm 2 describes the fitting process for n hypothesised planar models, generated from a set of user-supplied boundary points $\{\mathbf{u}_i\}$ and any existing model geometry $\mathbf{M_c}$. The models are evaluated with a set of m functions $\{\mathcal{L}_i(D_i, \mathbf{M})\}$ used to evaluate the likelihood of model $\mathbf{M}$ given data D_i. $\{f_i\}$ gives the fraction of the hypothesised models remaining at test stage i.

Algorithm 2 User-defined model fitting

```
Generate a set of hypothesised planar models $\{\mathbf{M}_j\}$ from the sequence data $D$,
boundary points $\{\mathbf{u}_i\}$, and the model already defined $\mathbf{M}_c$
for $i = 1$ to $m$ do
  for $j = 1$ to $f_i n$ do
    Evaluate $\mathcal{L}_i(D_i, \mathbf{M}_j)$, optimising if required
  end for
  Sort $\{\mathbf{M}_j\}, j = 1 \ldots f_i n$ by $\mathcal{L}_i(D_i, \mathbf{M}_j)$, in ascending order
end for
Add the geometry of the best model $\mathbf{M}_1$ to $\mathbf{M}_c$
Optimise $\mathcal{L}_U(D_U, \mathbf{M}_c)$ for additional user-supplied point data $D_U$, if required
```

4.3 Single face fitting

We will first consider applying fitting for an individual face, then consider applying fitting to multiple faces. The following section describes the process of fitting an individual model face to the image set. Figure 4.2 shows an example of a face which has been fitted to the images.

4.3.1 Generating hypotheses

Given the 2D face supplied by the user, we generate a set of possible corresponding 3D models. For a model consisting of a single face, each model is generated by projecting the 2D points supplied by the user onto a hypothesised plane on which the face could lie. To generate these hypothesised planes, we follow the method of RANSAC [8] plane fitting. To generate a set of possible planes given the data, we generate planes on the basis of feature points within the user-supplied boundary. Feature points derived from

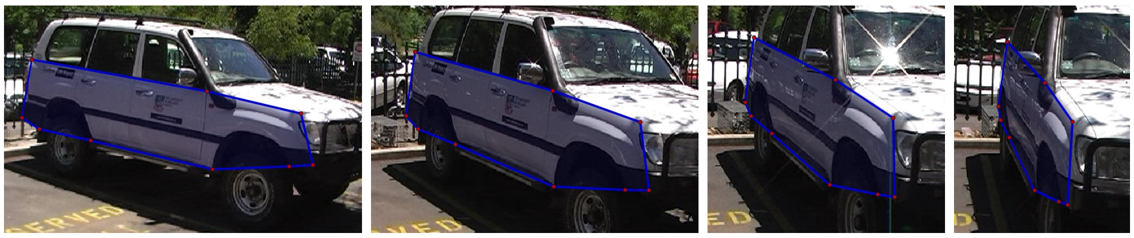


Figure 4.2: Model face fitted to the images

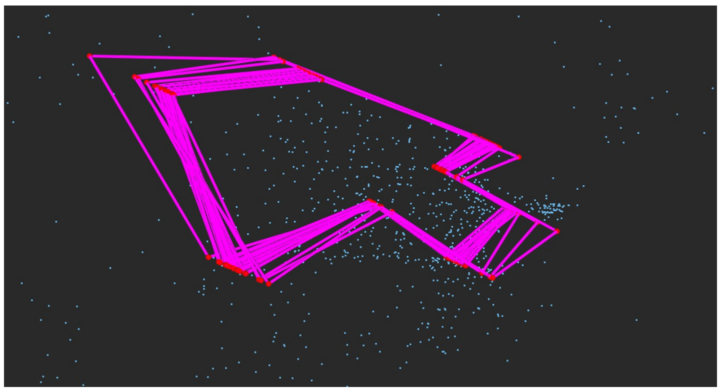


Figure 4.3: A selection of the hypotheses generated for a model face

features of the object are expected to correspond to points on the object surface. In contrast to the method for fitting general models described in the previous chapter, the parameters of a planar face model can be determined directly from a set of points on the face. A face generated from a set of inlier points is expected to be close to the optimal model parameters. For a sufficient proportion of inlier points, sampling plane parameters from the point data requires significantly fewer hypothesised faces to be generated to find one close to the true model parameters than is the case for sampling directly from the space of plane parameters.

The number of points required to generate a plane will depend on the number of points in the corresponding face which already have a defined 3D position. Three points are needed to specify a plane. A face with no pre-defined point positions will require three feature points. A face attached to an edge of the model with defined 3D positions for both vertices will only require one additional feature point.

Planes are generated from selected sets of three points. Sets of points are selected randomly, without replacement. In the initial plane generation, plane normals are set to face the camera for the initial frame, as we assume that the user is modelling visible geometry. Sets of points are rejected if the points are collinear, as in this case the plane will be undefined. Figure 4.3 shows some of the planes generated for a face defining part of the roof of a building.

4.3.2 Selecting feature points

Generating a set of hypotheses requires first selecting an appropriate set of feature points. Feature points are selected from the set of features with a corresponding point in the frame where the 2D model was constructed. This limits possible points to points that are visible in this view. If features from the complete set were used, the set could also include points from the hidden side of the object being modelled, or points from other objects behind the object being modelled. Points are selected by finding those with a projection into the current frame within the boundaries of the user-specified 2D face.

4.3.3 3D likelihood function

As with the model fitting method described in Chapter 3, the goal of 3D fitting is to determine the parameters for a model which are most probable given the available 3D point cloud data. We again employ a coarse-to-fine fitting strategy to aid in performing fitting at interactive speed. As determining the quality of fitting results based on sparse 3D data is typically significantly less computationally expensive than determining the quality of the fit given dense 2D image data, 3D fitting is again used as a first stage of the fitting process, after which candidates with low likelihood can be rejected before more costly 2D tests are performed.

We perform a robust likelihood evaluation for the models given the 3D data. We anticipate that the set of features within the region specified by the user may include features for which the SfM process failed to correctly estimate the 3D position, due to errors in identifying the point over multiple frames, or insufficient baseline between views of the point. The region specified by the user will also not necessarily correspond to a true plane. It may, for example, be a curved surface which is close to planar, or a largely planar region which includes some protrusions. In this case, although the 3D position of the points may be correctly estimated, some points will not lie close to the best-fit plane for the specified region.

For each candidate plane, we evaluate the 3D distance from feature points to the closest point on the model surface, as described in Section 3.3.1. We could alternately evaluate the 2D distance between feature points in a set of frames and the corresponding closest points on the plane projected into those frames. However, this is more expensive to evaluate, requiring multiple projections and distance computations for each point, and we found the less expensive 3D distance measure to give acceptable results.

After evaluating 3D likelihood, we select a set of the best hypothesised faces for further evaluation. Faces are selected using the count of inliers from the selected set of features. Inliers are determined using the robust measure of standard deviation described in Section 3.3.1. The 10% of models with the highest inlier counts are retained. Optimisation is then applied to the parameters of these models, using the robust 3D likelihood function $\mathcal{L}_3(\{\mathbf{P}_i\}, \mathbf{M})$, evaluated for the set of inlier features.

4.3.4 2D likelihood function

For a given hypothesised face, the 2D likelihood function determines the likelihood of the appearance of its projection into the images. Evaluating this function requires first selecting a set of images where the hypothesised face is visible.

4.3.5 Frame selection

For appearance comparison, we select frames in which we can compare the appearance of the hypothesised faces with the appearance of the face in the initial frame. We use a single set of frames for all hypotheses, as this gives a consistent basis for comparisons between hypotheses. For appearance comparison to be informative, we should select frames where the camera is sufficiently far from its position in the initial frame for misalignment to be visible, while avoiding frames where the face is outside of the frame, facing away from the camera (or at an angle where details are obscured), or largely occluded.

Selecting these frames requires estimating the visibility of the face in the frames, before the face has been fitted. For this we estimate the visibility of the face using the visibility of the set of feature points inside the boundary specified by the user. If a feature point has an associated correspondence in a given frame, this indicates that

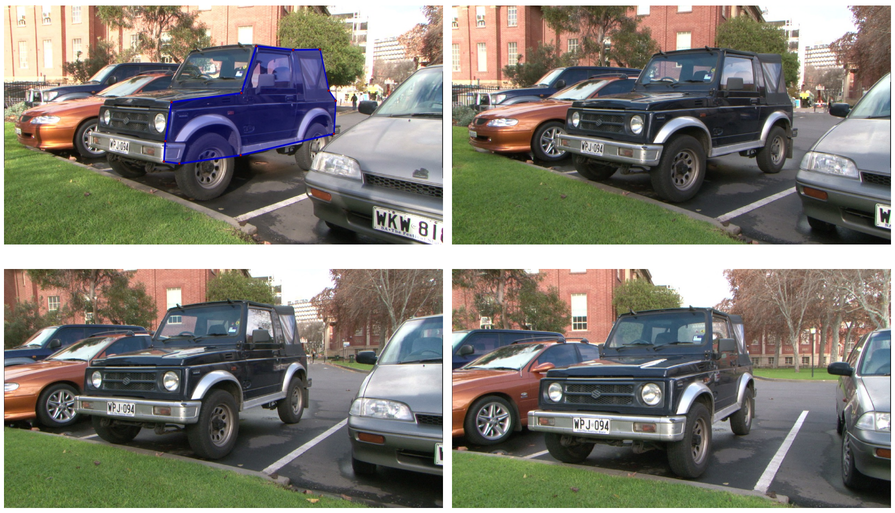


Figure 4.4: Frame selection: the sketch in the initial frame, and the set of selected test frames.

a match for that feature was found by the tracker in that frame. From this visibility estimate, we select a set of candidate frames.

This set may be too large to evaluate the appearance of each hypothesised model in each candidate frame at interactive speed. Using all candidate frames is unnecessary, however, as for a video sequence we expect neighbouring frames to contain much the same information. Rather, we select a representative subset of the candidate frames, spanning the range of views from which the face is visible.

We use the count of correspondences in each frame of the sequence as an estimate of the visibility of the face in that frame. We do not, however, limit the set of candidates to frames which have correspondences for all features. A feature lacking a correspondence in a frame does not necessarily mean that the corresponding point on the object is not visible in that frame. Features may not have correspondences in a frame due, for example, to temporary occlusion or appearance variation due to lighting making the feature difficult to track over the sequence. As such, we allow some tolerance for missing correspondences. The set of frames with correspondences for $> 75\%$ of the features is selected as a set of candidate frames for appearance comparison.

Feature point visibility is similarly applied for view selection in [4], with views being selected based on the count of points in a frame with a match in the frame where a face was defined. As this method is intended for reconstruction from photo collections, it can be assumed that this will provide views of the object from substantially different perspectives. As our method is intended for use with video sequences, selecting views based only on the count of matching features would give preference to views very close to the initial frame, yielding insufficient baseline for useful photoconsistency comparison. Rather, we iteratively select frames from the set of candidate frames based on the divergence from the frames selected so far. We measure divergence with $a(F_i, F_j)$ giving the angle between a ray from the camera centre for the frame F_i to the centroid of the set of 3D features within the specified face, and the corresponding ray for frame F_j. This selection technique gives a set of frames spanning the range of views from which a plane is visible, maximising the divergence between each view. As this technique does not rely on any ordering of the frames, it is also suitable for sequences consisting of an unordered image set. For m test frames, the set of test frames $\{T_j\}$ is selected as described in Algorithm 3.

Algorithm 3 Frame selection

```
Determine a set of candidate frames $\{C_i\}$ with correspondences for $> 75\%$ of features
in the initial frame
Add the initial frame to $\{T_j\}$
**while** $|T| < m$ and $|C| > 0$ **do**
  Sort $\{C_i\}$ by minimum divergence from a frame in $\{T_j\}$, in descending order
  Remove $C_1$ from $\{C_i\}$ and add to $\{T_j\}$
**end while**
```

4.3.6 Appearance testing

For the final selection and optimisation of the best hypothesised face, we perform appearance-based comparison and optimisation. This can improve fitting results compared with fitting using only the point cloud data, as the images give information for the quality of the fit over the complete surface, rather than for a sparse set of points on the surface. For appearance testing we use a metric similar to the metric described in Section 3.3.2.1. This metric was used to evaluate the photoconsistency of a hypothesised model with unknown appearance. Photoconsistency was determined from the variance in the RGB values of a set of sample points on the model surface. In contrast to the unknown appearance of the models in the previous chapter, the user-defined boundary gives a known appearance for the model in the initial frame. This allows us to directly compare the appearance of the face in other frames with its appearance in the initial frame. Photoconsistency likelihood for a bounded plane is defined as

$$\mathcal{L}_{2_b}(\{T_j\}, \mathbf{M}) = \frac{f_{2_b}}{n} \sum_{i=1}^{n} \sum_{j=2}^{m} \frac{(T_1(\boldsymbol{A}_{T_1}\mathbf{S}_i) - T_j(\boldsymbol{A}_{T_j}\mathbf{S}_i))^2}{m-1} \tag{4.1}$$

where $\{\mathbf{S}_i\}$ is a set of n point samples from the model surface, $\{T_j\}$ is the set of m test frames, $\boldsymbol{A}_{T_j}$ is the camera matrix for test frame j, $T_j(\mathbf{a})$ gives the pixel colour for point $\mathbf{a}$ in test frame j, and f_{2_b} is a constant scale factor.

To evaluate the likelihood, we first select a set of point samples for the face. To generate a set of samples from the region specified in the initial frame, we first fit a 2D bounding box around that region. Within this bounding box, we evenly sample points in 2D, and add all points within the boundary of the face to the sample point set. This gives a consistent set of samples for all hypothesised faces. The 3D position of each sample point for a hypothesised face is determined by back projecting the 2D point onto the corresponding plane.

Sample points could alternately be generated by sampling evenly on the 3D surface of each hypothesised face. This was the method used for models in the previous chapter. This, however, would mean that the photoconsistency of each model was evaluated for a different set of 2D points in the initial frame. This could significantly influence the result of the evaluation, particularly as the variation in the orientations of the hypothesised planes would affect the density of sample points in different regions of the face in the initial frame.

2D likelihood functions are evaluated on images downsampled to a width of ≈ 180 pixels using area-weighted downsampling. By removing fine detail and noise, downsampling reduces the effect of local minima on the optimisation of the 2D likelihood. While the same effect can be achieved by evaluating on smoothed versions of the images, operating on downsampled frames can also significantly reduce the memory required to store the set of frames used.

The 2D likelihood function is evaluated for each hypothesised face remaining after the previous evaluation from 3D points. The 25% of faces with the highest likelihood are retained and optimised, again using LM optimisation. The remaining face with the highest likelihood given the image data is selected and further optimised before being added to the model.

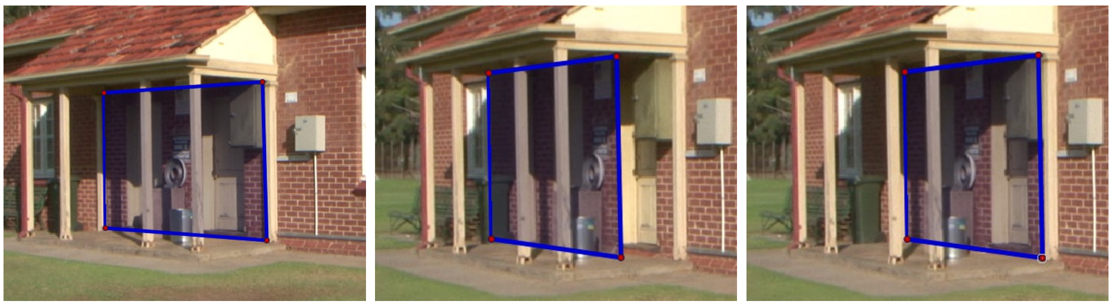


Figure 4.5: Manually refining a face. On the left, the face drawn by the user. In the centre, the resulting fit, seen from another view. On the right, the fitting result after manual refinement

4.3.7 Optimising the selected face

The 2D likelihood evaluation performed on downsampled frames gives an initial localisation of the selected face. To further refine this fit given all available image data for the test frames, optimisation of the 2D likelihood is iteratively applied using images of increasing resolution stored in an image pyramid. Images in each level of the image pyramid have twice the resolution of images in the previous level, and the final optimisation is performed using the original unscaled frames. Initialising the model parameters at each level of the pyramid from the optimal parameters for the previous level reduces the chance of the fitting process falling into a local minimum.

4.3.8 Manually refining fitting results

As in the model fitting method described in the previous chapter, the results of this fitting process can be manually refined with additional information supplied by the user. The user can refine faces by specifying vertex locations in additional views. Refinement is performed by minimising the likelihood function for user-supplied data $\mathcal{L}_U(\{\mathbf{u}_{ij}\}, \mathbf{M})$ described in Section 3.4.6.

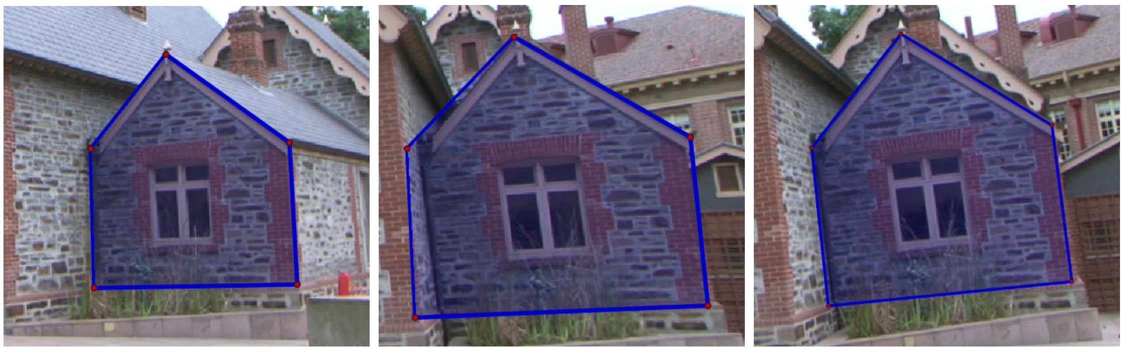
Manual refinement can also be used to achieve acceptable results if the fitting process could not find a face fitting the data. For some regions of a scene, insufficient views will be available for fitting to be performed successfully. An acceptable fitting result may also not be achieved for regions which are significantly occluded for some portion of the sequence.

If the fitting process is unable to fit a specified face, that face is still added to the model, but is rendered with a red outline, indicating that the fitting process failed. We set initial parameters for the face, which can then be modified by the user adjusting the vertex locations. The orientation of the face is set to be parallel to the image plane of the initial frame, and positioned so that the centroid of the set of features used in generating the plane hypotheses lies on the plane. We expect this to give a face with vertices relatively close to the true vertex locations. The user may accurately position the face by specifying vertex locations for at least three vertices in at least one additional frame. Vertex locations may alternately be specified in a rendered view of the scene with a camera position selected by the user, showing the point cloud and the model specified so far.

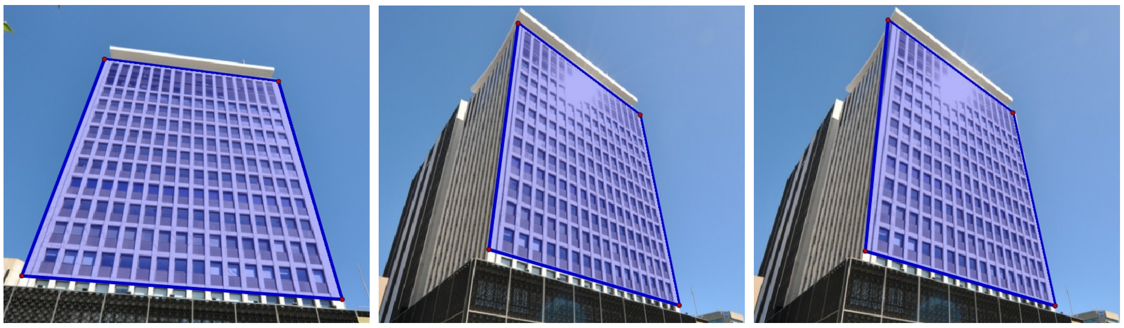
Figure 4.5 shows an example of manually refining a face when the fitting result is inaccurate due to occlusion. The image on the left shows the face specified by the user. The image in the centre shows the resulting fit from another view. The image on the right shows the manually refined face.

4.4 Experiments

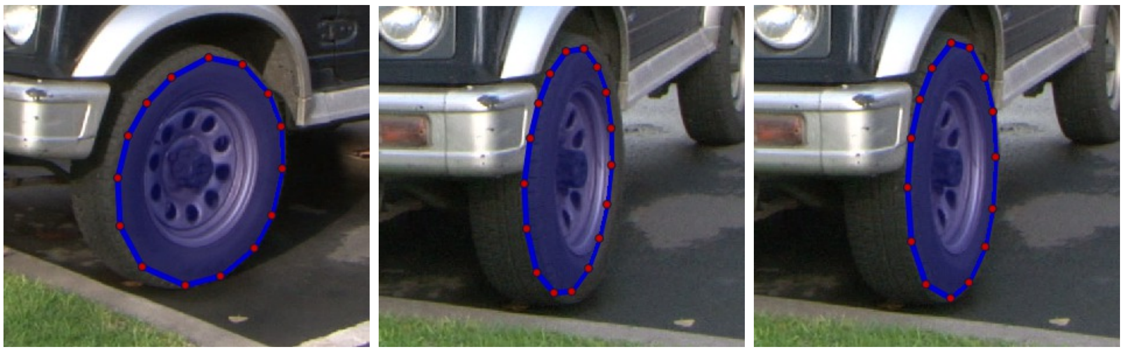
Table 4.1 shows test results for 5 runs of fitting for a set of faces. Rows labelled *3D* give the error for faces fitted using only 3D features, while rows labelled *Photo* give results for faces subsequently refined using the photoconsistency likelihood. Error was again computed for reprojected vertex locations against manually refined vertex locations, using the measure of *image error* defined in Section 3.5. The fitted faces are seen in Figure 4.6. Results for all faces are close to the optimal face as manually specified by a user. In most cases, fitting results were improved by optimising photoconsistency. However, *Wall 2* gives an example of a face for which better results were obtained by only fitting to 3D points, due to the reflective surface of the building not satisfying the assumptions of photoconsistency. *Wheel* gives an example of a face where the error when fitting only to 3D points is relatively large, as the surface is not truly planar, and features are only found for one region of the surface. The error is reduced by fitting using the photoconsistency likelihood, as this measure considers the complete surface.



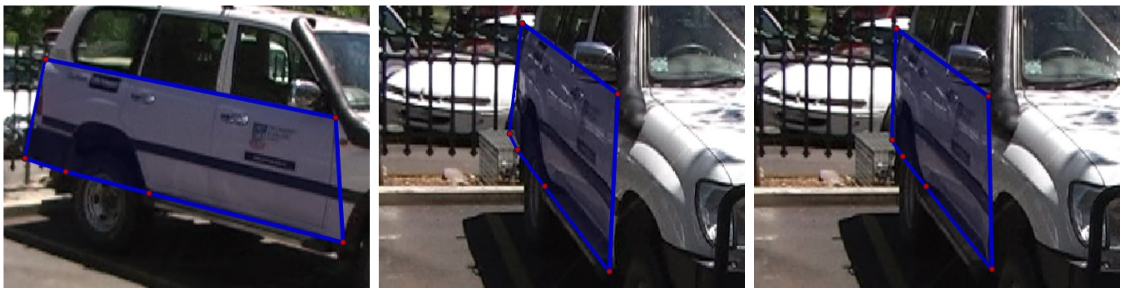
(a)



(b)



(c)



(d)

Figure 4.6: Test results for face fitting. (a) Wall1 (b) Wall2 (c) Wheel (d) Car. Images on the left show the face specified by the user. Images in the centre show fitting results using the 3D likelihood. Images on the right show fitting results using the photoconsistency likelihood.

	Trial				
Wall 1 3D	2.673519219	4.285908628	2.896498638	3.560933582	3.447250409
Wall 1 Photo	1.117254192	2.592815362	2.26130659	2.001052414	2.235406928
Wall 2 3D	0.710681435	0.958665312	0.846690661	1.036445027	0.592759046
Wall 2 Photo	3.769922496	4.408817418	3.986664597	4.948889677	4.077994916
Wheel 3D	14.20578528	13.01682798	13.15488077	11.31634984	11.27753697
Wheel Photo	4.433451548	4.582046999	3.948425967	3.504700793	4.992471418
Car 3D	3.538321613	4.914470836	5.420591414	4.565600069	5.804872735
Car Photo	3.749245517	4.317802066	3.03609876	4.356960821	4.449852883

Table 4.1: Image error for faces using the 3D and photoconsistency likelihoods

4.5 Fitting for multiple faces

The previous sections described a fitting process operating on single faces, with each face being fitted individually. The modelling interface that we describe in Section 4.7 allows a model to either be generated from a sketch of the complete model, or generated by specifying each face individually. If each face is added individually, the fitting result for a newly added face attached to an existing face will be constrained by the fitting result for that previous face. As such, with each face being added individually, the results of the fitting process will depend on the order in which faces were specified. This has a potentially significant effect on the quality of the resulting model. Features may not be evenly distributed over the surface of an object. For regions of the object with minimal texture, the SfM process will recover sparse features. These sparse features may not correspond to points on the dominant plane in that region of the object, giving low accuracy for a face fitted to those points. An inaccurately fitted face in a low detail region may limit the accuracy of the fitting result for adjacent faces added subsequently. Conversely, the constraints provided by an accurately fitted face in a region with high detail limit the number of inlier features needed to accurately fit an adjacent face in a region with lower detail.

To allow good fitting results for faces in high detail regions to inform the fitting of faces in regions with less detail, avoiding a dependency on the order in which a model is constructed, the model fitting process can be applied simultaneously to sets of faces. This allows for construction of a model based on an initial sketch consisting of multiple 2D faces, and for improving the fit of a progressively constructed model given the additional information provided by newly added faces. In fitting for sets of connected faces, the same likelihoods are evaluated as when fitting for individual faces, with likelihood values being summed over the total set of faces. However, the dependency between faces must be considered in generated hypothesised models from the point cloud and user-supplied data.

4.5.1 Generating hypotheses with multiple faces

Generating a hypothesis for a single face requires selecting three points from the set of feature points and existing vertices. Generating hypotheses for multiple faces is more complex, due to the dependencies between faces.

To generate hypotheses for multiple faces, while observing the dependencies between faces, each hypothesis is constructed progressively. An initial face is selected, and a plane is assigned as when generating a hypothesis for a single face. Vertex positions are set for any undefined vertices of the face. A second face attached to the first is then selected, and a plane is assigned to this. In assigning a plane to the attached face, the hypothesised vertex positions of the first face are used. If the second face has no pre-defined vertices, only one feature point is required.

We then proceed to generate hypotheses for any remaining faces attached to the first face, in the same manner. Once all faces attached to the first have had planes and vertex locations assigned, a new face attached to one of the newly defined faces is selected, and generation continues until all faces have been defined. Restricting the selection of subsequent faces to those which are attached to an already defined face enables faces fitted to regions with high detail to assist in fitting faces to regions with less detail. As the resulting hypothesised models will depend on the order in which faces are assigned, this ordering is randomised, as is the selection of feature points.

If faces of the model have been manually refined, the vertex locations specified by the user are assumed to be correct. These vertices are treated as constraints in the fitting process, and their locations are not modified when applying fitting to multiple faces.

The ordering applied for generating each hypothesised model can subsequently be used in the likelihood optimisation applied in the fitting process. This optimisation requires the selection of a set of free parameters sufficient to define the location of each model vertex. An appropriate parameterisation can be defined by first allowing all plane parameters for the first face selected in generating a hypothesis to be free. Following the order in which faces were defined for the hypothesis, a free parameter can then be added for each attached face, until the set includes sufficient parameters to fully define the model.

Figure 4.7 demonstrates the results of applying fitting to a selected set of faces. 4.7 (a) shows a face specified in a region with minimal features. Fitting results for that face are seen in 4.7 (b). 4.7 (c) shows additional faces connected to the original misaligned face. Fitting results for this set of faces, with fitting applied individually to each face in sequence, are seen in 4.7 (d). The fitted faces are also significantly misaligned, due to being constrained by the first face. 4.7 (e) shows the significant improvement in the result achieved by collectively fitting the set of faces.

4.6 Fitting with sparse points

In regions of a scene with low texture detail, the SfM process will typically only be able to recover a sparse set of feature points. In such a region, the number of points found within a user-drawn boundary may be insufficient to generate the correct plane, even with the constraints provided by fitting adjacent faces. We define a region with sparse points as any region for which fewer than a threshold of $k = 10$ points are found within the plane boundary. In this section, we describe a further technique for fitting single faces in such cases. This is a more challenging problem than fitting with a large number of features available, as we cannot reliably generate candidate planes from sampled features, or use the set of features to estimate visibility. Rather than sampling planes using sets of feature points within the plane boundary, we sample planes over a depth range determined from features within and near to the boundary. Sparse features present a challenge for the frame selection method described in 4.6.2, and we present an alternate frame selection method which does not rely on features.

4.6.1 Generating hypotheses with sparse features

With insufficient features to generate the correct plane by sampling sets of points, we sample directly from the space of possible model parameters. This sampling strategy was also applied for model fitting in the previous chapter. The number of sampled

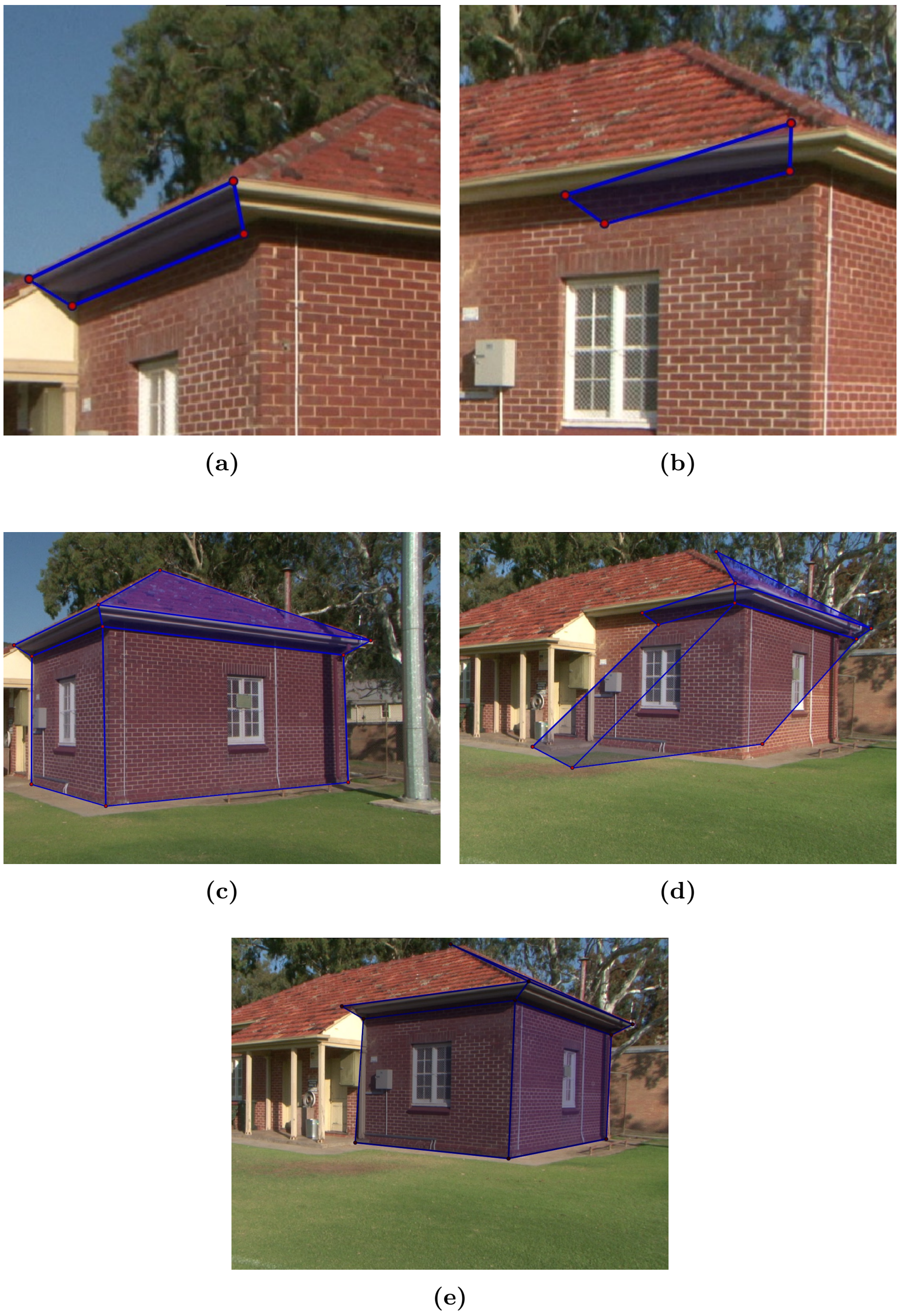


Figure 4.7: Fitting for multiple faces. (a) The initial face drawn by the user. (b) The resulting fit, seen from another view. (c) Additional faces drawn by the user. (d) The resulting fit for these additional faces, which are constrained by the fit of the face in (a). (e) The resulting fit when using all faces.

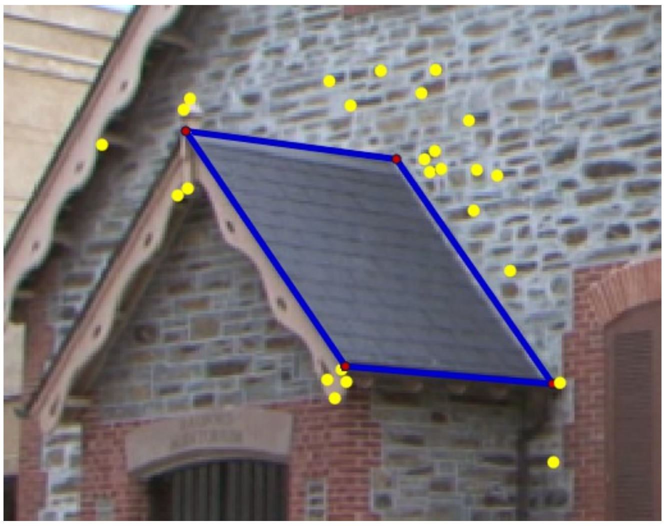


Figure 4.8: Selecting features outside of the face

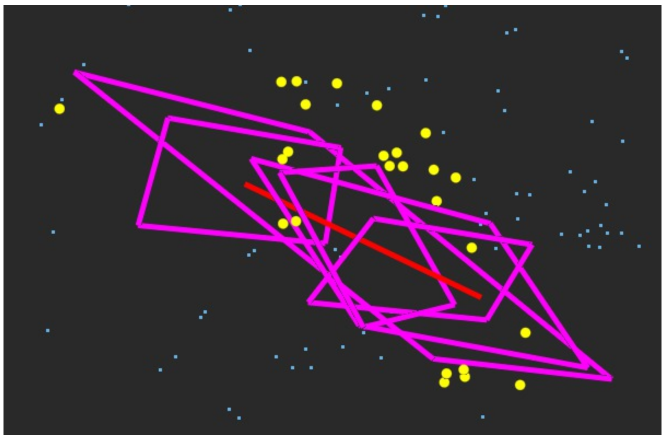


Figure 4.9: Sampled planes for the selected features

parameters required to define a face will again depend on the number of points in the face which already have a defined position. This strategy requires determining a plausible range for each parameter. For a face with less than two pre-defined point positions, we parameterise plane orientation as a rotation of the viewing direction, and consider an angle range of $\mathbf{R} = ([-85°,85°], [-85°,85°])$. Here, we assume that the plane is facing the camera, and is not at an extreme angle to the current view. For a face attached to an existing edge of the model, a single rotation parameter defines rotation around this edge.

For a face with no pre-defined point positions, a parameter specifying the distance of the plane from the camera is also required. Determining the range of this parameter is more challenging than determining the range of orientations, as it requires selecting a range of depths in an unbounded 3D space. We make the assumption that features in the region around the face will include features from nearby surfaces, and use the depths of these neighbouring features in specifying the depth range.

To select these features, we set a threshold of k feature points required for hypothesis generation, and select the k feature points with the minimum distance to the drawn face in the initial frame. We found that a value of 25 was sufficient for most areas of the tested scenes. Figure 4.8 shows the set of nearby features selected for a face with minimal texture. The SfM process generated no points within the face. The depth range of hypothesised planes is set so that the distance from the point corresponding to the centroid of the 2D face to the camera is between the distance to the camera of the closest and furthest features from the set. In generating hypotheses, orientation is sampled evenly, while depth is sampled on a log scale, so that depths are sampled more densely closer to the camera, where difference in depth has a larger effect on the appearance of the plane in the sequence. Figure 4.9 shows a selection of the hypotheses generated for the depth range determined from the selected features.

With hypothesised faces generated, optimisation and selection of hypotheses can be performed using the 2D likelihood function. 2D likelihood is evaluated for all generated hypotheses using the likelihood function $\mathcal{L}_{2_b}$ described in Section 4.3.6. The top 5% of hypotheses are retained and optimised, and the best hypothesised face is selected and added to the model. Figure 4.10 shows fitting results for a pair of faces containing minimal features. For the first face, 0 feature points were found within the boundary. For the second face, 2 features points were found.

An alternative method of initialising mesh patches which are subsequently refined by a photoconsistency-based process is given in [50]. In this method, the initial depth of patches is estimated by intersecting the viewing direction of nearby images (with an additional depth hint from the user used for correction where this initialisation fails). This method has the advantage of not requiring any feature points for the initialisation. However, while a similar initialisation method could be sufficient for planes close to a central point around which the camera path orbits, we would not expect this method to generate appropriate initial depths for other camera paths and plane positions.

4.6.2 Frame selection

For evaluating 2D likelihood, the method described in Section 4.3.5 selects a set of frames in which a face is expected to be visible using the set of features within the boundary of a face. With minimal features available, we cannot reliably use features to estimate the visibility of a face in a frame. This frame selection method also selected a single set of frames used in evaluating all hypotheses. As the set of hypotheses generated for a face with sparse features covers a wide range of angles, we expect that selecting frames where all hypothesised faces are visible would give a set of frames without enough divergence from the initial frame for reliable appearance comparison. For faces with sparse features, an alternate frame selection method is used, with a set of frames independently selected for each hypothesised face.

The 2D likelihood function measures the difference in appearance between a face in the initial frame and a set of comparison frames. For a given degree of misalignment between a hypothesised face and the true location of that face, we expect that the greater the divergence between the cameras used to capture the initial and comparison frames, the greater the measured difference in appearance. To reduce the effect of the choice of frames on the evaluated likelihoods of different hypotheses, we aim to select a set of frames for each hypothesised face with an equivalent divergence in camera positions. We measure divergence with $a(F_i, F_j)$ giving the angle between a ray from the camera centre for the frame F_i through the centroid of the face, and the corresponding ray for frame F_j. For a desired number of comparison frames m (set to 5) and a maximum divergence r (set to 45°) between the initial frame and any comparison frame, a set of frames $\{T_j\}$ is selected as described in Algorithm 4 for each hypothesis.

Algorithm 4 Frame selection with sparse points

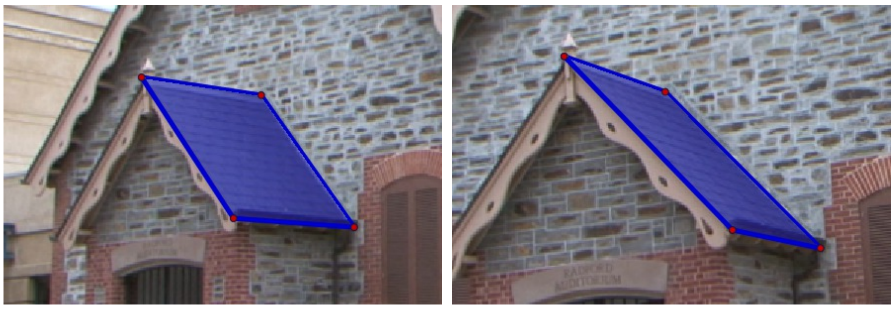
```
Determine a set of candidate frames $\{C_i\}$ for which the hypothesised face is facing the camera and within the image bounds
Add the initial frame to $\{T_j\}$
while $|T| < m$ do
    Set a minimum allowed divergence between frames in $\{T_j\}$,
    $d_{min} = \min(r/m, \max_{ij}(a(C_i, T_j)))$
    Sort $\{C_i\}$ by minimum divergence $d_i = \min_j(a(C_i, T_j))$ from any frame in $\{T_j\}$
    for $i = 1$ to $|C|$ do
        if $d_i >= d_{min}$ then
            Remove $C_i$ from $\{C_i\}$ and add to $\{T_j\}$
            break
        end if
    end for
end while
```

	Trial				
Roof 1	1.103516912	0.727803713	1.304542497	1.022104142	1.021560943
Roof 2	8.816799029	6.96323659	6.600319533	6.581734414	6.052167065

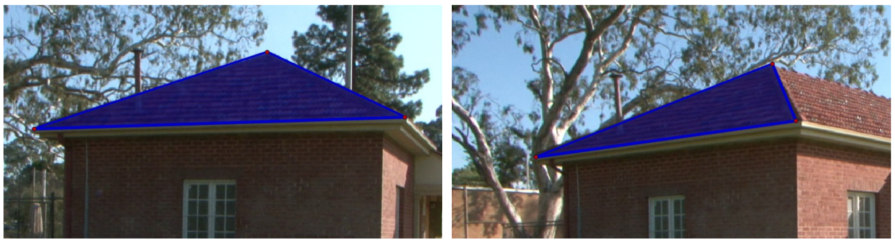
Table 4.2: Image error for fitting with minimal features.

To avoid biasing the selection of the best hypothesised face towards hypotheses which are only visible in frames with a small divergence from the initial frame, a penalty cost is added to the 2D likelihood. The complete function used to select the hypothesis to add to the model is $\mathcal{L}_{2_b} + f_d \sum_j \max(0, r/m - d_j)$, where d_j is the minimum divergence between frame T_j and any other frame in $\{T_j\}$, and f_d is a constant scale factor. This penalises hypotheses with less than the desired divergence between the selected frames.

Table 4.2 shows test results for faces fitted to surfaces with minimal detail using the method for fitting with sparse features. The fitted faces are seen in Figure 4.10. In both cases, the fitting process generated faces close to the optimal user-specified face.



(a)



(b)

Figure 4.10: Test results for fitting with sparse features. (a) Roof 1 (b) Roof 2. Images on the left show the user-specified face. Images on the right show fitting results.

4.7 2D modelling interface

We implemented and tested two interfaces for specifying the model, aimed at different user requirements. The first interface is designed for precise construction of the model, one face at a time. With this interface, the user clicks on an image to mark out a single face, which is then fitted to the image set. The user can inspect the fitting result in different frames, and adjust the result if required. This interface is similar to that provided by traditional 2D drawing software, and systems for modelling from images such as [4] and [42]. The second interface is designed to require less precise interaction from the user, and is closer to the sketch-based interface of systems such as [56] and [60]. With this interface, the user provides a complete sketch of the object in one view. This sketch is processed to recover a set of 2D faces, and this set of faces is collectively fitted to the image set.

4.7.1 Specifying vertices

To allow the user to directly specify the boundary vertices $\{\mathbf{u}_i\}$, we provide an interface that allows faces to be quickly constructed by clicking on vertex locations. Clicking on the image adds a new vertex at the cursor position. Existing vertices are highlighted when the cursor is placed nearby, and can be selected by clicking. If an existing vertex is selected, adding a new vertex will also add an edge between the new and selected vertices. A newly added vertex is selected automatically, so that continued clicking can be used to construct a series of edges, and clicking again on the original vertex completes the loop of a new face.

4.7.2 Finding faces

When the user completes a loop by creating an edge between two existing vertices, a new face is added, if possible. This requires determining whether a cycle is formed by the existing edges and the newly added edges, and which cycle should be added as a face. To determine the new face to add, cycle finding is performed using a depth-first search along edges where we exclude from the search edges which are unlikely to be included in a new face. Under the assumption that models being constructed represent the surface of solid objects, we expect that the edges supplied by the user represent a set of non-overlapping faces. We assume that each edge may be found in at most two

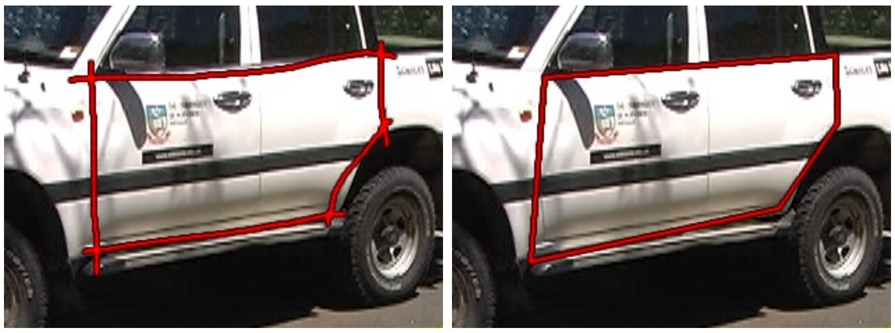


Figure 4.11: Specifying shape with freehand lines

faces, if the direction of the edge is different in each face given a consistent winding order. As such, directed edges already contained in a face can be excluded from the search. This typically excludes much of the existing geometry of the model, and can make the search significantly faster.

4.7.3 Specifying shape with a sketch

Directly specifying the shape of the model with mouse clicks allows for precise placement of vertex locations. However, such precise placement may not be required if edge information in the frame can be used to determine accurate edge locations. To reduce the precision of user interaction required, an alternative interface was developed, where the user specifies the shape of the model by roughly drawing a set of line segments, each corresponding to an edge of the model. This sketch is processed to recover a set of edges, and image edge information is used to refine vertex placement.

To construct the 2D polygonal model from this collection of lines, the user-supplied sketch is first converted to a set of vertices and edges, where the endpoints of each stroke give positions for a vertex pair, connected by an edge. Nearby vertices are clustered and merged, giving connected edges. These connected edges are then processed to determine a connected set of faces.

4.7.3.1 Clustering vertices

Vertex clustering is performed using the adaptive clustering method proposed in [85]. This clustering method was chosen due to it having the desirable property of being

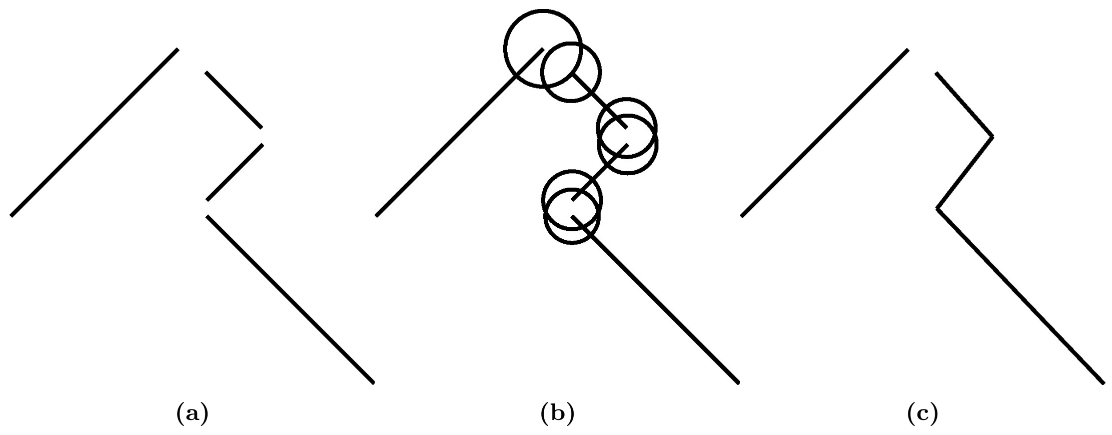


Figure 4.12: Clustering vertices

able to connect all neighbouring vertices, while still preserving fine detail in the sketch. This is achieved with a tolerance for clustering that is determined separately for each vertex, adapting to the level of detail in the local region. Any pair of vertices separated by less than the minimum of each other's tolerance will be grouped, and each member of the group will be replaced by a newly created vertex. This tolerance is determined by finding the average distance from the vertex to each edge, and taking the smallest of these as the tolerance. This adaptive method results in smaller tolerances in areas of fine detail, preserving shorter edges while still connecting longer edges. As the edge containing the vertex is included when determining the smallest average distance, no edge will be removed by merging its two endpoints. Figure 4.12 shows an example of applying this clustering to a set of unconnected edges, seen in Figure 4.12a. Figure 4.12b shows the corresponding tolerance windows for each vertex, with radius given by the minimum average distance to any edge. Figure 4.12c shows the edges resulting from clustering vertices falling within each other's tolerance windows, and creating a new vertex at the centroid of the cluster. Figure 4.11 shows a set of user-drawn strokes and the resulting edges.

This adaptive tolerance method can also be applied to determine the tolerance for vertex selection in the initial, click-based interface, reducing the precision required when selecting existing vertices.

4.7.3.2 Finding faces

After clustering, the set of edges is processed to find a set of cycles, and a set of faces is determined from these cycles. We again assume that the sketch supplied by the user represents a set of non-overlapping faces. As such, each possible directed edge will appear at most once in the set of faces. For each directed edge found in one or more cycles, the cycle with the smallest 2D area containing that edge is included in the set of faces, to avoid creating overlapping faces. Any edges not contained in a face are removed.

4.7.3.3 Fitting edges to the image

As the initially supplied sketch may be rough, and several endpoints are merged to give each vertex location, the positions determined for edges of the model may not correspond to the positions of the represented edges in the image. To better align the edges of the 2D model with the image, an optimisation process is applied. LM minimisation is applied to optimise the 2D vertex locations, minimising the distance between a series of samples on the edges, and the nearest strong edge point detected in the image. This process is similar to the edge fitting procedure used in [34] and other papers. The Facade modelling system [2] similarly performs fitting to an image to optimise user-supplied edges.

4.7.4 Editing tools

The interface also provides a set of tools for editing the model. These tools can be used to perform various operations which are common in 2D and 3D drawing and modelling packages. The available tools are listed in Table 4.3.

4.7.5 Adding curves to faces

User-defined faces which are strictly composed of a set of straight edges will be poorly suited to modelling geometry featuring significant curvature. To model such geometry, the user may use the *Curve Tool* to define a face as consisting of both curved and straight edges. Two techniques for defining curved edges were implemented, one extending the sketch-based interface to include curves, and the other based on clicking and dragging points on an edge. In the first technique, a polynomial curve is fitted to a

- *Erase:* Remove vertices, edges, and faces from the model
- *Divide:* Split existing edges at a specified point
- *Anchor:* Adjust the position of vertices in one or more views
- *Transform:* Adjust the position, scale, and orientation of a selected set of faces
- *Planar Transform:* Move and rotate faces, vertices, and edges on a specified plane
- *Curve:* Replace edges with curves

Table 4.3: Editing tools

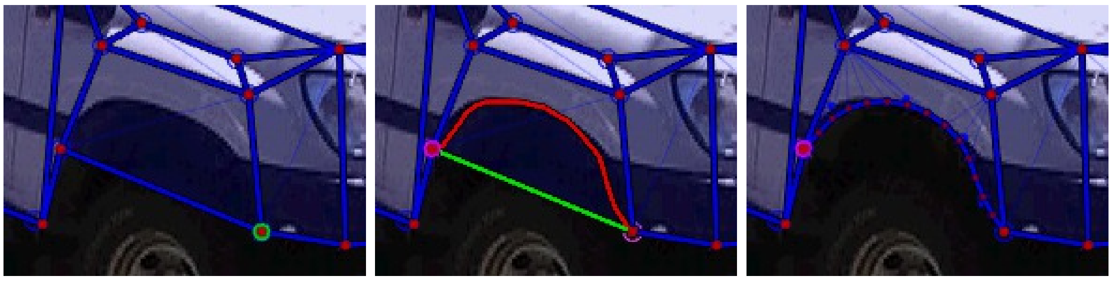


Figure 4.13: Specifying a curve by fitting to a drawn curve

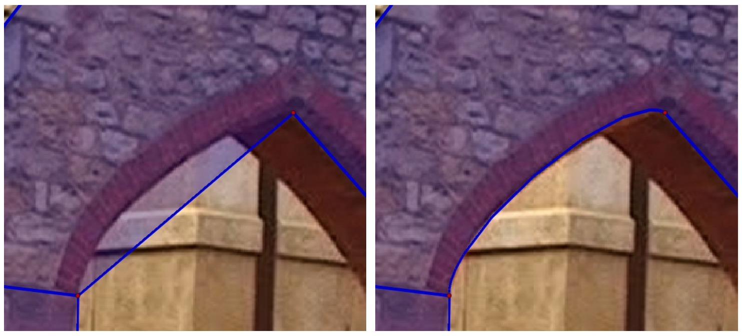


Figure 4.14: Specifying a curve by dragging an edge of the model

sketch of the curved line drawn by the user. This is shown in Figure 4.13. As accurately drawing a freehand curve may be difficult for some users, a curve may alternately be specified by clicking and dragging a point on a straight edge of the face, with a curve being fitted through the specified point location and the end points of the edge. This method is shown in Figure 4.14. For a face including curved edges, a polygonal boundary is generated for the face by sampling a set of straight line segments representing the curve.

4.8 Results

Figure 4.15 shows examples of models generated with the modelling process described in this chapter. This figure includes a frame of the sequence from which each model was generated, the set of user-defined faces for the model, and a novel view of each model with texture mapping applied. The selection of models shows the type of scenes for which modelling with this process is appropriate. The process is suited to modelling objects which can be intuitively decomposed into a set of planar facets. This makes the process suited to modelling a wide range of man-made objects and environments, including scenes which do not present the constraints on vanishing geometry used by modelling techniques intended for reconstructing architecture such as [4]. This process can be successfully applied for reconstructing objects with reflective and translucent surfaces, such as the car seen in 4.15a, and objects with curved surfaces and surfaces with minimal detail, such as the opera house seen in 4.15c.

4.9 Summary

In this chapter we have presented a method for interactive modelling from video sequences with a sketch-based interface. This method allows for free-form modelling of objects as a collection of planar faces. In contrast to traditional sketch-based modelling methods, our method uses image and point cloud data recovered by SfM to resolve the ambiguity of a 2D sketch representing a 3D object. Unlike other methods for interactive modelling from image and video sequences, modelling does not require the user to specify point locations in multiple frames, or require other constraints such as vanishing lines. As with the method described in Chapter 3, fitting is performed hierarchically,

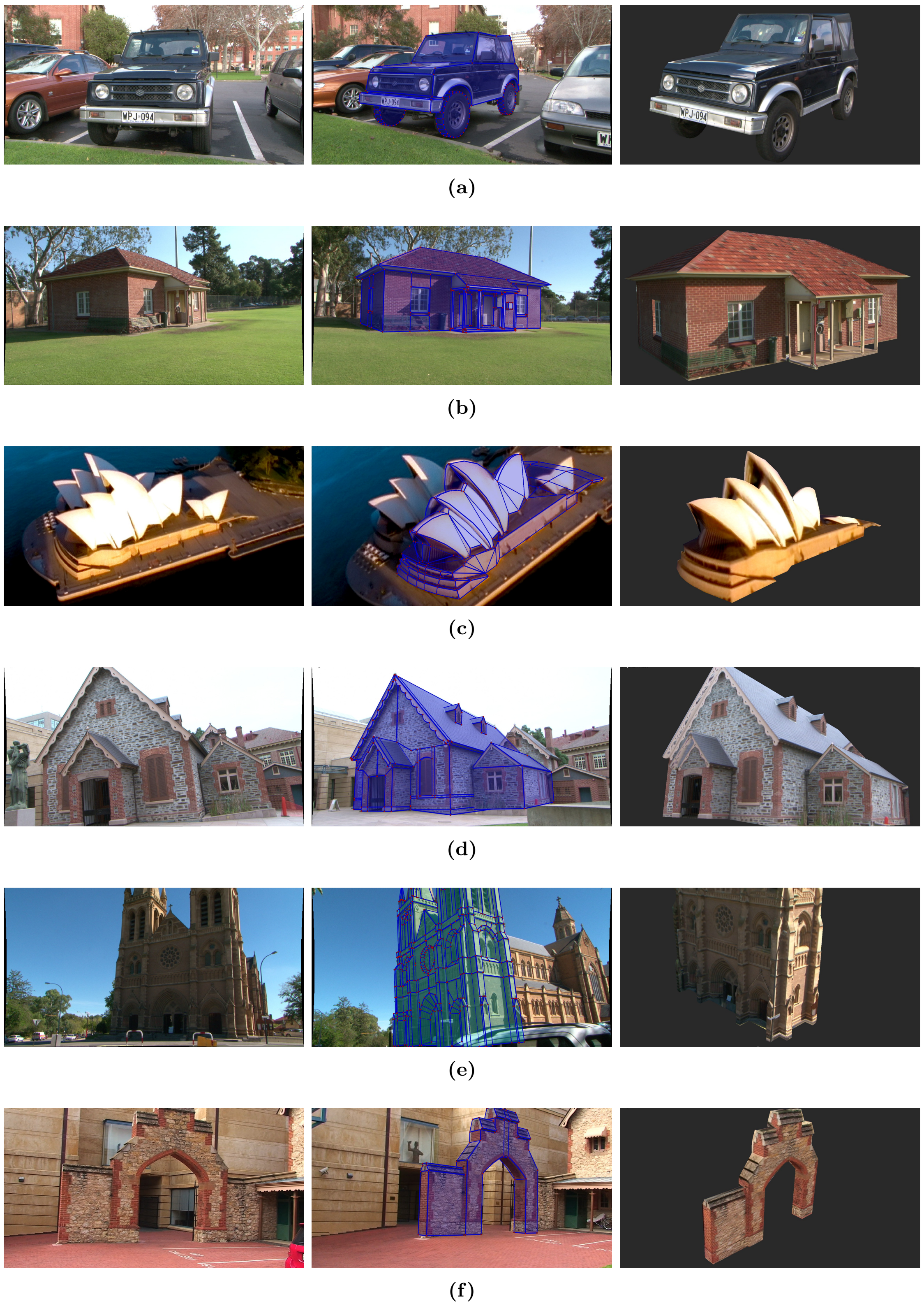


Figure 4.15: Models generated for a range of sequences. Images on the left show a frame of the original sequence. Images in the centre show the completed model overlayed on the sequence. Images on the right show the textured model from a new view.

with candidate models selected using point cloud data before performing image-based testing and optimisation. We have described the application of this fitting process to individual faces, and to connected sets of faces, and a technique for generating hypothesised faces in regions with sparse features. We have also described the modelling interface and tools provided for refining and editing the model.

5

Modelling With Geometric Constraints

The method described in the previous chapter allows for the construction of a 3D model by fitting planar faces. However, fitting faces to the visible surface of an object in a video will often be insufficient to construct a complete 3D model. In many cases it will be difficult or impossible to obtain images of all faces of an object. For many object types, the user will be able to estimate the geometric relationship between the visible structure and the structure in the unseen regions. The user may also be aware of geometric constraints on the structure (such as coplanar faces, parallel edges, and edges with equal lengths) which are not enforced for a model generated by this polygonal modelling method.

The following chapter describes modelling operations allowing the user to incorporate knowledge of geometric constraints on the object structure into the modelling process. Structural information supplied by the user can be used in replicating partial geometry to create complete 3D objects, and to constrain the shape and position of model faces. Geometric constraints can be applied to modelling from images in systems such as [1], [2], and [41]. These methods typically require the user to specify sufficient constraints to fully specify the model parameters. In this chapter, we focus on the problem of applying geometric constraints to models defined by user interaction in a single view, with the model parameters being potentially underconstrained.

We first introduce extrusion and mirroring operations in Section 5.1. These operations can be used to generate a complete 3D model from a selected set of faces. We

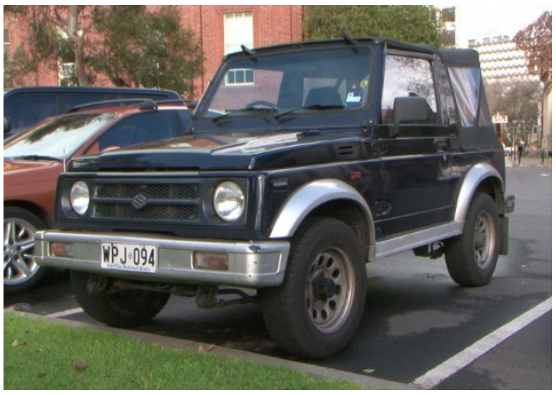


Figure 5.1: A car with symmetric structure

describe the interactive process for specifying these operations, the method of generating a set of hypothesised models from the information supplied by the user, and the selection of the final model. These operations require determining the parameters of a 3D operation from a 2D interaction.

In Section 5.2 we discuss modelling with primitive shape types, which allow for the construction of planar faces with regular geometric features. The available primitive types are introduced, and the method of interactively specifying primitives in a single frame is described. We describe the method for generating and evaluating hypothesised primitives, including the method of fitting primitive shape boundaries to the points specified by the user.

In Section 5.3 we describe techniques for incorporating geometric constraints into the bounded plane models described in the previous chapter. We describe planar constraints applied to the plane parameters of generated faces, and boundary constraints defining properties of the edges of a generated face. Some material on the mirroring and extrusion operations was originally presented in [84] and [86]. The techniques described in this chapter were developed and implemented by the author.

5.1 Extrusion and mirroring

If the views of an object are limited, models constructed using the techniques of the previous chapter will only provide a partial model of the object's surface. However, this partial model can provide a basis for constructing complete 3D models. 3D models can be constructed from individual faces or sets of faces by operations such as extrusion and mirroring. In an extrusion operation, a set of faces is replicated, with faces added between corresponding outer edges. Mirroring likewise replicates a set of faces, with the replicated faces undergoing a reflection transform.

Mirroring operations can be applied in modelling many objects, as the 3D structure of manmade objects is often symmetric about a plane. For objects such as the car seen in Figure 5.1, a set of images from one side of this symmetry plane can be sufficient for modelling the object's unique geometry. The remainder of the geometry can be obtained from this modelled geometry by a mirroring operation. Exploiting structural regularity with these operations can both reduce the time required to model an object, and allow for modelling of objects when unoccluded views of the complete object are unavailable. Texture maps generated for the visible surface of an object can be replicated on newly generated geometry, giving texture to faces which are not visible in the sequence.

Mirroring and extruding geometry are common operations in CAD systems and traditional 3D modelling packages such as 3ds Max [54]. These operations are also used to generate complete models in the sketch-based single image modelling method of [60]. These operations require the user to specify a 3D direction and distance in the case of extrusion, and a plane of symmetry in the case of mirroring. The novelty of our approach is that we allow the user to perform these operations with a 2D dragging interaction in a single frame of the sequence. Although the operations are underconstrained given only this interaction, we again use the image and point cloud data to resolve the ambiguity, minimising the amount of interaction required.

5.1.1 Defining the operations

Mirroring operations are defined by a plane M about which vertices of the object are reflected. The parameters of the mirror plane can be determined given a single mirrored vertex $\mathbf{V}_M$ and corresponding original vertex $\mathbf{V}_O$, with the plane normal $\mathbf{N}_M$ determined from the vector between the two vertices, $\mathbf{N}_M = \frac{\mathbf{V}_M - \mathbf{V}_O}{\|\mathbf{V}_M - \mathbf{V}_O\|}$, and the distance of the plane from the origin given by $d_M = -\mathbf{N}_M \frac{\mathbf{V}_M + \mathbf{V}_O}{2}$. For a set of n original vertices $\{\mathbf{V}_{Oi}\}$, the corresponding mirrored vertices $\{\mathbf{V}_{Mi}\}$ are given by

$$\mathbf{V}_{Mi} = \mathbf{V}_{Oi} - 2(\mathbf{V}_{Oi} \cdot \mathbf{N}_M + d_M)\mathbf{N}_M,\ i = 1 \ldots n \tag{5.1}$$

Extrusion operations are defined by an extrusion vector $\mathbf{E}$ applied to each vertex. For an extruded vertex $\mathbf{V}_E$ and original vertex $\mathbf{V}_O$, this vector is given by $\mathbf{E} = \mathbf{V}_E - \mathbf{V}_O$.

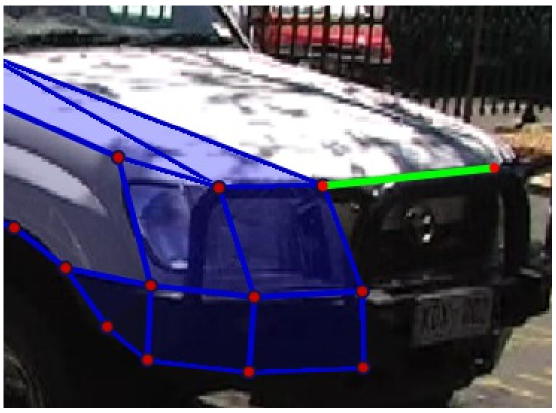


Figure 5.2: Initialising the mirroring operation for the car model

For a set of n original vertices $\{\mathbf{V}_{Oi}\}$, extruded vertices $\{\mathbf{V}_{Ei}\}$ are given by

$$\mathbf{V}_{Ei} = \mathbf{V}_{Oi} + \mathbf{E},\ i = 1 \dots n \tag{5.2}$$

5.1.2 Model definition

As a single 3D vertex position is sufficient to specify a mirror plane or extrusion normal, these operations are initialised by the user selecting a model vertex, and clicking on the image position of the corresponding mirrored or extruded vertex. This was chosen as the simplest interaction that could provide enough information to define the operation. Figure 5.2 shows the points selected in initialising this operation for the car model. A mirrored or extruded model $\mathbf{M}$ is parameterised by the distance C_M from the camera of the 3D vertex $\mathbf{V}_M$ corresponding to the specified 2D vertex $\mathbf{v}_M$. The new geometry has faces corresponding to the existing model faces. To construct a complete 3D model, faces are also added connecting each outer edge of the current model to the corresponding edge of the new geometry. Selection of the most probable model follows the model fitting process described in Chapter 3. This process allows us to efficiently explore the space of model parameters, and determine the most probable model given the image and point cloud data. A series of hypothesised models are generated spanning the space of plausible models, and their likelihood given the 2D and 3D data is evaluated hierarchically.

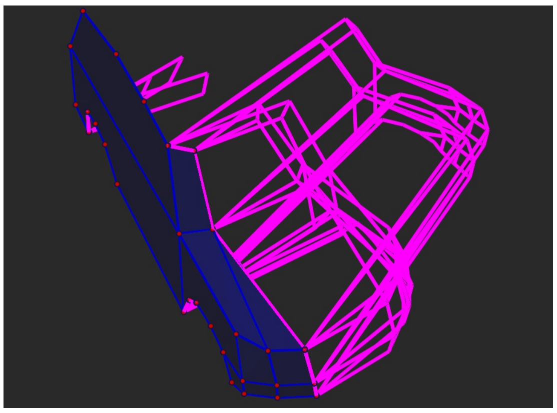


Figure 5.3: A selection of the hypothesised models generated in the mirroring operation

5.1.3 Generating hypothesised models

A set of hypothesised models is generated spanning the range of plausible values for C_M. A set of these models is seen in Figure 5.3. We again use the point cloud to determine the plausible range of model depths. The limits of this range are set to the distance of the furthest and closest feature points from the camera for the frame where fitting is initialised. Depth values are again sampled on a log scale, giving a greater density of models closer to the camera. A mirrored model is only considered plausible if its surface is not self-intersecting. As a simple test for self-intersection, we only consider a hypothesised model if each newly generated vertex of the model is on the far side of each face containing the corresponding original vertex.

5.1.4 Likelihood functions

Likelihood given the 3D data is evaluated for the hypothesised model with the function $\mathcal{L}_3$ defined in 3.3.1. The best 25% of hypothesised models are retained and optimised. For these models, 2D likelihood is then evaluated and optimised using the function $\mathcal{L}_{2_p}$ defined in 3.3.2.1. This likelihood function, based on the variance in colour values over a set of images, is used instead of the function used in fitting bounded plane models in Section 4.3.6, as we do not have a defined boundary for the newly modelled faces in the initial frame. As before, the most probable model is presented to the user. This model can be manually adjusted by specifying additional vertex locations in one or

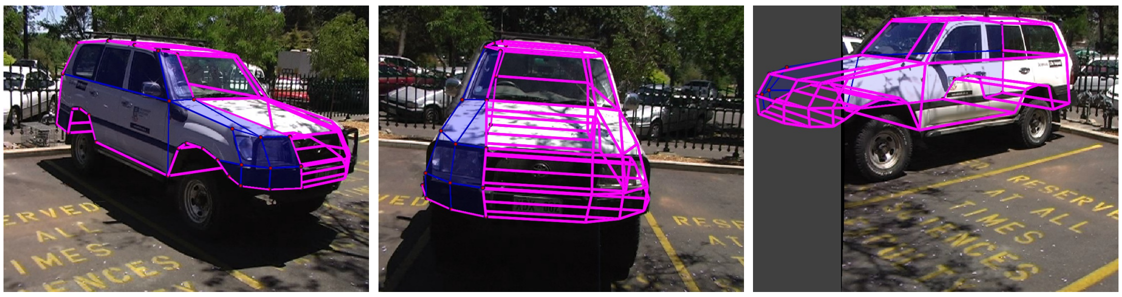


Figure 5.4: Mirroring the geometry of the car

more frames, and optimising the function $\mathcal{L}_U$ defined in 3.4.6. An example of geometry generated with a mirroring operation is shown in Figure 5.4.

5.2 Modelling with primitive types

The modelling method described in Chapter 4 is suitable for modelling with irregular polygons. In modelling manmade environments, planar facets with some regularity such as rectangles and circles are commonly encountered. If a user is aware that an element of a scene has a regular shape, a more accurate model can be obtained by allowing the user to incorporate this information into the fitting process. This can also reduce the interaction required to specify shape, as it may be possible to define a complete shape given a partial set of user-specified vertices. A circle model, for example, which would be time consuming and difficult to construct accurately by specifying each individual point on its boundary, can be fully specified from only three 3D boundary points.

This prior information on the shape of faces is incorporated through the use of primitive object types. These primitive types are similar to the primitive models fitted in Chapter 3. Fitting for those models was initialised with a single click or selection interaction. In contrast, fitting for primitive types described in this section is initialised with user-defined boundary points. This requires more interaction, but allows for greater control over the precise position of the resulting model, and makes it simple to specify vertices of the primitive which are shared with existing model geometry.

Primitive shapes are used for modelling in systems such as ICARUS [1] and Facade [2]. In these systems, the constraints provided by the structure of primitive models reduce the number of vertex or edge position constraints that the user is required to provide to specify the model parameters. The novelty of our method is that we only require the user to specify a partial set of constraints in one frame to generate the initial model fit, then allow additional constraints to be specified in further frames if required. In contrast to the method for specifying planar facets already described, for primitive shapes we do not treat user-specified boundary points as a hard constraint, reducing the precision required in specifying regular shapes. We do not rely on precise specification of the boundary by the user as accurately drawing projections of 3D shapes is difficult even for experts [87]. This would not be possible in a system requiring precise point locations to determine the model parameters from only user-specified constraints. The following section describes our method for incorporating primitive types into the modelling process, and gives examples of a pair of primitives.

5.2.1 Primitive shape models

Primitive shape models are defined similarly to the bounded plane model described in Section 4.1.1. These models are defined by a plane position parameter C and rotation parameters $\mathbf{R}$, and parameters $\mathbf{B}$ defining the boundary of the shape. The shape parameters required vary depending on the particular shape primitive. The shape of a circle primitive is defined by 3 shape parameters, specifying scale and 2D position on the plane defined by the position and rotation parameters. A rectangle primitive is defined by 5 shape parameters specifying 2D position and orientation, and scale in two dimensions.

5.2.2 Interactively defining a primitive

As in defining a bounded plane, a primitive shape is interactively defined by the user clicking on points on the boundary of the shape in one frame of the sequence. This provides a simple and intuitive method of defining primitives. After fitting has been applied, a primitive shape can be refined by adjusting locations of points, and by adding constraints by specifying point locations in additional frames. For the rectangle primitive the user specifies points corresponding to corners of the shape in the image. For the circle primitive, the user specifies arbitrary points on the boundary of the shape.

5.2.3 Generating primitive hypotheses

The fitting process for bounded plane models described in Chapter 4 is applied similarly for fitting primitive shapes. While defining a bounded plane model requires specifying each vertex of the polygon boundary, primitive shapes may be generated from a partial set of boundary points. With defined plane parameters, rectangles can be defined by three corner points, and circles can be defined by three points on the boundary. As such, a polygon defined by the points provided by the user may only partially specify the interior of the primitive shape. This partial region may not contain sufficient feature points to define plane parameters for the hypotheses or to perform frame selection for the appearance likelihood function. Where the partial region does contain sufficient points, these points still may not be representative of the complete surface. The method for generating hypotheses with sparse features described in Section 4.6 is therefore applied, with features selected from within and around the user-defined shape. With position and orientation parameters sampled for a hypothesis, the optimal shape parameters are determined given the user-specified points, and likelihood functions can be evaluated for the fully defined primitive shape.

5.2.4 Evaluating likelihood for primitives

In fitting bounded plane models, the boundary points specified by the user are assumed to be correct. We do not make this assumption for primitive shape types, due to the difficulty of drawing such shapes precisely in freehand. Rather, the boundary points specified are treated as soft constraints on the primitive shape. To apply these soft constraints, we generate hypothesised shape models satisfying the properties of the selected primitive, and include a cost on the distance between the user-specified points and the boundary of the shape in the likelihood evaluation. In selecting and optimising hypothesised primitive shapes, we evaluate the likelihood of the hypothesised shape given the set of boundary points $\{\mathbf{u}_i\}$ supplied by the user with the likelihood function $\mathcal{L}_B(\{\mathbf{u}_i\}, \mathbf{M}) = f_B \sum_i d_B(\mathbf{u}_i, \mathbf{M})^2$, where $d_B(\mathbf{u}_i, \mathbf{M})$ measures the distance from each specified point to the primitive, measured in the image where the point was specified, and f_B is a constant scale factor. For the rectangle primitive, this measures distances from points to the nearest corner of the shape. For the circle, this measures the distance to the nearest point on the shape boundary. A similar cost was previously applied in optimising models given additional user-specified vertex constraints.

To evaluate and optimise a primitive model, we combine the likelihood from this function with likelihood from the photoconsistency function $\mathcal{L}_{2_p}$, defined in Section 3.3.2.1. This likelihood, based on colour variance over the images, is again used as we do not know the true boundary of the face in the initial frame. We evaluate hypothesised primitive models with the combined likelihood $\mathcal{L}_P = \mathcal{L}_{2_p} + \mathcal{L}_B$. In optimising the combined likelihood for primitive models, optimisation of the position and orientation parameters C and $\mathbf{R}$ for $\mathcal{L}_P$ is interleaved with optimisation of the shape parameters $\mathbf{B}$ for $\mathcal{L}_B$. For each optimisation step for $\mathcal{L}_P$, $\mathcal{L}_B$ is optimised for a shape given the current position and orientation. $\mathcal{L}_B$ is relatively inexpensive to evaluate, and this interleaving reduces the number of more expensive appearance measurements required, compared to optimising all parameters simultaneously.

For the circle primitive, for any hypothesised plane a model can be generated with the set of 3 points specified by the user lying on its boundary. For the rectangle primitive, there will only be a limited range of planes for which a rectangle can be generated with corner points projecting to points close to all boundary points specified by the user. To reduce the number of hypothesised primitives for which the complete likelihood needs to be evaluated and optimised, increasing the speed of the fitting process, for the rectangle primitive we first evaluate $\mathcal{L}_B$ for all generated hypotheses, and only retain the 10% with the greatest likelihood. This rejects models for which the optimal shape, given the hypothesised plane, has a boundary which is far from the user-specified points.

5.2.5 Experiments

Table 5.1 shows the pixel error for 5 trials of fitting the circle and rectangle primitive faces. The error in the results is partly due to the surfaces which primitives were fitted to, which are not truly planar. The fitted primitives are seen in Figure 5.5.

	Trial				
Rectangle	7.34841287	9.918221188	7.87094532	9.11565263	9.083658757
Circle	4.679717576	6.335417062	3.828069591	5.097663461	7.029795156

Table 5.1: Image error for fitted primitives.

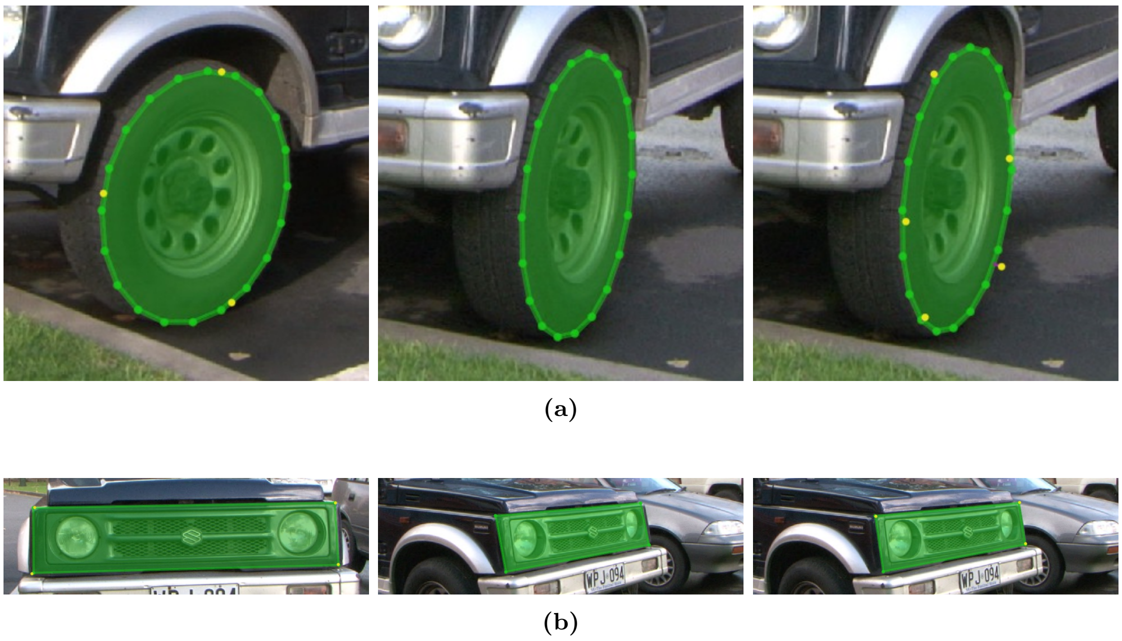


Figure 5.5: Test results for fitting primitives. (a) Circle (b) Rectangle. Images on the left show the user-specified face. Images in the centre show fitting results. Images on the right show manually refined results.

5.3 Defining faces with geometric constraints

In addition to modelling with primitive types, we provide other options for applying constraints commonly encountered in manmade environments to the modelled geometry. These constraints allow bounded planes to be generated with properties such as having the same orientation as another face in the scene, or including edges which are parallel or have the same length. We first describe a method for applying orientation constraints to planar models. We then describe a method for applying constraints to the edges of a face.

Constraints such as coplanarity and perpendicularity are also used in the single view modelling system described in [41], and in systems allowing modelling from multiple views such as [42]. In contrast to these systems, we do not require the number of constraints provided by the user to be equal to the number of free model parameters to generate the initial fit of the model to the image set. We again treat user-specified vertices as a soft constraint, and so do not require the user to perform the difficult task

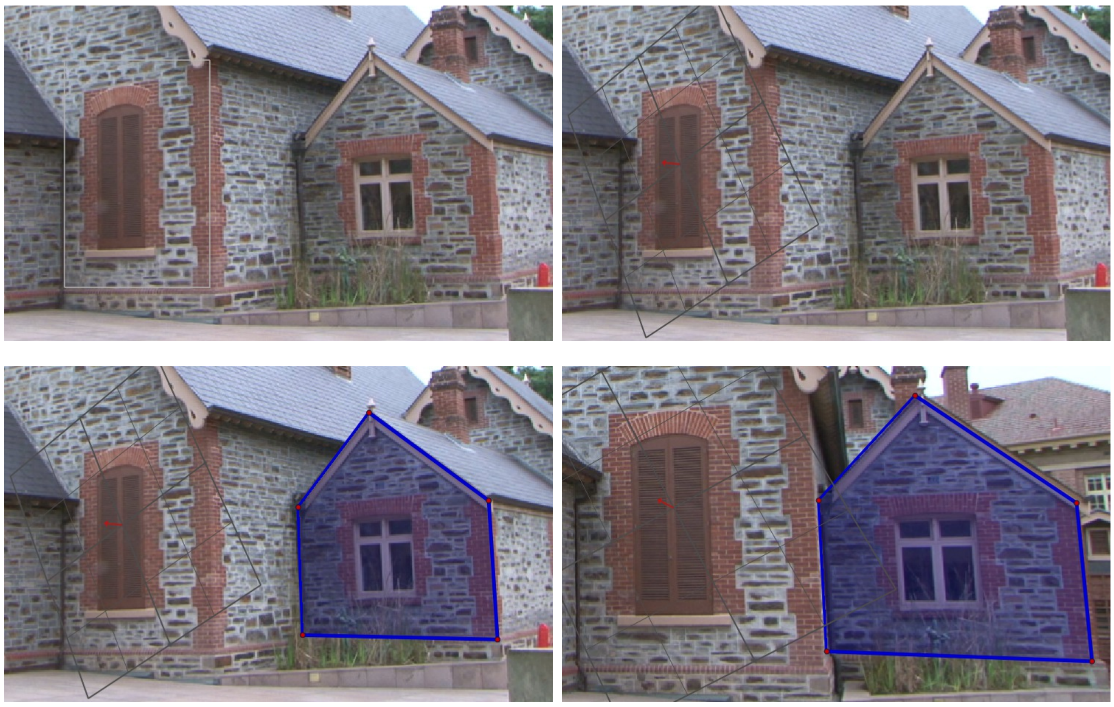


Figure 5.6: Selecting an image region to define an orientation constraint

of providing an accurate representation of the projected 3D geometry. An example of the use of geometric constraints to aid sketch-based modelling without images is given in [58].

5.3.1 Plane constraints

In the modelling method described in Chapter 4, the position and orientation of newly added faces of the model could be constrained by the parameters of adjacent faces. We allow similar constraints to be applied in fitting faces which are not adjacent, but between which the user is aware of some relationship. Surfaces with the same orientation are commonly encountered in manmade environments. For example, in a row of houses, the front wall of each house will typically have the same orientation. Orthogonal surfaces are likewise commonly encountered. We expect the front wall of a building to be oriented orthogonally to the street. Incorporating knowledge of relationships between orientations in the scene into the fitting process can be used to produce models which satisfy known properties of the modelled scene. This can also be useful in modelling areas of a scene with minimal texture detail. Accurate fitting for a face in a region with minimal detail may be aided by prior knowledge of the orientation of that face.

Knowledge of relationships between planes in the scene can be applied in our fitting method by using a specified orientation as a constraint on the fitting process, or by using all parameters of a selected plane as a constraint. Previously, in fitting a bounded plane model which was not attached to any existing model faces, we sampled parameters for both plane orientation $\mathbf{R} = (R_0, R_1)$ and position C. We apply planar constraints as hard constraints. These constraints can be used to specify the orientation parameters $\mathbf{R}$ for a face, and optionally to also specify the plane position C. Plane parameters can be specified by selecting the parameters of an existing face of the model, or by specifying a plane perpendicular to both the normal of an existing face and a selected edge of that face. A specific tool is provided for this task.

Specifying plane parameters reduces the number of free parameters in hypothesis generation and optimisation. This can reduce the ambiguity in fitting to regions where high photoconsistency likelihood can be found for planes with a wide range of parameters due, for example, to minimal texture in that region. In generating hypothesised faces from the point cloud data, an orientation constraint reduces the number of features required to determine the parameters of a hypothesised plane to 1.

Orientation constraints can also be used in specifying a new face attached to an existing face. In this case, the orientation constraint combined with the existing vertices fully specifies the parameters of the face. For structures where each face is aligned with one of three orthogonal axes, which are common in manmade environments, fitting to the image set is only required for the first face of the model, and orientation constraints can subsequently be used to fully define the parameters of each added face, if each added face is attached to an existing face. Planar constraints can similarly be applied in extrusion and mirroring operations. With a plane normal specified, the operation can be fully specified with a dragging operation indicating the position of a point on the mirrored or extruded surface.

Planar constraints can also be specified from regions of a scene which are not included in the model. This can be used to allow fitting results for regions with high detail to assist in fitting faces to regions with the same orientation, but sparser features. Such a region can be specified by dragging a selection box around a region of an image.

The fitting method described in Chapter 4 is applied to fit the specified rectangle to the images, and the orientation and optionally depth of the fitted region are selected as constraints. Figure 5.6 shows the process of specifying orientation constraints given an unmodelled plane in the scene, and using these constraints to fit a new face.

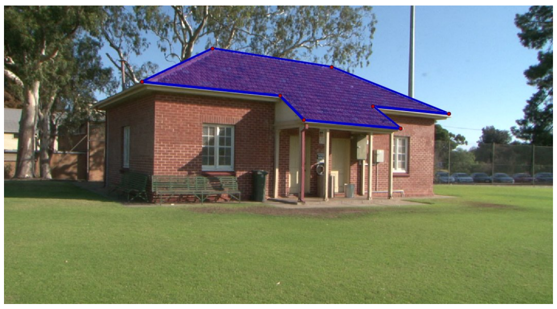
5.3.2 Boundary constraints

Further constraints can be applied to the model as *boundary constraints.* Boundary constraints define relationships between edges of a given face. These constraints can be used to specify that, for example, two edges of one face of the roof of a house are parallel, while two other edges have the same length. The following constraints can be specified for a model edge:

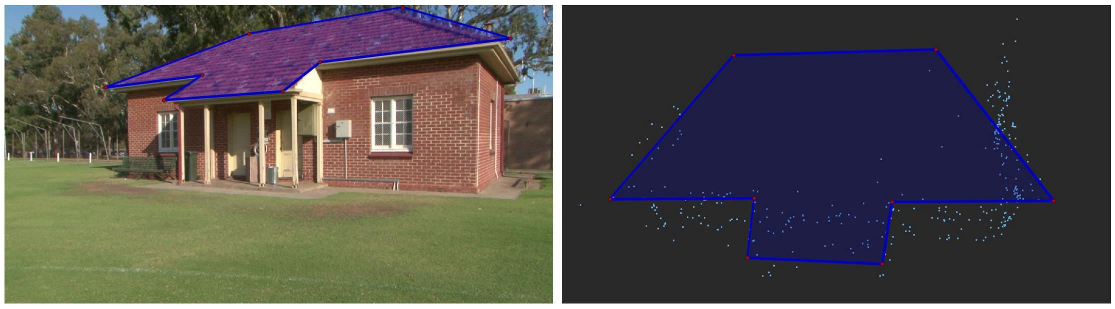
- Parallel to another edge
- Perpendicular to another edge
- Same length as another edge

These constraints are again treated as hard constraints, being enforced in the generation and optimisation of a hypothesised face. As in fitting a primitive shape, the set of vertices $\{\mathbf{u}_i\}$ supplied by the user are treated as soft constraints, as we do not expect the user to draw a face satisfying the desired properties exactly. We again optimise a model with the specified geometric constraints, given parameters for the plane and a set of user-specified points. This requires a parameterisation of the boundary of the model. We parameterise the boundary by an origin point $\mathbf{o}$ and a set of length and angle pairs $\{(l_i, \theta_i)\}$ defining the length and orientation of the edge to each subsequent point in polar coordinates.

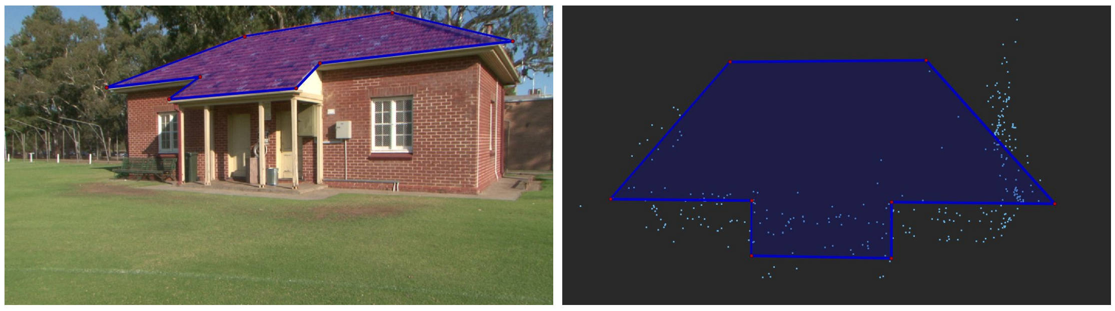
Each specified boundary constraint defines one of these edge parameters. As in primitive fitting, the likelihood function $\mathcal{L}_B$ is evaluated to determine the likelihood of a set of boundary parameters given the user-supplied set of points, with $d_B(\mathbf{u}_i, \mathbf{M})$ measuring the distance from each user-specified point to the corresponding vertex of the model. The fitting process for faces with constraints follows the fitting process for primitive types. After hypothesis generation, the 10% of models with the greatest boundary likelihood are retained. Optimisation of the boundary parameters is again interleaved with optimisation of the plane parameters, avoiding the need to re-evaluate



(a)



(b)



(c)

Figure 5.7: Test results for fitting with boundary constraints. (a) Initial frame with user-specified face (b) Fitting result for the face without constraints (c) Fitting result with constraints on edge angles and lengths

photoconsistency on each minimisation step for the potentially large set of boundary parameters. The hypothesised face maximising the likelihood $\mathcal{L}_{2_p} + \mathcal{L}_B$ is selected and added to the model.

Figure 5.7 shows an example of fitting with boundary constraints applied. 5.7a shows the user-specified face in the initial frame. 5.7b shows the resulting fitted face. 5.7c shows the fitted face after applying boundary constraints to enforce perpendicular and parallel edge angles, and edges with the same length. This fitted face more accurately models the geometry of the scene.

5.4 Summary

In this chapter we have presented some techniques for incorporating the user's knowledge of geometric properties of the scene into the modelling process described in Chapter 4. These techniques can be used to increase model accuracy, improve results in regions with minimal detail, and model parts of the scene not visible in the video sequence. Mirroring and extrusion operations can be used to generate complete models from partial structure. Point cloud and image data is used to determine the parameters of an extrusion or mirroring operation initialised with a simple 2D interaction. Primitive face types can improve the accuracy of the model and reduce the interaction required in specifying commonly occurring shapes. Planar constraints are used in modelling properties such as common orientations and coplanarity commonly encountered in manmade scenes. Boundary constraints are used to apply geometric regularity to faces which are not modelled by a primitive face type. In contrast to a number of other methods for modelling from one or more images, we do not require sufficient constraints to fully specify the model parameters before the model can be generated, and do not require the user to precisely specify 2D projections of the constrained 3D geometry.

6

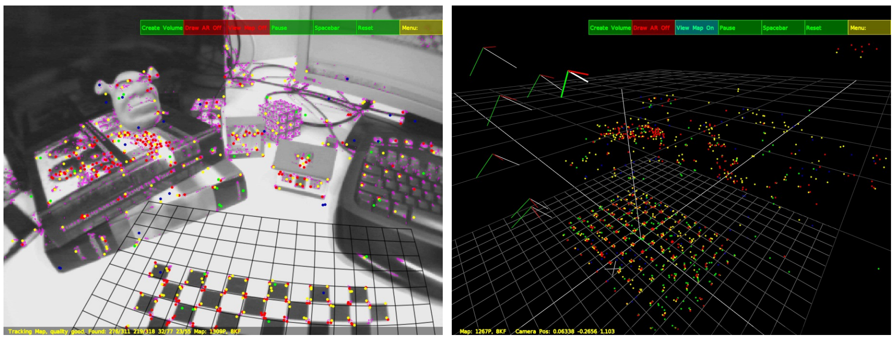
Modelling for Augmented Reality

In Augmented Reality (AR), computer generated imagery is incorporated into live video footage of real environments. Convincingly incorporating synthetic elements into a real environment can require modelling geometry of that real environment, so that this geometry can be used to construct occlusion masks, be used to model interactions between real and simulated objects, and be used for graphical effects such as lighting and shadows. Interactive modelling provides a means of rapidly constructing this geometry.

In the following chapter, we describe two interactive approaches to the problem of modelling geometry for AR applications. We first present a method based on the modelling process described in previous chapters. This method is intended for use in AR applications where significant mouse-based interaction is possible. We then present a method for modelling in scenarios where the camera is also used as the interface, and the application requires modelling of specific objects in the scene.

We first describe the tracking system which provides camera and point cloud information for a live camera, and describe the incorporation of the interactive modelling method described in previous chapters into an AR system. This allows rapid modelling of an environment while footage of the environment is being captured, and provides several advantages over other methods of modelling for AR applications.

We then introduce our in-camera modelling method. This method combines existing methods for graph-cut segmentation and silhouette carving in a novel feedback framework in which an object selected by a user in a frame from a live camera is segmented in that initial frame, and this segmentation is used to initialise segmentation in subsequent frames. From the set of segmented images, we construct a volumetric model



```
Tracking Map, quality good. Found: 276/311 219/318 32/77 23/55 Map: 1309P, 8KF
```

```
Map: 1267P, 8KF   Camera Pos: 0.06338 -0.2656 1.103
```

Figure 6.1: The PTAM interface, showing the points being tracked, and the associated point cloud and camera positions for the keyframes and current frame.

of the object using silhouette carving, and this model informs the segmentation of the object in further frames. This process is performed in real-time, allowing the user to observe the current state of the model and add further views as needed to refine the reconstruction. In contrast to other methods for reconstruction in AR environments, this method can be used to obtain detailed reconstructions of selected objects in the scene, and can be used to reconstruct objects which do not satisfy photoconsistency constraints. This chapter gives details on each stage of the method, and presents results generated with this approach.

Material on incorporating the interactive modelling method into an AR system originally appeared in [88]. For this method, the author was responsible for extensions to the modelling system to aid modelling for AR, and for integrating models into the AR environment. The in-camera modelling method was originally presented in [89]. For this method, the author was responsible for developing and implementing the graph cut segmentation, user input, and initial mesh model components of the system, and collaborated with his co-authors in conceiving the complete system.

6.1 Video tracking for AR

The SfM process previously used to reconstruct camera and feature point data is designed for processing a complete sequence after it has been captured. To incorporate synthetic 3D elements into live video, AR applications require recovering camera parameters for video frames as those frames are being captured. The PTAM system [62]

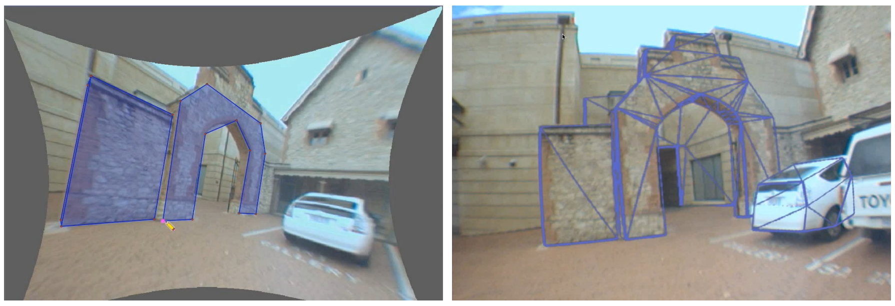


Figure 6.2: The model being constructed on a frame of the video sequence, and the completed model rendered in the AR environment.

achieves this by performing a combination of real time tracking and bundle adjustment. PTAM decouples the tracking process from the process of building a map of features in the scene. Beginning with an initial estimate of camera pose and a set of 3D feature points generated in an initialisation step, camera pose is subsequently estimated by minimising the distance between the current estimate of the map of 3D features and the current tracked feature positions. In parallel, but not in real time, the map estimate is refined by performing bundle adjustment on a subset of frames seen so far. This provides a live stream of frames, feature points and camera parameters. The PTAM system is shown in use in Figure 6.1.

The PTAM system was designed for use in small indoor environments. In these environments, results compare favourably with offline SfM systems. As with other SfM systems, the point cloud generated by PTAM is sparse in regions with minimal texture detail, and in regions which do not satisfy photoconsistency constraints. Reconstructed point positions also tend to be quite noisy is regions only viewed in keyframes with short baseline between the corresponding cameras. The reconstruction improves in these regions when more views have been provided by the user. PTAM tends to lose track frequently in outdoor environments, particularly where lighting is variable and where the scene features a large number of similar features. For successful tracking, PTAM requires a camera with a wide-angle lens.

6.2 Modelling for AR environments

In Section 2.3.3 we reviewed a range of approaches to 3D reconstruction for AR. In AR systems such as [63] and [65], a model of the environment is constructed by a user manually specifying vertex locations in multiple views of the scene. These systems typically do not make use of image information in the modelling process. The polygonal modelling process described in previous chapters provides a method of building models for AR applications with mouse-based interaction in single frames, significantly reducing the interaction required from the user, enabling more rapid reconstruction. To apply this method in an AR system, we use the camera, point cloud, and keyframe data obtained from PTAM in the same way that data from an offline SfM process was used previously.

This modelling process allows the models required by an AR application to be generated for a previously unseen environment, and the use of the data obtained by PTAM allows construction of the models to be interleaved with their use in the application. This makes it straightforward to add additional views to the set of frames being used for modelling, avoiding the need to recapture frames already seen if the set of views was found to be insufficient, as might be required if reconstruction was performed offline on a precaptured video sequence.

Before modelling is performed, the PTAM tracking process is paused. Modelling interactions, as described in Chapter 4, can then be performed on static frames from the sequence captured so far. A camera with a wide-angle lens is typically used with PTAM, as this significantly improves tracking performance. Such a lens significantly distorts straight lines in the images. As straight lines provide important cues for the user in constructing the model, modelling is performed on undistorted frames of the sequence. Images are automatically undistorted using calibration information recovered in a calibration process required by the PTAM system. A useful byproduct of the initialisation process for tracking with PTAM is an extracted dominant plane in the scene, which can be used as an AR surface. This plane can be also used as a ground plane in the modelling process, which can be used to apply planar constraints to the model as described in the previous chapter.

Once a model has been partially or fully generated, the tracking process can be resumed. When tracking is resumed, the current model is overlayed on the frame,

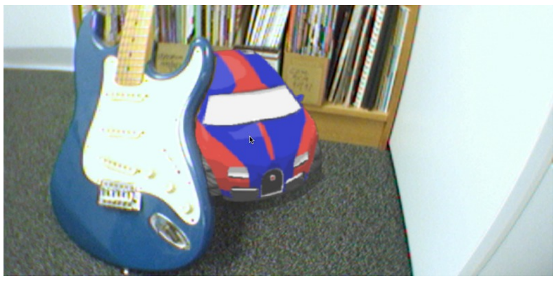


Figure 6.3: A synthetic car model occluded by a real guitar

as seen in Figure 6.2. By interleaving tracking and modelling, the user can inspect the model as it is being constructed, and see which parts of the model need more detail, then move the camera to a position from which the missing details are visible, before continuing modelling. The user can likewise correct the model if it is found to be poorly aligned with the images from a particular viewpoint. This would not be possible with an offline modelling process. As 3D models are constructed in the same frame of reference as the map of the world maintained by the tracking system, this interleaving also removes the need for additional processing to register the model with the map before it can be used in an AR application. However, this process does still require relocalisation by PTAM to recover the new camera position on resuming the tracking process.

Once incorporated into the map maintained by PTAM, models generated with this process can be used in multiple ways by AR applications. The projection of a model into the current image can be used as an occlusion mask, so that synthetic objects inserted behind modelled objects will be correctly occluded by real geometry. Figure 6.3 shows a synthetic car being occluded by a real guitar for which a model has been constructed. Models can also be used for simulating interaction between synthetic objects and the real environment. For better integration of synthetic objects into the real environment, models can be used to cast shadows onto synthetic objects, and for synthetic objects to cast shadows onto the world. In the scene in Figure 6.4, modelled geometry is used for physical simulation. A synthetic car model is jumping off a ramp, and casting shadows onto the real scene.

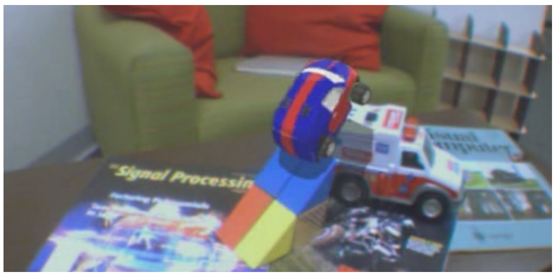


Figure 6.4: A synthetic car model casting a shadow onto a real ramp and toy ambulance

6.3 A method for in-camera modelling

While the method described above allows modelling to be interleaved with a live camera reconstruction process, the modelling process still requires significant mouse-based interaction. Depending on the environment where the AR application will be used, this type of interaction may be awkward or infeasible, or excessively time-consuming. An in-camera modelling approach, with a camera being used to control the modelling without an additional input device, may often be more suitable. An example of in-camera modelling is found in [65], where the input to the modelling process is provided via a combined camera and wheel and button interface. A description of several systems in which Virtual Reality or AR applications are constructed in the same virtual environment in which they will be used is provided in [90].

The following section describes an approach to the problem of reconstructing objects in an AR environment where the camera is used as the input device. We present a novel in-camera modelling technique, using a silhouette carving approach to recover the visual hull, applied to silhouettes obtained live through a graph cut segmentation process (an overview of visual hull methods was presented in Section 2.1.2). Silhouette carving has the advantage of being suitable for reconstruction in cases where photoconsistency constraints are not satisfied, such as objects with translucent or reflective surfaces, or objects with minimal surface features.

Silhouette carving is not able to reconstruct any concavities on the object surface, and requires a wide range of views for a complete reconstruction. As such, we expect that models generated with this method may be less accurate than models generated with the methods described in previous chapters, and with other reconstruction techniques. The goal of this method, however, is to generate reconstructions which are sufficiently accurate for the AR applications. For generating occlusion masks, the most important property of the model is the accuracy of its boundary over the range of views from which the object will be observed. For replicating modelled objects in an AR environment, the most important property of the model is the accuracy of its appearance. By applying projective texturing to map the appearance of the object in frames already observed onto the model in a new frame, we can generate an accurate appearance for the object from a relatively low-fidelity model. For example, this texturing method will preserve the appearance of concavities on the object which are not represented in the model.

As the model is constructed, the current estimate of its shape is used to inform the segmentation process in subsequent frames. In contrast to the modelling methods described previously, this in-camera modelling method operates in real-time, with the model being updated and presented to the user for each new frame. The modelling process is initialised with a simple user interaction, using an intuitive interactive technique commonly used for image segmentation.

This method is novel in two main ways. First, we perform silhouette carving in real-time in an AR framework, allowing the user to view the current state of the model and add further views of the object as required. Secondly, to increase the robustness of the reconstruction process, we combine existing techniques of graph cut segmentation and silhouette carving in a feedback framework. In this framework, the current model estimate informs the segmentation in each new frame, and this segmentation is then used to refine the model. A common challenge for silhouette carving methods is the difficulty of obtaining accurate segmentations of an object over an image set. We use the current model of the object to provide an estimate of its shape in each frame, and this estimate is used to construct colour models for the appearance of the object in the new frame. These colour models inform the segmentation procedure, providing robustness in reconstructing objects with an inconsistent appearance from different viewpoints.

In contrast to in-camera modelling methods such as [65], this reconstruction method does not require the user to manually specify vertex locations in multiple images. Similarly to our system, the method of [34] uses graph cut segmentation to track the silhouette of an object over successive frames. However, no 3D reconstruction is performed. Methods for reconstructing from the visual hull [15] [16] typically assume that a sufficient range of calibrated views of the object are available from the start of the reconstruction process, and that silhouettes can be recovered by computing the difference between the available views of the object, and the known background for those views. The system described in [68] reconstructs an object with no interaction beyond rotating the object in front of a camera. Unlike our system, this method assumes that the object being reconstructed can be manually rotated, and this motion is required to separate the objects from the background. This method also assumes that the object has a planar structure and sufficient feature detail to allow reconstruction from tesselated feature points. The method of [69] allows for dense reconstruction of a scene in real-time. Unlike our system, this method assumes that the reconstructed scene satisfies photoconsistency constraints, and is designed for reconstructing a complete scene, rather than a selected object in the scene.

6.3.1 Method overview

In this method, we apply a silhouette carving process to model a selected object from a set of segmented images of that object. This process recovers a volumetric representation of the object. This is a common model representation used in silhouette carving procedures such as [13]. This representation was selected as it can be computed efficiently and robustly from a set of silhouettes. Compared with the image-based representations computed by methods such as [14], an explicit 3D reconstruction of the object allows for simpler and more flexible manipulation of the reconstructed object and integration of real and synthetic objects in the AR environment.

The goal of the method is to reconstruct a volumetric model $\mathbf{M} = \{m^i\}$ of an object in the scene selected by the user. Each voxel m^i in a grid is assigned a binary label. $m^i = 1$ indicates that the space corresponding to that voxel is occupied by the object, $m^i = 0$ indicates that the space is unoccupied. A silhouette of the object is recovered for each frame of the video. Voxels are labelled as being outside of the object if they are repeatedly observed outside the silhouette of the object. Requiring multiple observations before a voxel is removed reduces the susceptibility of silhouette carving to the erroneous removal of voxels due to errors in the camera parameters or silhouette recovery.

To recover silhouettes, we use a graph cut procedure [91]. Graph cut procedures are commonly used for interactive image segmentation problems. We chose a graph cut based segmentation method as this provides a means of extracting an accurate initial selection of the object from coarse user markup on a frame of video. As the camera is used as the interface for selection, we do not expect the user to perform precise interactions. Subsequent to the initial selection, the graph cut procedure provides a means of rapidly obtaining an accurate segmentation from image colour and edge information, constrained by any additional markup provided by the user.

Constraints on the size of the graph used for segmentation and the dimensions of the voxel grid allow us to achieve real-time performance. To assist in recovering the object silhouette in new frames, even in the presence of significant camera motion, the current model estimate is used to provide an initial estimate of the silhouette in each new frame. This estimated silhouette is used to construct and apply constraints to the graph, and in building the colour models used in the graph cut. An initial coarse mesh model is used for this projection until sufficient views of the object have been obtained to initialise the volume. Camera parameters for each live frame and a sparse point cloud for the scene are again provided by PTAM.

6.3.2 Graph cut segmentation

Graph cut [91] techniques are commonly applied for graph labelling problems. Applied to generating a silhouette of the object in a frame, this is a problem of labelling each pixel in the image as either belonging to the selected object, or to the background. The image is treated as a Markov Random Field, where each pixel is a node in the graph.

The Graph Cut minimises an energy function of the form

$$E(l) = \sum_{i \in \mathcal{N}} U(l_i) + f_V \sum_{p,q \in \mathcal{M}} V(l_p, l_q) \tag{6.1}$$

where l is the set of labels, $\mathcal{N}$ is the set of nodes, $\mathcal{M}$ is the set of all neighbouring nodes, and f_V is a constant weight.

The unary potential function $U(l_i)$ assigns a cost to each node for each possible label assignment (implemented as a cost on cutting the edge to the graph source or sink). It is defined as $U(l_i) = -\log \Pr(l_i = L|C_L)$, where L is either foreground or background, and C_L is the corresponding colour model. Foreground and background colour distributions are modelled as Gaussian Mixture Models (GMM). The probability that node i belongs to GMM C_L with k components is given by

$$\Pr(l_i = L|C_L) = \prod_{j=1}^{k} w_{Lj} \frac{1}{\sqrt{\det \Sigma_{Lj}}} \exp(\frac{-(I(i) - \mu_{Lj})^T \Sigma_{Lj}^{-1} (I(i) - \mu_{Lj})}{2}), \tag{6.2}$$

where $I(i)$ gives the pixel colour for node i, and w_{Lj}, μ_{Lj}, Σ_{Lj}, and $\det \Sigma_{Lj}$ are the weight, mean, covariance, and determinant of the covariance respectively for the jth component of the GMM. We use mixture models with $k = 5$ components.

To generate these models, samples of pixels belonging to these distributions are required. Both sets of samples can be provided directly, by the user drawing strokes on the image indicating foreground and background pixels. Alternatively, samples can be directly provided for only one labelling. Samples for the other are then taken automatically from the unselected pixels, and this sampling is refined through an iterative process. This is the method utilised for the GrabCut technique [92], and this is the technique that we use for initialising the modelling process.

Objects will typically feature contiguous regions of similarly coloured pixels. A good segmentation will preserve this, correctly labelling all pixels from such a contiguous group. To encourage this in image segmentation the binary potential function $V(l_p, l_q)$ assigns a cost to each edge for assigning different labels to the attached nodes. It is defined as $V(l_p, l_q) = \begin{cases} \exp(-\beta|I(p) - I(q)|^2),\ l_p \neq l_q \\ 0,\ l_p = l_q \end{cases}$, where β is a constant weight, and $I(a)$ is the RGB pixel corresponding to node a. This term encourages the separation between foreground and background to occur along edges in the image, these being the points at which the separation between foreground and background is most likely to occur. Combined with the colour term, this has the effect of separating pixels according to whether they more closely resemble pixels from the foreground or background distribution, with the separated pixels forming contiguous regions, separated along strong image edges.

6.3.3 Segmentation for successive frames

The inital segmentation relies on coarse marking provided by the user to generate colour models for the object and background, and to indicate the locations of some pixels belonging to the object. Providing this information for each frame would be tedious for the user. Instead, following the initial segmentation the current estimate of the model is used to initialise the segmentation for the subsequent frame. The

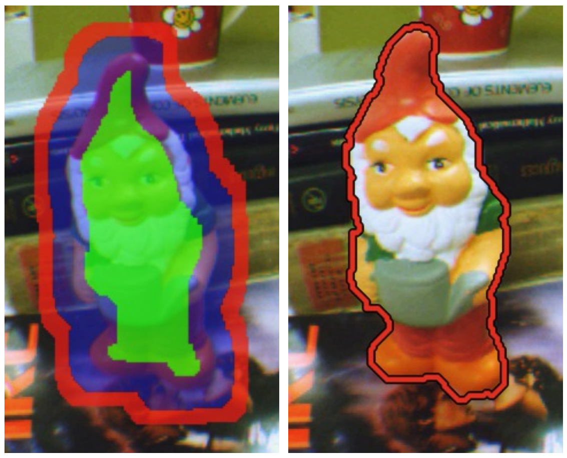


Figure 6.5: The graph cut region and resulting silhouette. The cut is performed on pixels within the blue region, with pixels in the red region labelled as background, and pixels in the green region labelled as foreground

silhouette of the model projected into a new frame is used to construct and constrain the graph in that frame, and to build colour models. Using a 3D model to estimate the silhouette has the advantage of providing robustness when the camera track is lost and relocalisation is necessary, as can occur relatively frequently with PTAM. The 3D model and the parameters of the camera in the current frame and the last frame before the track was lost are sufficient to project the silhouette into the current frame, without requiring knowledge of the camera motion for the intervening frames.

After the initial cut, the graph for subsequent frames is constructed for a band of pixels around this projected outline. For frames with a resolution of 640 by 480, a 20 pixel band is used. Performing the cut in this band, rather than on the full image, reduces the computation required to obtain the cut result, helping to maintain real time performance. Nodes on the interior edge of the band are highly weighted so that they will be labelled as foreground in the cut result. Nodes on the outer edge are similarly weighted as background.

New colour models for the graph cut are generated in each frame, as we do not assume that the object has a consistent appearance over the sequence. A foreground colour model is constructed from samples of pixels within the projected silhouette. A background colour model is constructed from pixels on the outer edge of the band.

Figure 6.5 shows an example of the region over which a graph cut is performed, and the silhouette given by the cut result. Although this method does not require the object to have a consistent appearance over the sequence, it does require that the object can be separated from the background on the basis of colour.

To encourage frame-to-frame consistency in the silhouette, a structure term is added to the energy function minimised by the graph cut. This function now takes the form

$$E(l) = \sum_{i \in \mathcal{N}} U(l_i) + f_S \sum_{i \in \mathcal{N}} S(l_i) + f_V \sum_{p,q \in \mathcal{M}} V(l_p, l_q) \tag{6.3}$$

where $S(l_i)$ is a term encouraging consistency between frames to reduce the probability of points being incorrectly labelled, and f_S and f_V are constant weights. $S(l_i)$ applies a penalty for labelling points close to the outer boundary as belonging to the object, and labelling points close to the inner boundary as belonging to the background. The structure term $S(l_i) = \exp\left(-\alpha d_s(i, b_{l_i})^2\right)$, where the function $d_s(i, b_{l_i})$ gives the distance from node i to the inner or outer boundary of the graph b_{l_i}, depending on the label of this node, and α is a constant weight.

An alternative method of initialising the cut for subsequent frames would be to simply use the silhouette boundary for the current frame. While this would likely succeed for small camera movements, it would be expected to fail in the presence of large camera motion, particularly if tracking failed and relocalisation was necessary. In the system for tracking object boundaries described in [34], a prediction of the boundary of an object in a new frame is made from the object boundary in the current frame, updated with an estimate of optical flow. While such a method provides robustness to camera motion, this requires accurate frame-to-frame tracking, and would not be able to provide an accurate prediction of the boundary if tracking failed for a number of frames.

6.3.4 Initial model generation

The silhouette tracking process relies on using an estimate of the model shape to project the object boundary into subsequent frames. Once the silhouette carving process has been initialised, the current volumetric model is used to provide this shape estimate. The silhouette carving process is not initialised until the user is satisfied with the segmentation of the object. Initialising the carving process also requires determining the bounds of the model volume. The initial segmentation gives 2D bounds for the volume, but the required depth range of the volume is unknown. To determine this depth, we first collect sufficient views of the segmented object (covering a rotation of $\approx 90°$) to estimate the bounds of the object from the set of 3D feature points observed within the silhouette over this range of views. Prior to initialising the carving process, we use an initial coarse 3D model to project the current silhouette into each subsequent frame. As with initialisation from the carved model, initialising the silhouette using this coarse model provides robustness against camera motion, and a means of relocalising the silhouette if the camera track is lost and recovered.

6.3.5 Generating the coarse model

To construct an initial coarse 3D model from a silhouette of the object in one view, we use the depth information provided by the PTAM point cloud. The coarse 3D model is generated by tessellating feature points inside the initial object segmentation. Feature points generated by the PTAM process may be noisy, or not correspond to actual points in the scene. To compensate for this, the coarse model is generated with support from the images of the scene seen so far. Before adding tesselated patches to the coarse model, we evaluate the photoconsistency of these patches, to avoid adding patches which deviate significantly from the true depth of the object.

Triangular patches on the surface of the object are hypothesised from sets of feature points. The model is constructed by iteratively adding patches which are likely to lie on the object's surface. At the first iteration, an initial set of hypothesised triangle patches is generated, and the best patch is selected using the photoconsistency likelihood function $\mathcal{L}_{2_p}$, described in Section 3.3.2.1, evaluated for the current set of keyframes obtained by PTAM. Any features inside this selected patch are removed from the set.

On subsequent iterations, patches are generated from pairs of features forming the outer edges of the currently generated model, and the remaining features. Patches are rejected if they overlap existing patches. Coarse model generation completes when no feature points remain, or all remaining hypothesised triangles have low photoconsistency likelihoods. A coarse model, and a silhouette projected into a subsequent frame on the basis of this model are shown in Figure 6.6. We do not expect this method to generate a particularly accurate model of the object, particularly as the object itself

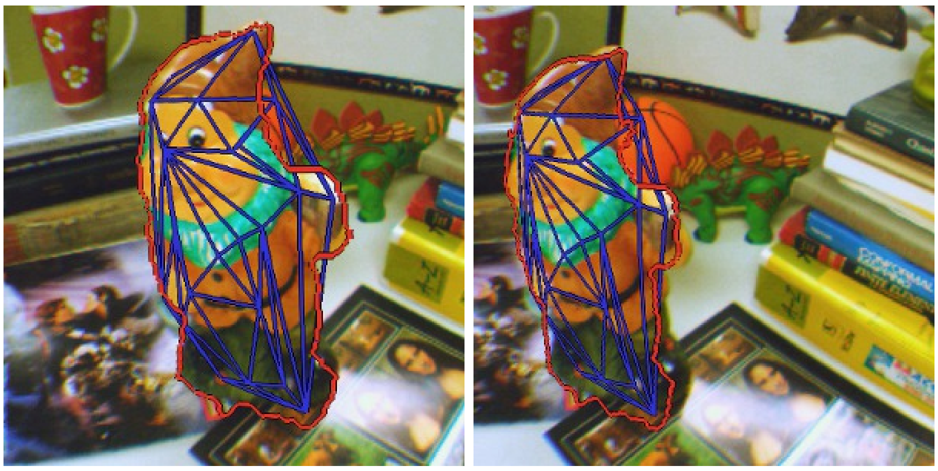


Figure 6.6: Automatic model generation. The generated triangle mesh is shown in blue. On the left, the red line shows the silhouette resulting from the graph cut in that frame. On the right, this line shows the silhouette from the first frame projected into a new frame on the basis of the generated model

may not fully satisfy photoconsistency constraints. However, the goal is not to generate an accurate model, but merely a model which is close enough to the true object structure to successfully initialise the graph cut process for each frame. A more sophisticated method for generating a model in an AR environment from tesselation of feature points is described in [68].

To project the silhouette into a new frame given the initial coarse model, a depth map is generated from the model, giving the depth of each pixel on the model surface in the current frame. The silhouette may extend beyond the bounds of the model. To give an estimate of depth for points outside the model boundary, depths along the boundary are interpolated to assign a depth to each point within the silhouette. Given these depths, a 3D polyline is constructed from the cut outline. This polyline is then projected into the next frame.

6.3.6 Initialising the graph cut with camera interaction

The initial object selection process requires marking part of the surface of the object. This allows the object to be selected with a simple, imprecise interaction. This interaction can be performed with the camera and two buttons as the input device.

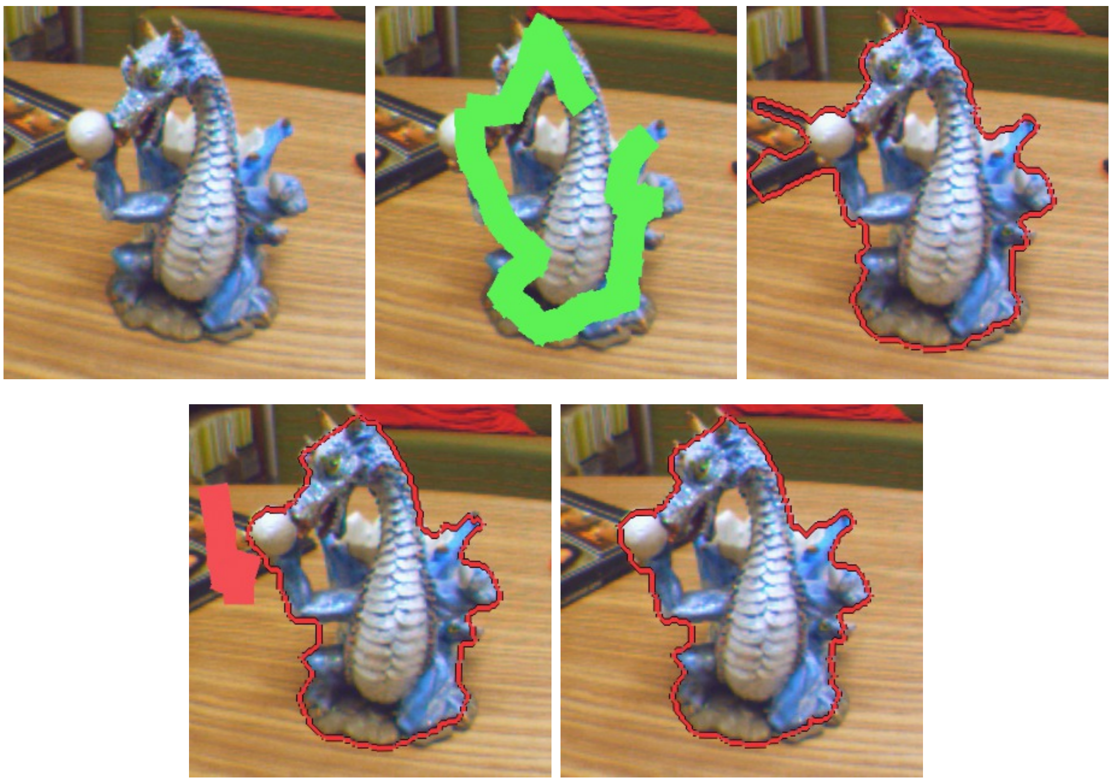


Figure 6.7: Initialising the graph cut. Green strokes indicate pixels on the object. Red strokes indicate background pixels, used to correct the initial cut result

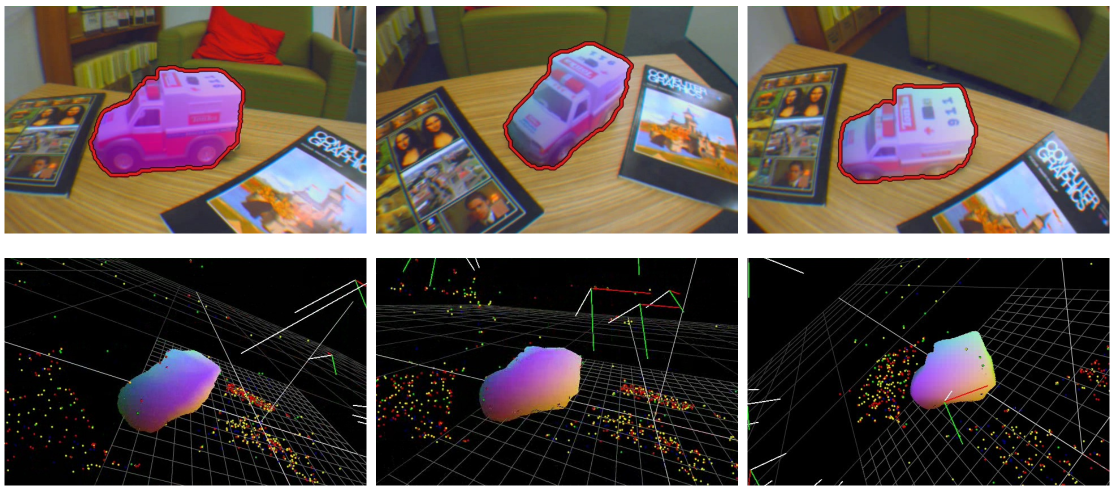


Figure 6.8: Silhouettes for a tracked object in several views, and the resulting model

Alternately, a mouse can be used if available. If the camera is used as the input device, the cursor position used in drawing on the frames of live video is fixed to the centre of the frame as the camera moves.

To create the appearance of drawing on the object using a moving camera and fixed cursor position, the markup should appear to be fixed to the geometry of the scene. We achieve this by estimating the depths of marked points as they are drawn, and constructing a polygon strip passing through these points. When the button is released, the projection of the polygon strip into the current frame gives the 2D markup used for initialisation. Figure 6.7 shows the graph cut process being initialised with user-drawn strokes, with additional strokes used to remove a background region from the initial cut result.

Depth estimation for the markup is performed with a similar procedure to the initial coarse mesh construction. We again use feature points recovered by PTAM to estimate the 3D structure of the scene. As the camera is moved with the button held, a mesh is progressively constructed by tesselating points close to the cursor. The cursor position is projected onto the nearest mesh triangle. While the depth estimates are coarse, this coarse depth is sufficient, as rough marking of the surface is all that is required for initialisation.

6.3.7 Volume model generation

This process generates a series of silhouettes of an object taken from multiple viewpoints. From these silhouettes, a volumetric model of the object is progressively refined by applying a sihouette carving technique. Figure 6.8 shows silhouettes for an object being tracked over several views, and a view of the resulting volumetric model. For each newly available silhouette of the object, an estimate of the model is updated given the current silhouettes.

Each voxel in the model is assigned a scalar label m_i. For each added silhouette, the set of voxels is projected into the corresponding frame, and labels are updated depending on whether each voxel lies inside or outside of the silhouette. For robustness to errors in the silhouette estimates in individual frames, and to errors in the estimated camera parameters for each frame, the silhouette carving process requires several observations of a voxel outside of the silhouette before it is removed from the model. Labels are initialised to a value $n = 8$, and decremented on each observation of a voxel outside of the silhouette. A current model estimate is generated for each frame by taking the set of voxels with a label $m_i > 0$. This model estimate is used in initialising the graph for each frame, providing an initial estimate of the silhouette in the new frame. We leave it to the user to determine when the model is complete, based on visual inspection of the result, as the required detail in the final model will depend on the application for which the model will be used. The model is stored when the user indicates that modelling is completed.

As camera or mouse interaction is used to generate the initial silhouette for this modelling process, it may also be used to refine the model during the process. If the silhouette generated for a frame includes regions not belonging to the model, strokes can be drawn on the frame to label that region as background in the graph cut, removing it from the resulting silhouette, and subsequently removing it from the volume. At present, interaction can only be used to remove additional regions from the volume, as label values are only maintained or decremented on each update step. Future work on this method could include a means for interactively adding regions to the volume, used to repair regions which can be erroneously removed for reasons such as similarity between the colour of the object and the background.

6.3.8 Results

The resulting model could be used to apply accurate lighting or other effects to an object, or directly manipulated, as seen in Figure 6.10, which shows objects which have been replicated in an AR environment. Texture for the replicated models was generated by interpolating a set of views of the object, projected onto the model surface. These views are drawn from a set of keyframes, selected based on proximity to the current camera position. Projective texturing of a model generated by silhouette carving is similarly applied in [15]. Compared with assigning a single colour to each voxel, this projective texturing method has the advantage of appearing to preserve features of the objects, such as concave regions, which are not reconstructed with a silhouette carving technique. This figure also shows replicated objects being occluded by the real object, demonstrating the use of the model for generating occlusion masks. Figure 6.11 shows replication successfully applied to a reflective object with minimal texture, and an object with a translucent and reflective surface, which would be difficult to reconstruct with a method based on photoconsistency. Figure 6.12 shows a further example of a real object modelled with this method occluding a synthetic object.

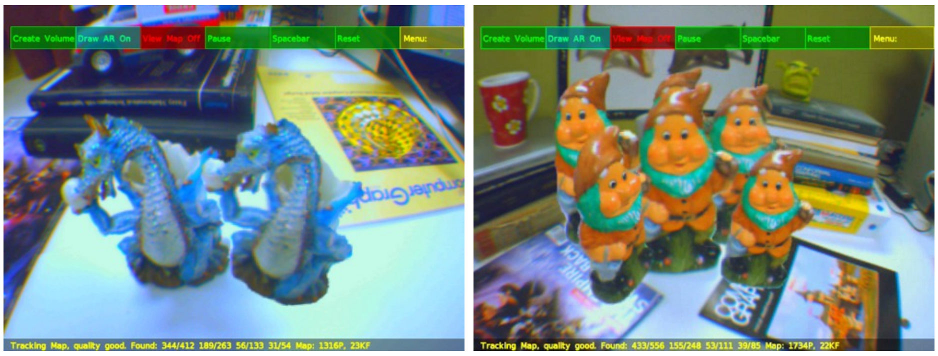
Table 6.9 gives timing results for each stage of the reconstruction process, averaged over 10 frames of a sequence. Timing results were computed on a PC with an 2.4Ghz Intel Core2 Quad CPU, 4GB of RAM, and an nVidia GeForce 8600 graphics card. Results were computed for a voxel grid with a resolution of 256 voxels in each dimension. These results demonstrate that the process is fast enough to give live feedback to the user.

6.4 Summary

In this chapter we have described the application of the modelling method introduced in Chapters 4 and 5 to Augmented Reality applications, and described a novel in-camera modelling method. In this in-camera modelling method, silhouette carving is used to generate a volumetric model from a set of silhouettes obtained with graph cut segmentation performed over successive frames. Modelling is performed in a feedback framework where the current model is used to initialise the segmentation in each new frame. The reconstruction is performed in real-time, and the user is able to inspect the current state of the model and add new frames as required. In contrast to other

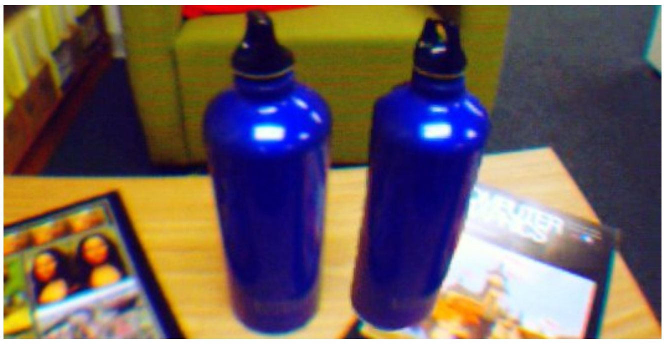
Stage	**Time**
PTAM Tracking	0.0859
Image segmentation	0.0438
Silhouette prediction	*0.0057*
Updating colour models	*0.0016*
Updating graph	*0.0300*
Computing graph cut	*0.0065*
Volume update	0.0016
Rendering	0.0690
Total	**0.2004**

Figure 6.9: Timing results (in seconds) for the stages of the modelling process

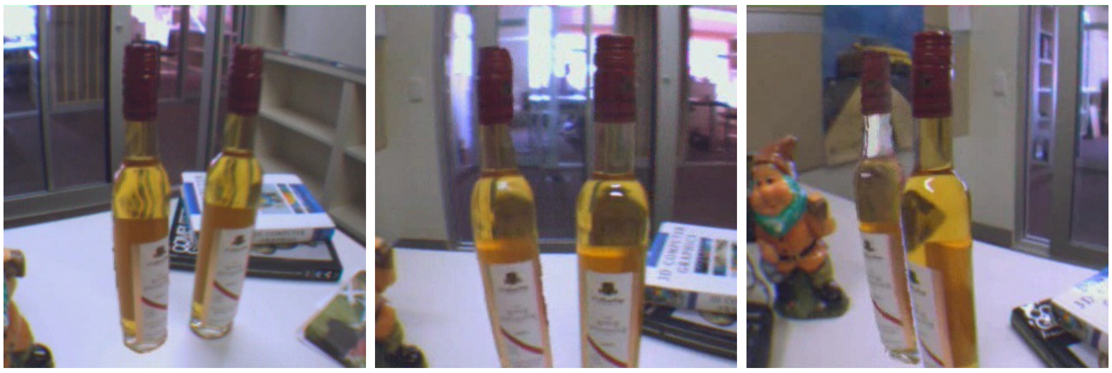


```
Tracking Map, quality good. Found: 344/412 189/263 56/133 31/54 Map: 1316P, 23KF
Tracking Map, quality good. Found: 433/556 155/248 53/111 39/85 Map: 1734P, 22KF
```

Figure 6.10: Replicated objects in an AR space



(a)



(b)

Figure 6.11: a: Replicating a reflective object with minimal texture. b: Replicating a translucent, reflective object

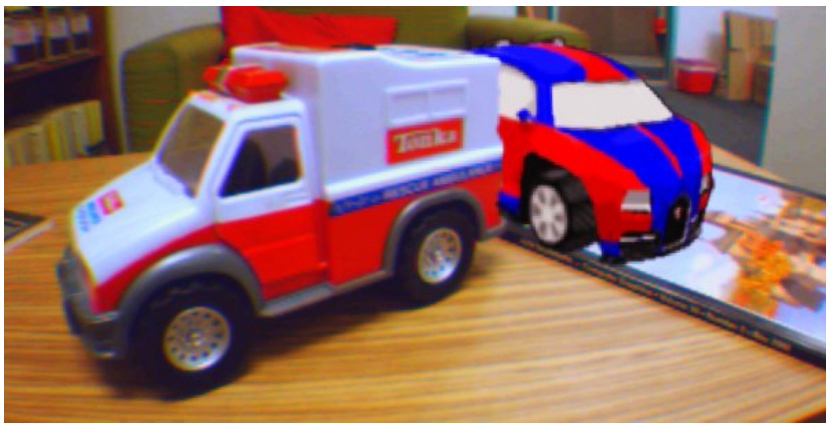


Figure 6.12: Real object occluding a synthetic object

in-camera modelling methods, this method uses simple painting interactions to initialise modelling, is able to reconstruct selected individual objects in a scene, and can reconstruct objects which have minimal features and do not satisfy photoconsistency constraints. We have demonstrated this method being used for copying and pasting objects and generating occlusion masks in an AR environment.

7

Conclusions

In this thesis, we have presented several methods for interactive 3D reconstruction from video sequences. This chapter summarises the contributions presented, and discusses possible directions for future research.

7.1 Summary

In Chapter 3, we described a method for modelling individual objects in a video sequence by fitting pre-defined model types selected by a user. A Structure from Motion process is applied to the sequence to recover camera parameters for each frame, and a sparse cloud of 3D points. A hierarchical fitting process, initialised by the user selecting a model type and roughly indicating its position in a single frame, is applied to find the most likely set of model parameters given the 3D point cloud and 2D image information.

We compute likelihoods given the 2D and 3D data for a large set of hypothesised models spanning the parameter space. Likelihoods are computed in order of computational cost, and further computations are not performed on models with low likelihood. The 3D likelihood is based on the distances from points to the model surface. This likelihood function is initially approximated by a simpler function, based on a histogram of surface point distances from the model centre, and this approximation is used to eliminate models with low likelihood before further testing is performed. The 2D likelihood function is based on photoconsistency. A further likelihood function based on user-specified vertex positions can be used to manually refine the fitting results.

In Chapter 4, we described a method for modelling with user-defined geometry, defined with sketch-based interaction in a single frame. To fit the specified model to the image set, we first described model fitting for an individual face. Plane parameters are recovered for a planar model with user-defined 2D boundary points. The fitting method follows the hierarchical method used for fitting pre-defined models in Chapter 3. The initial set of hypotheses is generated by sampling sets of feature points from the features within the region containing the specified face. A method was described for selecting a set of frames over which to compute the photoconsistency of the hypothesised faces, using visibility of features within the face to estimate the visibility of the complete face.

We then described the model fitting process for a set of connected faces, describing the process of sampling and optimising parameters for all faces in the set. An alternative for generating hypothesised faces was introduced for regions with sparse feature points. Features in the vicinity of the specified face are used to specify a depth range in the scene. Over this range hypothesised faces are generated spanning the range of possible orientations, and these faces are evaluated based on photoconsistency.

We described two input methods which can be used to specify the faces of the model in 2D. In the first method, the position of each vertex is directly specified. In the second, each edge is roughly drawn, and the endpoints of these edges are clustered to create a set of vertices which can then be optimised by fitting to image edges.

In Chapter 5 we described further interactive modelling techniques using user-supplied information on geometric constraints on the model structure. For modelling parts of the scene which are not visible in the image set, extrusion and mirroring techniques are used to generate complete 3D models from models of visible surface geometry. Given a user-specified location for a single vertex on the extended geometry, we use the model fitting method described in previous chapters to select the parameters for the mirroring or extrusion operation. To allow for user-defined models of commonly occurring shapes, we introduced primitive shape types. We described the method for fitting primitive shapes to the image set, with a set of user-supplied points used as soft constraints on the shape. To apply structural constraints commonly encountered in manmade environments, we described a technique for specifying planar constraints on model construction, and for applying geometric constraints to the shape of a face.

In Chapter 6, we discussed modelling methods which can be used in Augmented Reality scenarios. By applying the interactive modelling process described in previous chapters to 2D and 3D data from the PTAM tracking process, synthetic elements can be convincingly integrated into an AR environment.

We then described a reconstruction method using the camera as an input device, based on silhouette carving. This method is initialised with a graph cut interaction, from which a coarse polygonal model of the object is generated. This model is used to propagate graph cut segmentation results between frames. From a set of generated silhouettes, a silhouette carving approach is used to generate a volumetric model in a feedback framework where the current model estimate is used to initialise the graph cut in each new frame. Results of this method for occlusion boundary generation and object replication were shown.

7.2 Directions for future research

Our research in this area has suggested several possibilities for future work. The accuracy of reconstruction with these methods is limited by the accuracy of the original SfM reconstruction. While these methods can tolerate noise in the feature point positions, they are not robust to inaccuracy in the camera parameters. Interactive techniques similar to those used to specify model structure could potentially provide additional information to the SfM process. For example, the outline of the same shape provided in multiple views could be used to give additional feature matches, similarly to the way user-defined structure is used for camera calibration in systems such as Facade [2]. 2D shape fitting to the images could be performed to minimise the interaction required to specify the same shape over the sequence. Information on geometric properties could also be used to provide constraints on the reconstruction. For example, indicating that an outlined shape is a rectangle would place constraints on its edge lengths and angles between edges in the reconstructed space. Models generated for AR environments could also be used to improve tracking for those applications. Aligning models of objects in the scene with the captured images could increase tracking stability, and aid in recovering when the track is lost due to large camera movements.

By exploiting repetition and structural symmetry, the models generated using these techniques can include structure for occluded or poorly visible parts of a scene. This information could be used in conjunction with dense reconstruction techniques to provide a dense reconstruction for these parts of the scene. Presently, texture for the visible surface of an object can be repeated for the hidden surface when a mirroring operation is applied. Similarly, dense structure represented as a depth map over the model surface could be replicated when mirroring and extruding. Dense structure could also be replicated for modelled geometry which is repeated multiple times in the scene, so that the furthest instance of the geometry would have the same level of detail as instances closer to the camera.

Models of surfaces in occluded parts of the scene could also be used as a basis for a more complete dense reconstruction of the scene. Methods such as PatchMatch [93] can be used to synthesise 2D appearance in selected regions of an image. A similar technique applied to image and depth data for a surface could be used to similarly synthesise depth data for occluded regions.

Interactive modelling could also be applied to single images, by operating on data from single image reconstruction methods such as [94]. User-supplied information may be particularly useful in these cases, giving the high degree of ambiguity and occlusion expected in single images of a scene. While these methods typically rely on laser-scan data for training, models generated using the techniques described in this thesis could also be used as a source of data in training such an approach.

7.3 Conclusion

In this thesis, we have presented several related methods for interactive 3D reconstruction from video sequences. These methods allow reconstruction to be guided by a user, employing the user's understanding of a scene to obtain a complete and accurate reconstruction, while using automated methods to minimise interaction where possible.

Users intuitively understand the content of video sequences, and can readily perform interpretation tasks that are difficult to perform automatically, such as identifying objects present in a scene, identifying geometric relationships between structures, and estimating the structure of hidden regions. Users will often have specific requirements for which parts of a scene require modelling, and to what level of detail. By incorporating this knowledge into a reconstruction process through user interaction, we can generate detailed and accurate reconstructions which meet the user's requirements. We can also overcome some of the major limitations of reconstruction with purely automated methods, using information provided by the user to reconstruct surfaces which are occluded or only visible from viewpoints which are not available, and to reconstruct surfaces with a shape which can not be determined from image information due to an inconsistent appearance or lack of texture.

Interactive methods for reconstruction from image sets have traditionally relied on the user providing enough information to fully define the modelled geometry. Manually providing sufficient constraints to determine all the parameters of the model can be a highly involved and time-consuming process, typically requiring point or edge locations in a number of frames. In some situations, such as in AR environments, constructing detailed models with methods requiring involved interaction may be infeasible. In enabling reconstruction from single views, interactive methods have commonly placed significant restrictions on the types of scenes which can be modelled, for example by relying on geometric constraints common to architectural environments.

In contrast, the methods presented in this thesis allow the user to specify modelling operations with minimal interaction in single frames, and can be used to reconstruct a wide range of objects and environments, without being restricted to a specific domain such as architecture. These modelling methods are appropriate for a range of scenarios. In each method, modelling is performed on a video sequence, allowing the methods to use robust SfM processing to automatically recover camera parameters.

Where the scene can be modelled with pre-defined or primitive model types, we have presented a method allowing the user to fit the model to the images using simple, coarse interactions in a single frame, then refine the fit with additional interaction as required. Polygonal modelling is provided with a method allowing the user to specify a 3D model with interaction in a single frame using an intuitive sketch-based interface. This method can also be applied for modelling in AR environments. For AR environments where the camera is also used as the input device, a further method allows the user to model an object in the scene selected with simple graph-cut based interactions.

The methods presented in this thesis demonstrate that by combining interactive modelling with automated processes operating on data extracted from a video sequence, we can provide the benefits of interactive modelling while reducing the interaction required from the user to the minimum necessary to specify the desired modelling operations. These methods show the value and potential of providing intuitive interfaces which allow a user's understanding of a scene to inform a 3D modelling process.

Bibliography

[1] S. Gibson, R. J. Hubbold, J. Cook, and T. L. J. Howard. Interactive reconstruction of virtual environments from video sequences. *Computers and Graphics*, 27:293–301, 2000. 3, 23, 32, 34, 47, 55, 95, 100

[2] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96)*, pages 11–20, 1996. 3, 21, 32, 34, 89, 95, 100, 135

[3] D. P. Robertson and R. Cipolla. An interactive system for constraint-based modelling. In *Proceedings of the British conference on Machine vision (BMVC '00)*, pages 536–545, 2000. 3, 22, 34, 63

[4] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3D architectural modeling from unordered photo collections. In *Proceedings of ACM SIGGRAPH Asia 2008*, pages 159:1–159:10, 2008. 3, 23, 63, 71, 86, 91

[5] M. Wilczkowiak, P. Sturm, and E. Boyer. Using geometric constraints through parallelepipeds for calibration and 3d modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):194–207, 2005. 3

[6] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint.* MIT Press, 1993. 8

[7] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the the 4th Alvey Vision Conference*, pages 147–151, 1988. 9

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. 9, 67

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004. 10, 11

[10] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 10, 16

[11] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994. 10

[12] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:150–162, February 1994. 12

[13] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993. 12, 118

[14] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*, pages 369–374, 2000. 14, 118

[15] M. Li, M. Magnor, and H.-P. Seidel. Hardware-accelerated visual hull reconstruction and rendering. In *Proceedings of Graphics Interface 2003*, pages 65–71, 2003. 14, 117, 128

[16] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 345–352, 2000. 14, 117

[17] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 1067–, 1997. 15

[18] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000. 15

[19] G. Vogiatzis, C. Hernández Esteban, P. H. S. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 (12):2241–2246, December 2007. 15

[20] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *SPIE: Optical Pattern Recognition XII*, volume 4387, pages 95–102, Mar 2001. 16, 45

[21] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, 2007. 16

[22] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics symposium on Geometry processing (SGP '06)*, pages 61–70, 2006. 17

[23] J. Isidoro and S. Sclaroff. Stochastic Mesh-Based Multiview Reconstruction. In *Proceedings of the International Symposium on 3D Data Processing Visualization and Transmission*, pages 568–577, July 2002. 17

[24] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, pages 559–565, June 1999. 18

[25] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. From volumes to views: an approach to 3-d object recognition. *CVGIP: Image Understanding*, 55(2):130–154, 1992. 18

BIBLIOGRAPHY

[26] F. Lafarge, R. Keriven, M. Brédif, and H.-H. Vu. Hybrid multi-view reconstruction by jump-diffusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pages 350–357, 2010. 18

[27] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Transactions on Graphics*, 24(3):577–584, July 2005. 18

[28] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, May 1987. 19

[29] R. B. Fisher, A. Fitzgibbon, M. Waite, E. Trucco, and M. J. L. Orr. Recognition of complex 3-D objects from range data. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 509–606, 1993. 19

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the 3rd International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001. 19, 40

[31] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases (SSD '99)*, pages 207–226, 1999. 19, 43

[32] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999. 19, 33, 44

[33] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP '03)*, pages 156–164, 2003. 19, 44

[34] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002. 19, 89, 117, 122

[35] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, pages 187–194, 1999. 20

[36] J. M. Ferryman, A. D. Worrall, G. D. Sullivan, and K. D. Baker. A generic deformable model for vehicle recognition. In *Proceedings of the British conference on Machine vision (BMVC '95)*, pages 127–136, 1995. 20

[37] Y. Li, L. Gu, and T. Kanade. A robust shape model for multi-view car alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pages 2466–2473, 2009. 20, 32

[38] J. Bastian and A. van den Hengel. Computing surface-based photo-consistency on graphics hardware. In *Proceedings of Digital Image Computing: Techniques and Applications (DICTA '05)*, pages 76–, 2005. 20

[39] A. R. Dick, P. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *International Journal of Computer Vision*, 60(2):111–134, 2004. 20

[40] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40: 123–148, November 2000. 21

[41] P. F. Sturm and S. J. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. In *Proceedings of the British conference on Machine vision (BMVC '99)*, pages 265–274, 1999. 21, 63, 95, 104

[42] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Proceedings of the Eurographics Workshop on Rendering Techniques '98*, pages 93–104, 1998. 22, 86, 104

[43] H.Y. Shum, M. Han, and R.S. Szeliski. Interactive construction of 3d models from panoramic mosaics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pages 427–433, 1998. 22

[44] M. Wilczkowiak, P. Sturm, and E. Boyer. Using geometric constraints through parallelepipeds for calibration and 3D modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:194–207, February 2005. 22, 32

[45] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Transactions on Graphics*, 26(3), July 2007. 23

[46] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based facade modeling. In *Proceedings of ACM SIGGRAPH Asia 2008*, pages 161:1–161:10, 2008. 23

[47] S. El-hakim, E. Whiting, and L. Gonzo. 3D modelling with reusable and integrated building blocks. In *Proceedings of the 7th Conference on Optical 3-D Measurement Techniques*, pages 3–5, 2005. 23

[48] R. Ziegler, W. Matusik, H. Pfister, and L. McMillan. 3D reconstruction using labeled image regions. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP '03)*, pages 248–259. Eurographics Association, 2003. 23

[49] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based plant modeling. *ACM Transactions on Graphics*, 25(3):599–604, July 2006. 24

[50] M. Habbecke and L. Kobbelt. An intuitive interface for interactive high quality image-based modeling. *Computer Graphics Forum*, 28(7):1765–1772, 2009. 24, 63, 83

[51] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01)*, pages 433–442, 2001. 24

[52] T. Thormählen and H.-P. Seidel. 3D-modeling by ortho-image generation from image sequences. *ACM Transactions on Graphics*, 27(3):86:1–86:5, August 2008. 24

[53] B. Ward, S. B. Kang, and E.P. Bennett. Depth director: A system for adding depth to movies. *IEEE Computer Graphics and Applications*, 31(1):36 –48, Jan-Feb 2011. 24

[54] Autodesk 3ds max: A commercial 3d modeling package http://usa.autodesk.com/3ds-max/. 25, 62, 97

[55] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. SKETCH: an interface for sketching 3D scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96)*, pages 163–170, 1996. 25, 62

[56] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, pages 409–416, 1999. 25, 62, 86

[57] 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. 25, 62

[58] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2D-to-3D modeling. In *Proceedings of ACM SIGGRAPH Asia 2009*, pages 148:1–148:9, 2009. 25, 105

[59] Google SketchUp http://sketchup.google.com/. 25

[60] L. Olsen and F. F. Samavati. Image-assisted modeling from sketches. In *Proceedings of Graphics Interface (GI '10)*, pages 225–232, 2010. 25, 86, 97

[61] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3d models. *ACM Transactions on Graphics*, 22(1): 83–105, January 2003. 25

[62] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'07)*, November 2007. 29, 112

[63] Y. Baillot, D. Brown, and S. Julier. Authoring of physical models using mobile computers. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers (ISWC '01)*, pages 39–, 2001. 29, 114

[64] S. Kim, S. DiVerdi, J. S. Chang, T. Kang, R. Iltis, and T. Höllerer. Implicit 3d modeling and tracking for anywhere augmentation. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology (VRST '07)*, pages 19–28, 2007. 29

[65] P. Bunnun and W. Mayol-Cuevas. OutlinAR: an assisted interactive model building system with reduced computational effort. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR '08)*, pages 1–8, 2008. 29, 114, 116, 117

[66] J. Lee, G. Hirota, and A. State. Modeling real objects using video see-through augmented reality. *Presence: Teleoperators and Virtual Environments*, 11(2):144–157, April 2002. 29

[67] J. Mooser, S. You, and U. Neumann. Real-time object tracking for augmented reality combining graph cuts and optical flow. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR '07)*, pages 1–8, 2007. 29

[68] Q. Pan, G. Reitmayr, and T. W. Drummond. Interactive model reconstruction with user guidance. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR '09)*, pages 209–210, 2009. 30, 118, 124

[69] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, 2010. 30, 118

[70] Google Earth: A 3D map program http://earth.google.com. 26

[71] W. Niem and H. Broszio. Mapping texture from multiple camera views onto 3D-object models for computer animation. In *Proceedings of the International Workshop on Stereoscopic and Three Dimensional Imaging*, pages 99–105, 1995. 26

[72] Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics*, 23(3):584–591, August 2004. 28

[73] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Fitting multiple models to multiple images with minimal user interaction. In *Proceedings of the International Workshop on the Representation and Use of Prior Knowledge in Vision (WRUPKV)*, May 2006. 33

[74] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Hierarchical model fitting to 2D and 3D data. In *Proceedings of the Third International Conference on Computer Graphics, Imaging and Visualisation*, July 2006. 33

[75] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Building models of regular scenes from structure-and-motion. In *Proceedings of the British conference on Machine vision (BMVC '06)*, September 2006. 33

[76] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Rapid interactive modelling from video with graph cuts. In *Proceedings of Eurographics 2006*, September 2006. 33

[77] Voodoo camera tracker http://www.digilab.uni-hannover.de. 34

[78] E. T. Jaynes and Larry G. Bretthorst. *Probability Theory: The Logic of Science.* Cambridge University Press, 2006. 37

[79] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, page 199, 2003. 38

[80] P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, March 1964. 41

[81] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection.* John Wiley & Sons Inc., 1987. 41

[82] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951. 43

[83] D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963. 51

[84] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace: rapid interactive scene modelling from video. *ACM Transactions on Graphics*, 26, July 2007. 63, 64, 96

[85] M. Shpitalni and H. Lipson. Classification of sketch strokes and corner detection using conic sections and adaptive clustering. *ASME Journal of Mechanical Design*, 119:131–135, 1995. 87

[86] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Interactive 3D model completion. In *Proceedings of Digital Image Computing: Techniques and Applications (DICTA '07)*, December 2007. 96

[87] R. Schmidt, A. Khan, G. Kurtenbach, and K. Singh. On expert performance in 3D curve-drawing tasks. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM '09)*, pages 133–140, 2009. 101

[88] A. van den Hengel, R. Hill, B. Ward, and A. Dick. In situ image based modelling. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) 2009*, October 2009. 112

[89] J. Bastian, B. Ward, R. Hill, A. van den Hengel, and A. Dick. Interactive modelling for AR applications. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) 2010*, October 2010. 112

[90] G. A. Lee, G. J. Kim, and M. Billinghurst. Immersive authoring: What you experience is what you get (wyxiwyg). *Communications of the ACM*, pages 76–81, 2005. 116

[91] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. 118, 119

[92] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, August 2004. 120

[93] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28:24:1–24:11, July 2009. 136

[94] A. Saxena, M. Sun, and A. Y. Ng. Learning 3-d scene structure from a single still image. In *Proceedings of the ICCV workshop on 3D Representation for Recognition (3dRR07)*, 2007. 136