Reconstructing Surfaces from Images

Chapter I

Photographs capture the world from a particular point-of-view by projecting the 3D world onto a 2D image, irrevocably losing information in the process. Estimating this lost information remains a fundamental problem in *computer vision*. There are a number of approaches to this problem. Feature-based methods, for example, work first in image space to decompose the reference images into a set of 2D features before they are triangulated in three-space. Another approach works in scene-space by evolving a shape hypothesis until its projection recreates the reference images when imaged with the reference camera parameters. Surfaces that can recreate the input images are said to be *photo-consistent*. Although determining a shape's consistency is a powerful mechanism of verifying its fitness, photo-consistency alone does not resolve ambiguity inherent in recovering lost projective information.

Most photo-consistency based approaches do not use prior information about the scene, and must therefore resolve the projective ambiguity by finding the *largest* shape consistent with all reference views. While this is an appropriate solution given no other information, this approach cannot recover regular geometric structure. This limitation can adversely affect the reconstruction's appearance when reprojected from novel views for certain types of scenes. Rather than simply finding the largest shape, we propose an approach that recovers 3D shapes from multiple photographs subject to geometric constraints. We use a parameterised description of the surface with variables to describe the dimension, orientation, and position of 3D shapes, such as boxes and cylinders. Our goal is to find the parameter values that deform this parametric surface such that it recreates the reference images while satisfying the geometric constraints imposed on the solution space.

There are a number of applications for scene reconstruction, driven by the desire to store and manipulate a representation of the world, that would benefit from a regular geometric structure imposed by our approach. One application of computer vision is motion-picture post-production, where films are increasingly compositing computer generated images with live action to realize images that are not possible to film practically. 'The Matrix[1],' for example, used a modelling system based on Debevec's Façade [15] to reconstruct film stages. The computer models were used to generate novel, realistic views of the stage from points of view that were infeasible to capture with a practical camera. Some other applications which require a 3D representation of the world include:

- virtual touring;
- generating new views of scanned medical data; and
- reconstructing crime scenes for forensic analysis.

A *realistic* representation of the scene is required in most cases. To this end, many approaches focus on recovering shapes that can recreate the reference images on the assumption that the re-projection of the surface from new views will be convincing if the input can be faithfully recreated.

1.1 From features to generative models

Reconstruction is the problem of estimating 3D structure of a scene that is visible in a number of reference images. This problem has been extensively studied with different approaches making various assumptions about the problem. These assumptions includes limitations on the number of cameras, their parameters and configuration; and the geometry, texture and reflectance of the scene. There is not yet a single approach that can reconstruct arbitrary scenes with unknown camera parameters.

One approach involves first deconstructing the input reference images into a set of 2D features, rather than directly considering the image as a 2D light-field. The projective relationship between the world and the cameras can be estimated by correctly identifying the relationship between features in world space and their corresponding projection into the set of reference images. These correspondences are used to estimate the camera parameters so that the 2D features can be triangulated from different images into three-dimensions. This approach has been successful for a number of applications. Given certain assumptions about the camera rig, this process can be automated and made robust through consensus [20] and bundle adjustment algorithms [66].

[1]**The Matrix** © 1999, Warner Brothers.

Feature-based approaches critically rely on the presence and accurate localisation of image features. This has a number of implications. Firstly, the reference images must contain sufficient number of identifiable features. Depending on the type of algorithm, this may prohibit, for example, scenes with smoothly varying or repeated texture. Secondly, a small base-line is often assumed to facilitate matching features across images by limiting the displacement between projections of a single 3D feature. The key problem of feature-based methods, however, is that their scene model is limited by the types of features they extract from the images. Methods that localise points in images, for example, can only recover a sparse point-cloud in scene-space. Although this model often contains useful 3D structure, it generally does not contain sufficient information to generate realistic images from novel view-points.

In contrast to feature-based methods, many approaches directly build surface models in scene-space. In general, these approaches generate a succession of surface hypothesises which are evaluated by considering their projection in the reference views. The goal of these methods is to find a model that recreates the reference images when imaged by the reference cameras; such surfaces are said to be *photo-consistent*. The motivation behind finding photo-consistent surfaces is that they contain sufficient detail to generate realistic images of the reconstruction from novel view-points.

Appearance-based methods must maintain a model which can recreate the reference images when imaged by the reference cameras. A variety of surface models have been proposed. One class of approaches refines a point-sampled shape hypothesis until every point is consistent with the reference views. Beginning with a shape that is guaranteed to encompass the scene, the hypothesis is iteratively refined by removing inconsistent points. This approach can produce very convincing results because the entire image is used, rather than a set of features.

Another approach recovers polyhedral scenes subject to architectural constraints. Earlier systems required a user to manually identify and correlate 3D primitives to their projections in the reference images, but later systems are capable of automatically recovering polyhedral meshes in certain circumstances. These automated systems typically use assumptions about the type of architecture, such as using prior information on expected shape [17, 16]. While some approaches use texture to optimise their results with respect to the reference images, they typically require features to boot-strap the process. These approaches are examined in more detail in Chapter 2.

1.2 Fitting geometry to images via inverse rendering

Our approach reconstructs polyhedral surfaces from multiple, wide base-line, reference images subject to specific geometric constraints. A scene-graph is used to describe these constraints by relating 3D primitives with unknown but parameterised transformations. The user provides the scene-graph to reduce the ambiguity inherent in projecting 3D geometry into images. The reconstruction problem then becomes one of finding the optimal parameter values given the set of reference images.

Although the user must provide a parameterised scene-graph, our system uses a Bayesian learning process to draw inferences about the parameters' values. The likelihood of a given hypothesised parameter vector is given by the hypothesised surface's photo-consistency, which is evaluated by comparing synthetic views of the hypothesised surface against the corresponding reference images. The likelihood that the hypothesised surface represents the reference surface is given by the difference between these two images.

Occlusion makes the photo-consistency function highly non-linear, complicating the problem of finding the maximum *a posteri* estimate. To this end, we use Monte Carlo Markov Chains to draw inferences about the parameter distribution by generating a series of shape hypotheses. This process requires evaluating the surface's likelihood over a potentially large number of hypothesised parameters. While determining photo-consistency is a mathematically straight-forward process, it is also very computationally expensive.

Our approach measures the photo-consistency likelihood using graphics hardware because the performance of graphics processors outstrips the peak theoretical performance of many general-purpose processors [40]. Although graphics hardware is very adept at rendering a textured triangle mesh, a key part of the photo-consistency likelihood lies in estimating the structure's 'optimal' texture given the hypothesised parameter vector. This texture is applied to the surface as the hypothesised scene is rendered. The result is a synthetic image that is compared with the corresponding reference image to yield the hypothesised scene's photo-consistency likelihood.

1.3 Outline

Our argument toward a scene-graph photo-consistency metric is motivated by describing existing work on recovering models from multiple images. A trend toward using global photo-consistency metrics and probabilistic methods is shown in Chapter 2. A number of approaches that use graphics hardware are also described. This review indicates that while photo-consistency is used on a variety of surface models such as planar models, unconstrained triangular meshes, and voxel surfaces, it has not yet been used on constrained polyhedral meshes.

Chapter 3 describes our system for finding polyhedral surfaces subject to geometric constraints. It describes the frame-work for evaluating the scene likelihood and the challenges involved in computing photo-consistency on graphics hardware.

A photo-consistency based likelihood suitable for graphics hardware is presented in Chapter 4. This is tested on synthetic and real data, concluding that it is remarkably robust even with exceptional image noise, although the likelihood can be adversely affected by occlusion.

Chapters 5, 6 and 7 describe a surface-based consistency likelihood designed to address the occlusion problems identified in Chapter 4. A system for building surface textures from each view is described in Chapter 5. These textures are combined in a process described by Chapter 6 and tested for photo-consistency in Chapter 7. Synthetic and real tests demonstrate that although this approach is not affected by the same problem described in Chapter 4, its likelihood metric is still biased.

Chapter 8 concludes by relating our system to previous approaches and indicates areas for future work.

Prior Work on Surface Reconstruction

Chapter II

Reconstructing 3D structure given a set of 2D images is a fundamental problem in computer vision. One approach to this problem involves extracting features from the images, rather than using the image colour directly. These 2D features are matched across different images and triangulated to infer 3D structure. While this approach has proved successful in estimating camera parameters and 3D feature localisation, its scene model has limited applications.

Rather than recovering points or lines in scene-space, a number of methods have investigated reconstructing surfaces. Some approaches recover planes by minimising the distance between the hypothesised planes and points recovered from feature-based stereo, while others fit the edges of quadrilaterals to images [15] or maximise the correlation between two images [22, 2]. Although planes are not suitable for general scenes, they enforce geometric constraints which lead to plausible reconstructions for planar scenes.

A more general approach uses volumetric samples (or '*voxels*') to recover arbitrarily complicated scenes [51, 29, 13, 19]. These approaches measure photo-consistency on a per-voxel basis, but must use a threshold to overcome difficulties with calibration and surfaces that do not correctly fit the assumed reflectance model. A disadvantage of this threshold-based consistency is that it is typically used to irrevocably change the hypothesis. Unfortunately, this can completely eliminate a model when a mistake is made. More recent approaches use improved sampling [28], statistical consistency checks [61, 8] and optimisation [54] to overcome some of these problems while still using a voxel framework.

A number of approaches represent general scenes with polyhedral meshes [56, 68, 26, 27] to overcome the limitations of voxel models. The advantage of these approaches is that it avoids the problems with voxel resolution; the reconstructions are spatially coherent (because the surface is continuous); and the model is better suited to a global optimisation approach. Photo-consistency is typically modelled as an error-term in these approaches; but this can lead to short but inconsistent surfaces because photo-consistency is integrated over the surface. These problems are partially addressed by using heuristics [68] and silhouette constraints [27].

2.1 Geometry-based reconstruction

Habbecke reconstructs textured planes from calibrated images via predictive correlation [22]. Given a manually identified set of pixels in one view, the corresponding plane in scene-space is estimated by hypothesising a 3D plane and matching its projection in the other images. Each plane hypothesis defines a homography that maps the pixels in the source reference images to the comparison images. The sum of squared differences (SSD) of corresponding pixels mapped by this homography is used as a measure of fitness and an iterative scheme is used to find the plane equation with minimum cost.

Bailiard also recovers planar facets but uses an automatic approach to sweep planes about lines in the image [2]. Lines are first detected in the image and are triangulated into scene-space. 'Half-planes' are connected to each line, parameterised by a rotation about its anchor line. The value of this rotation parameter is estimated by maximising the correlation score between the plane and the other reference views. Although each plane in the image is estimated independently, coincident lines and planes are automatically grouped to form more complete objects.

Debevec reconstructs multiple planes at once, but requires the user to build a scene from a set of pre-defined building blocks in a reconstruction system called *Façade* [15]. The user constructs the scene-graph by placing blocks on the image and defining spatial relationships by connecting blocks with parameterised translation and rotation transformations. Scale is encoded by defining 'structure variables' to define edge length within the building blocks. The Façade structure is particularly useful for architectural scenes because a carefully considered scene-graph is able to describe a relatively complicated surface with very few parameters.

The Façade scene-graph reduces the reconstruction problem to one of estimating the values for the structure and camera parameters that best recreate the reference views. The user is required to match edges in the reference images with 3D edges along the scene blocks. The edges are used to determine a measure of fitness, given by the disparity between the edges of the predicted scene parameters and the corresponding edges marked by the user in the reference image. The disparity of a point on the

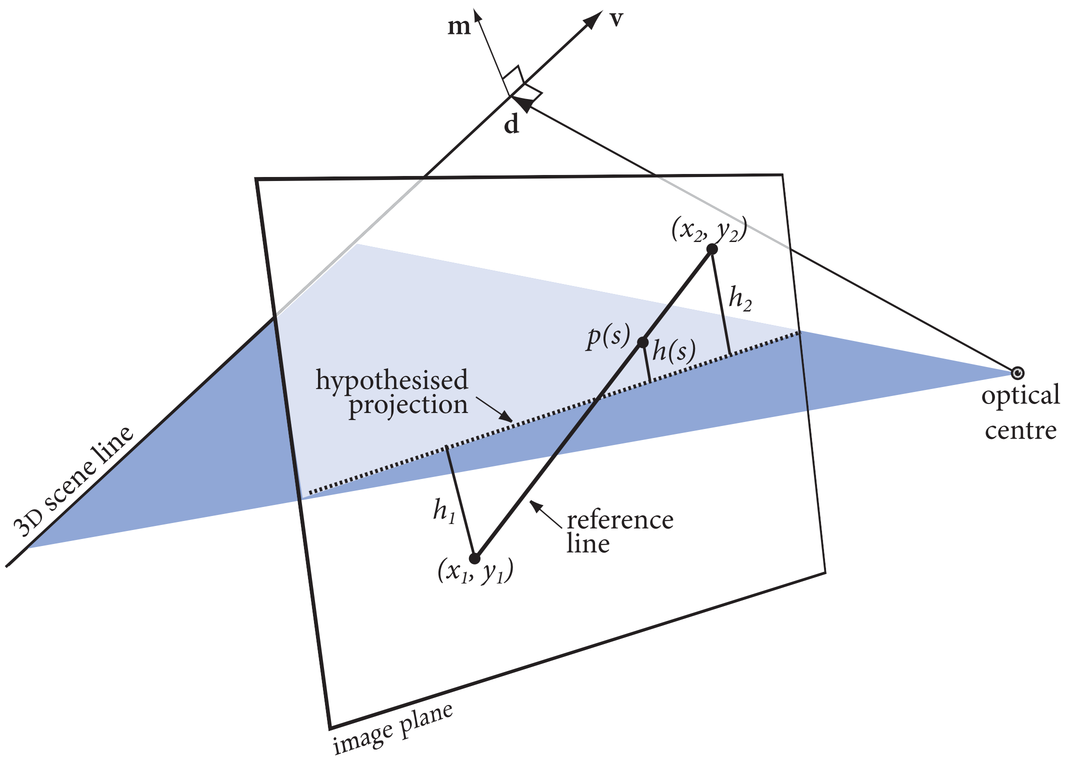


Figure 2.1: Line-based re-projection error is given by the distance $h(s)$ between the marked, reference edge and the hypothesised projection.

geometry edge in one reference view is defined as the orthogonal distance from the point's projection to the reference line. The disparity of a point $\boldsymbol{p}(s)$ parameterised by s on the reference line is denoted $h(s)$ in Figure 2.1. The re-projection error for the edge is then $\sum_j \int h_j(s)ds$ over all reference images j. The hypothesised scene lines that minimises the disparity is found by estimating camera rotation and then camera translation and model parameters.

2.2 The visual hull

An approximation of an object's 3D shape can be recovered using only its silhouette in each image [30]. Because the projection of every point on the object's surface must lie within its silhouette in all views, the 3D surface must lie in the *visual cone* formed by back-projecting the silhouette into scene-space. The visual cone for one image is

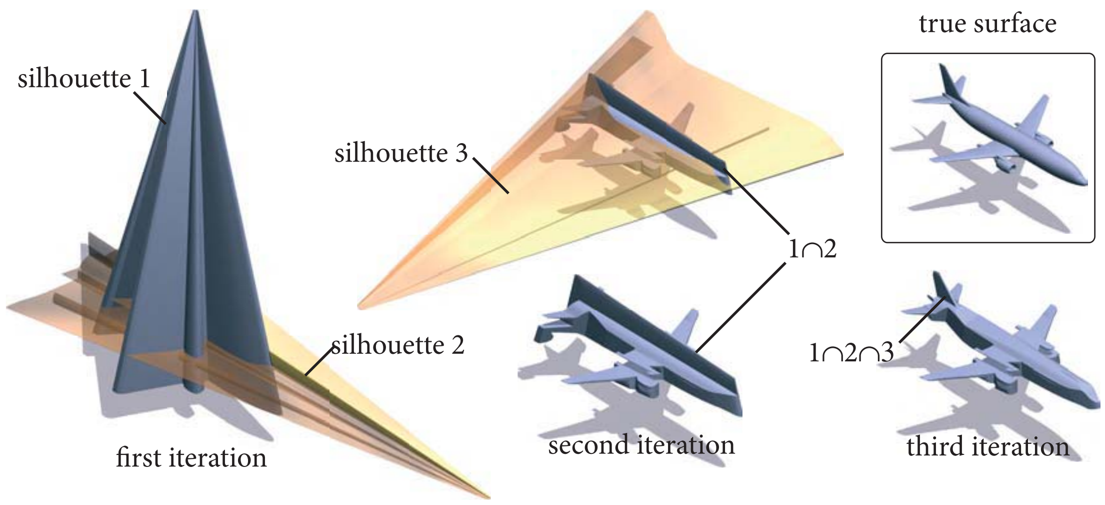


Figure 2.2: The visual hull is constructed by intersecting the silhouette cone from each reference image.

defined by the surface silhouette representing the cone's cross-section, and the camera's optical centre as its apex. Each new view increasingly constrains the shape, leading to the *visual hull* constructed by the intersection of each camera's visual cone (see Figure 2.2). Although the visual hull is the tightest bound on the reference surface given only its silhouette, it cannot recover concavities.

There are a number of ways of representing and constructing the visual hull. One approach is to represent the scene with voxels. Each voxel is tested for inclusion in the visual hull by projecting it into each segmented image; voxels that are not contained within silhouette of *all* images are marked as free space. Rather than exhaustively testing every voxel in a sub-space, Potmesil [45] and Szeliski [62] represent the surface as an oct-tree where each node represents a certain subset of the scene-space. The reconstruction is estimated by discarding nodes whose projection falls completely outside of at least one silhouette. Nodes are subdivided until a pre-determined resolution is reached if their volumes cross a silhouette curve.

Although voxel-based approaches can efficiently deal with arbitrarily complicated visual hulls, their reliance on point-sampling is both time consuming and prone to aliasing. Matusik proposed representing the surface as a polyhedral mesh to address these problems [35]. The intersection of two visual cones can be reduced to a 2D operation by projecting segments of one cone into the image of another and clipping the projected segment against the second camera's silhouette. Because the segments are actually rays in scene-space, the results are easily 'lifted' back into three-dimensions. While this approach requires computing the intersection of two arbitrary polygon meshes, processing the visual cones in pairs makes the problem significantly more efficient than computing the intersection of an arbitrarily complicated mesh.

An explicit visual hull is not necessary if the primary purpose is to generate synthetic images. Matusik synthesises novel views by implicitly computing the visual hull in image space [36]. A ray is constructed for each pixel in the virtual camera. Each ray is clipped against each silhouette in the reference images to yield a list of segments in scene-space that span the visual hull for that pixel. The segment closest to the virtual camera is then projected into the nearest reference image to define the colour for that pixel.

These visual hull algorithms all assume the silhouette can be successfully extracted from the image. The difficulty with using silhouettes as a hard constraint is that segmentation errors critically undermine the visual hull's reconstruction. Rather than first explicitly recovering the silhouette, Snow uses graph cuts to compute a voxelised visual hull by minimising the energy of incorrectly assigning binary labels to voxels [58]. The graph is defined by assigning one node per voxel and connecting each node to its immediate neighbours, the sink, and source nodes. The weight connecting adjacent nodes is designed to penalise different label assignments between neighbours, thereby adding a smoothness term. The weight connecting each node to the source node is a function of the voxel's silhouette likelihood measure calculated by a background subtraction metric from the voxel's projection in all images. The solution that best partitions the voxel space into exterior and interior sub-spaces is found using graph cuts.

2.2.1 *Computing visual hulls on graphics hardware*

Li extended Matusik's silhouette clipping approach [36] to use graphics hardware[1] to clip an entire silhouette at once, rather than considering only the back-projection of a single pixel at a time [31]. The object's silhouette is approximated in each image by a polygon. Each polygon edge is back-projected into scene-space and rendered into the virtual view, while a texture from each reference camera is back-projected onto the extruded

[1]Graphics hardware terminology is described in Appendix B.

edge. Each of these textures is an α−map that corresponds to the object's segmentation in the particular view; texels in the map have have $\alpha = 1$ if the texel corresponds to a pixel in the object's silhouette. Back-projecting the segmented textures onto the extruded edge is equivalent to projecting each fragment into each reference view. The α-channel of the corresponding texels are multiplied together; fragments where $\alpha = 0$ are rejected, implementing the constraint that a scene-point must be within all silhouettes to be within the visual hull.

Since Li's initial implementation uses multi-texturing to combine the silhouette responses, the number of reference cameras it can use is limited by the number of texture units in the graphics-pipe. This limit is overcome by using the stencil buffer to track cone intersections in Li's multi-pass rendering extension [32]. Only one segmented texture is back-projected onto an extruded edge at a time, which computes the intersection of two visual cones. The results are accumulated in the stencil buffer and repeated with the segmented texture from each camera in turn to compute the intersection with respect to all views. The α−test is used to reject fragments with $\alpha = 0$; the stencil buffer is incremented by fragments that pass this test. The extruded edge is rendered once per reference image into the virtual view; after n views have been rendered, the stencil values of n denote pixels that back-project to the visual-hull.

Hornung reconstructs the visual hull by using graphics hardware as a multiple-instruction/multiple-data machine rather than 'directly' working in scene-space [25]. Textures are used to represent an arrays of voxels, where each texel represents a voxel position encoded by its its RGB colour. Each texture is therefore a 2D serialised representation of a 3D voxel space which is processed by writing it to the frame-buffer. The visual hull is partially recovered by projecting each fragment into a segmented image and writing the Boolean result with respect to one camera into the frame-buffer. The intersection of these segmented maps is computed on the processor.

2.3 The photo-hull

The visual hull is the tightest bound on an surface given only its silhouette, but it often requires a large number of voxels to recover subtle details and it can *never* recover concavities. Furthermore, heuristics must be used to map colour to the hull's surface; Li's graphics approach [31], for example, uses view-dependent textures similar to Debevec's approach with Façade [15]. Although textured visual hulls can look convincing if the surface is convex, their inability to model concavities can introduce significant visual artifacts when a synthetic view deviates significantly with the reference cameras. Rather than relying on the surface's silhouette alone, a class of approaches seek to recover a surface that can recreate the reference images when the shape is reprojected with the reference camera parameters. Such surfaces are said to be *photo-consistent*. The union of all photo-consistent shapes is defined as the *photo-hull* and is a tighter bound on the surface than the visual hull.

While voxels in the visual hull are binary labeled (either they project within all silhouettes or they are excluded), voxels in the photo-hull must also be labeled with a colour that is consistent with all images. This consistency describes a constraint on each voxel's colour. Occlusion must be taken into consideration, however, so that only eligible reference views can affect a voxel's colour. To this end, a number of methods maintain a conservative estimation of occlusion. They begin with a super-set of the scene and refine this estimate by removing voxels that are not photo-consistent.

Photo-consistency is evaluated by measuring the disparity between a voxel's hypothesised reflectance and its projection in each image. Since a voxel represents a finite volume of the scene, its projection masks an area in the image which is greater than one pixel. Determining a voxel's hypothesised reflectance from its footprint is a voxel sampling problem. Some approaches consider the voxel's centroid as the only point of interest, while others use quadrilaterals [61] or box filters [8]. The sampling method has repercussions on determining photo-consistency and ultimately on the quality of the reconstruction.

Irrespective of the chosen sampling method, any photo-consistency criterion must allow for noise in the image and camera parameters. The most straightforward approach to this problem has been to allow for a consistency threshold [51, 29]. Unfortunately, the model's accuracy critically depend on a carefully chosen threshold. Some solutions to this problem treat photo-consistency as an optimisation problem [54, 65] or use a statistical framework [1, 6, 8] rather than relying on a threshold that can irrevocably damage a hypothesised model.

2.3.1 *Feature-based refinements*

Li recovers fine detail in the visual hull using stereo correspondence [34]. Computing correspondences between wide base-line images can be error prone because a surface point may project to significantly disparate points in the two images. Li's approach

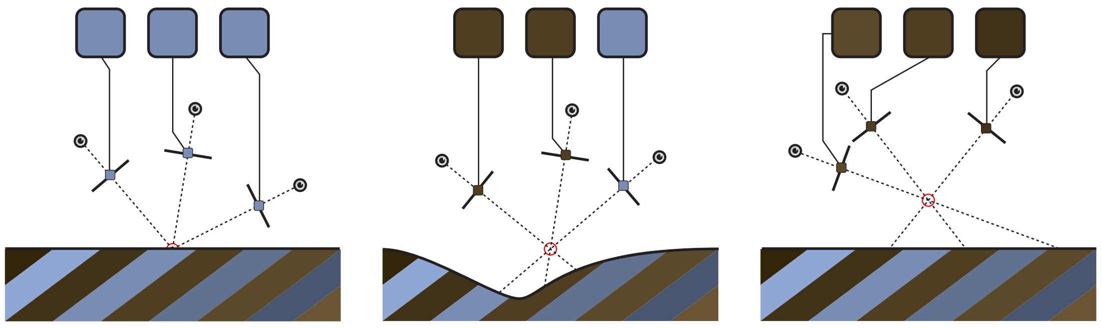


Figure 2.3: Photo-consistent points have a similar colour in all views where they are visible.

solves this problem by using a visual hull to estimate the disparity range for a given view using the bounding box of the visual hull. The visual hull is rendered into the target view to find the lower disparity bound on a per-pixel basis. This disparity range limits the search space when searching for correspondences. Having identified correspondences, the depths of neighbouring 3D points are interpolated to generate a per-pixel depth-map.

Stereo correspondence has been used by Collins to reconstruct features through consensus, but without reconstructing the visual hull [11]. In this approach, features are extracted from multiple, wide base-line images, and a plane is swept through the scene-space to collect 'votes' for likely positions of the features. Votes are collected by a cells distributed over the plane; each cell counts the number of back-projected feature-rays that it intersects. Cells with sufficient votes are assumed to correspond to features identified in the image, and therefore the 3D position of the cell becomes part of the reconstruction. The plane is swept through the scene-space to test different 3D positions and build a complete model.

Seitz uses a similar sweep-based approach to recover a dense, photo-realistic reconstruction from a set of strongly calibrated cameras [51]. Rather than identifying features in image-space, reconstruction is treated as a problem of assigning colours to a set of voxels in such a way that the surface recreates the reference images when imaged from the reference camera positions. The visual hull is also capable of generating the reference images by back-projecting the reference view closest to the virtual camera onto the surface, but this can only recreate the reference images because a voxel's colour depends on the viewing angle. In contrast, Seitz's voxel colouring approach is able to recreate all reference views from a reconstruction with a *single* colour assignment.

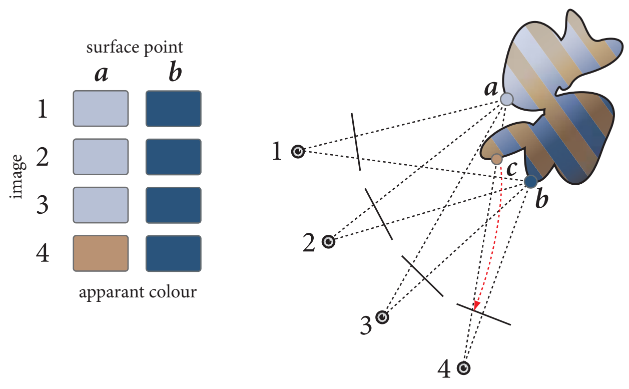


Figure 2.4: Points on the reference surface will appear inconsistent unless occlusion is used to decide which cameras can vote for a point's consistency.

The principle behind voxel colouring is that diffuse points on the reference surface will appear to have an identical colour in all views where it is visible. This is the *photo-consistency constraint*, and is an important property held by all surface points Points that are not on the surface are likely to have different apparent colours because their projection in each image actually corresponds to different surface points. Such points cannot be in the scene because a single, diffuse surface point cannot reflect different colours (see Figure 2.3).

Estimating a voxel's photo-consistent colour requires first determining the set of views in which it is visible. Occlusion must be modelled because a voxel's apparent projection may correspond to different scene-points: even for voxels that are on the reference surface. These projections would adversely affect the voxel's estimated colour if the occluded observations are not removed. This problem is illustrated in Figure 2.4, where the colour of ***a*** appears different in one of the views because the projection of ***a*** in the fourth image actually corresponds to the point ***c*** on the reference surface. The photo-consistency of point ***a*** would be undermined if the fourth image is allowed to participate in estimating the colour for ***a***.

While a voxel's colour can be estimated if it is known which views can participate in its estimation, determining a voxel's visibility requires determining the very surface that the algorithm is trying to recover. Furthermore, there are a multiplicity of solutions that can regenerate a finite set of reference views [51]. An example of this ambiguity is illustrated in Figure 2.5 where there are *five* voxel colour assignments that are consistent with two reference images.

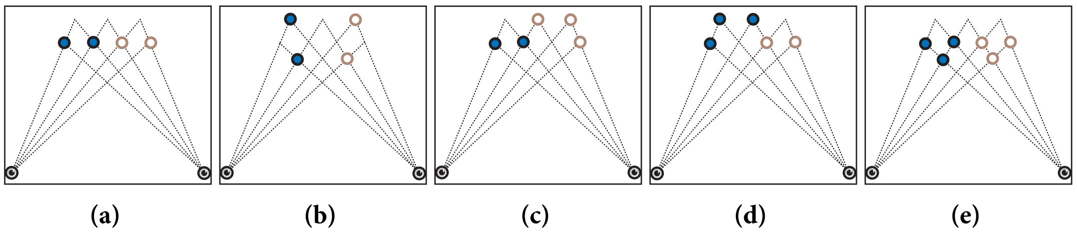
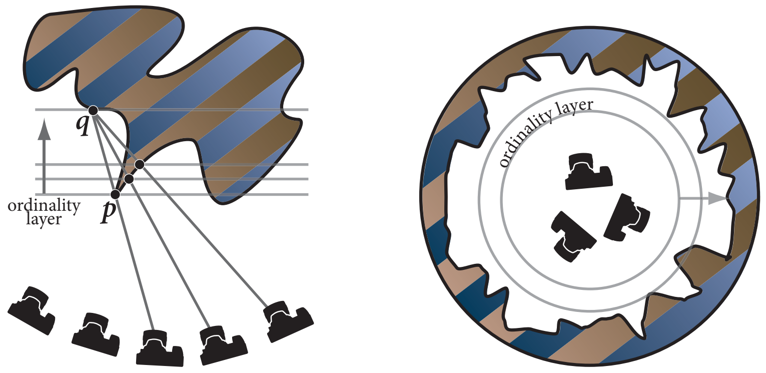


Figure 2.5: Photo-consistency is a property held by an infinite number of hypothesised surfaces.

Seitz addresses both problems by limiting scenes to those that can be partitioned by an ordinality constraint. This constraint is described by a function $f(\cdot)$ such that $\boldsymbol{p}$ occludes $\boldsymbol{q}$ with respect to *all* reference views iff. $f(\boldsymbol{p}) < f(\boldsymbol{q})$. While this prohibits completely general camera arrangements, it is nonetheless suitable for a broad range of practical configurations. Two compatible configurations are illustrated in Figure 2.6 where the scene can, in both cases, be partitioned into a series of '*ordinality layers*'. A suitable partitioning function for the first example is a plane; points on the plane $z = k'$ will only be occluded by points on the plane $z = k$ when $k < k'$. The scene in the second example can be partitioned by a series of spheres with increasing radius.

The ordinality constraint defines a depth-compatible traversal though the scene. It greatly simplifies the colour-consistency check by assigning colours to all voxels on a given ordinality layer before moving onto the next. Since the layers are defined by the partitioning function $f(\cdot)$, all potential occlusions for a given voxel $\boldsymbol{q}$ are from voxels $\{\boldsymbol{p} : f(\boldsymbol{p}) < f(\boldsymbol{q})\}$. Visiting all voxels $\{\boldsymbol{p}\}$ before considering $\boldsymbol{q}$ guarantees that all occlusions have been identified and therefore means the colour-consistency check does not include erroneous observations of the surface.

The photo-consistency constraint is used to identify voxels that are not on the reference surface. If the surface is assumed to be perfectly diffuse, then a surface point will have a similar appearance in all reference images. Computing a consistent reflectance model in this case is a problem of minimising the disparity between the model and its observation in each reference view. Since a diffuse surface scatters light equally in all directions, a voxel's average observation defines the colour that is most consistent with the reference images, and its observed variance is therefore a measure of its photo-consistency. The projection of a voxel that spans the pixels $\mathcal{P}_i$ in image r_i is



(a) Planar ordinality layer. **(b)** Spherical ordinality layer.

Figure 2.6: Two classes of depth-compatible scene configurations.

consistent with the reference images if

$$Var(\bigcup_{j=0}^{n} \mathcal{P}_j) < \tau \tag{2.1}$$

where τ is a consistency threshold designed to compensate for small inconsistencies due to sensor noise and voxel aliasing.

A voxel's visibility must be determined in order to measure its photo-consistency. This involves determining that there is no occluding voxel on the ray between the camera's optical centre and a given voxel. Rather than traversing the space repeatedly for each voxel, Seitz uses an occlusion mask defined over each reference image. The projection of consistent voxels are added to the occlusion mask as they pass the consistency check. Testing the visibility of a voxel only requires checking that the occlusion mask is clear, since this indicates that the ray between the voxel and the cameras optical centre does not yet contain a surface point.

The voxel colouring algorithm sweeps through the voxel space in the order defined by the ordinality constraint, colouring voxels that pass the consistency test and removing those that fail. It deals with the shape ambiguity by sweeping backward through the scene, implicitly finding the largest photo-consistent surface. Since all points that cannot recreate the reference images are removed, the result is a set of coloured voxels that recreate the reference images when imaged from the reference camera parameters.

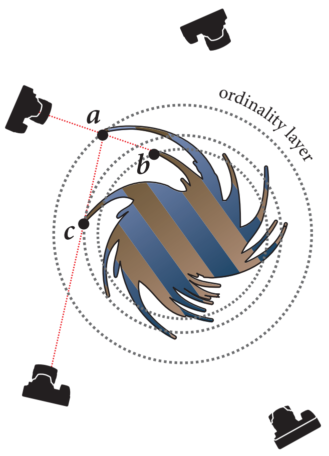


Figure 2.7: Arbitrary configurations can break the ordinality constraint.

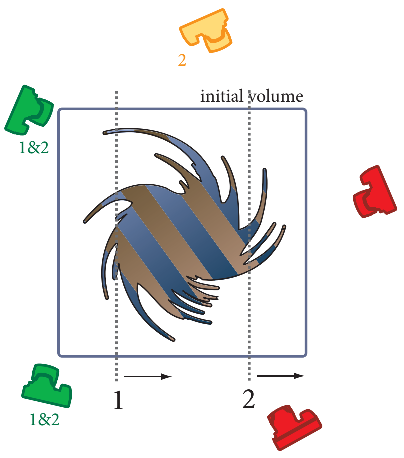


Figure 2.8: Space carving sweeps along each scene axis.

2.3.2 *Space carving*

Kutulakos extended Seitz's Voxel Colouring algorithm by allowing camera placements that break the ordinality constraint [29]. An example of this case is illustrated in Figure 2.7, where $\boldsymbol{a}$ occludes $\boldsymbol{b}$, suggesting that $f(\boldsymbol{a}) < f(\boldsymbol{b})$. However, $\boldsymbol{c}$ also occludes $\boldsymbol{a}$, but $f(\boldsymbol{b}) < f(\boldsymbol{c}) < f(\boldsymbol{a})$. Kutulakos's sweeps through the scene space from different directions. Each direction defines a new ordinality function which partitions the scene into a depth compatible order, allowing cameras on the 'trailing' subspace to vote for the consistency of a given voxel. This adaptive voting scheme is illustrated in Figure 2.8. Like Seitz's work, this approach finds the maximal photo-consistent hull.

Kutulakos' approach defines an ordinality constraints for subsets of the cameras, but it does not make use of all available information. Figure 2.9 illustrates the problem of partitioning the set of reference cameras according to the ordinality layer. In this case, one voxel appears photo-consistent from two different sweep directions, but each sweep direction yields two different colours. Culbertson's Generalised Voxel Colouring addresses this problem by using *all* of a voxel's hypothesised observations to recover shape [13]. All unoccluded projections of a voxel are used to determine a voxel's colour, even from cameras that would be considered ahead of the sweeping plane in Kutulakos's method. The cost of Culberton's generality, however, is that it requires a more complicated visibility check than the occlusion mask used by both Seitz and Kutulakos.

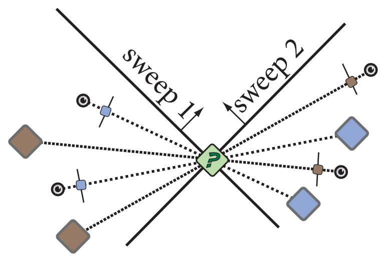


Figure 2.9: Partitioning cameras can include inconsistent voxels.

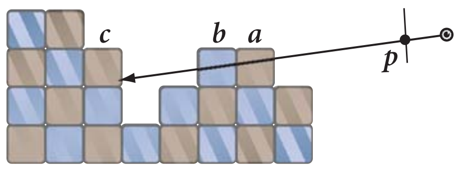


Figure 2.10: Removing voxels can reveal voxels that are not adjacent.

Eisert proposed a multi-hypothesis approach that is able to overcome the partitioning problem illustrated in Figure 2.9 [19]. A voxel's colour hypothesis is proposed by pairwise comparison of each voxel in two images. Each observation is added to a set of colour hypotheses if its projection in both images are similar. Because the surface geometry is not considered when generating voxel hypothesis, voxels are likely to have invalid colour hypotheses. Each reference view is considered in turn to remove invalid voxel hypotheses.

Like space carving, Eisert's approach begins with an initial surface hypothesis. Each visible surface voxel is compared with its projection in the given reference view, and any hypothesis that fails the consistency criterion is removed. A voxel that loses all its colour hypotheses is marked as free-space. While this approach is algorithmically different from Culbertson's Generalised Voxel Colouring, it is similar in that all views are used to establish a voxel's photo-consistency.

All space carving approaches require a conservative estimate of occlusion and therefore rely on an initial surface hypothesis. While defining an initial surface is straightforward for various scene configurations, it is not clear how to initialise the reconstruction for *ad hoc* camera placements. While Figures 2.11(a) and 2.11(b), for example, are common space carving layouts with a straightforward initialisation, the scene depicted in Figures 2.11(c) and 2.11(d) is more difficult to initialise.

One solution might involve carving out areas around the cameras, as illustrated in 2.11(c). Unfortunately, this initial solution would yield a trivial photo-consistent scene. Because the consistency-criterion critically depends on more than one camera voting for the consistency of a voxel carving areas around the cameras, in the case of 2.11(c), means that the hypothesised surface is visible only to the enclosed camera. Adopting a

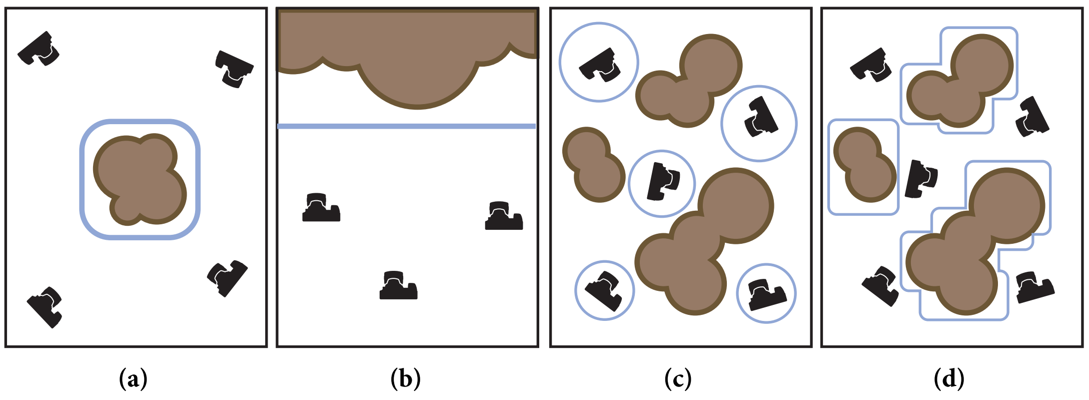


Figure 2.11: Various scene configurations and the envelopes used to initialise space carving. The envelope is denoted by the blue line surrounding the reference scene.

scene super-set model depicted in Figure 2.11(a) may yield an initial surface similar to Figure 2.11(d), but this requires a large degree of user input to ensure the hypothesised surface is sufficiently visible and does not cut into the 'true' surface.

While Space Carving is applicable for relatively constrained scenes, a prohibitive number of voxels is required for scenes that include foreground and distant background elements. Slaubaugh proposed a volumetric warp that uses '*spatially adaptive voxels*', which models surfaces further from the camera with less resolution and is capable of dealing with scenes stretching to infinity [55]. This volumetric warp is similar to Saito's projective voxel space, which uses epipolar lines to ensure voxels project to the same number of pixels, independent of their distance from the camera [49].

2.3.3 *Visibility*

Space carving must exhaustively test each voxel for consistency until the hypothesised shape converges. This requires testing the visibility of each voxel with respect to the continually evolving surface; determining its projection in each image; and evaluating the consistency of the observations. Checking whether a voxel is visible requires scanning the volume to find occluding voxels; but this can be quite time consuming. Rather than scanning the volume, Seitz and Kutulakos maintain a bit-mask over the ordinality layer to track which areas of the image that have yet to be attributed to voxels [51, 29].

The image-based occlusion tracking cannot work for more general approaches that do not use ordinality layers to partition the scene. Culbertson proposed two algorithms to monitor voxel visibility so that only surface voxels are considered [13]. The first approach uses an item buffer for each camera to track the identity of surface voxels. This buffer stores the identifier and distance to the camera of the closest voxel that projects to a given pixel. It is generated by rasterising voxel identifiers, changing the item and depth buffers only if the given voxel is closer than the current depth stored under its projection.

The item-buffer is computationally expensive to generate, and it must be updated when a surface voxel is carved. Removing a surface voxel does not always reveal adjacent voxels; new voxels can be exposed when holes are punched through the surface. This is illustrated in Figure 2.10, where removing voxel ***a*** exposes an adjacent voxel ***b***; but removing ***b*** exposes a non-adjacent voxel ***c***. Rather than storing just the closest voxel in the item buffer, layered depth images are used to store *all* surface voxels that project to a given voxel. Storing the identifiers of all voxels that project to a given pixel allows the algorithm to update visibility even when a remote surface voxel is revealed.

Although Culbertson uses an indexing system to track visibility, every voxel must be tested for consistency. Several methods have considered ways to reduce the required number of consistency checks. Montenegro avoids evaluating the photo-consistency of large amounts of free space by beginning the carving process from the visual hull [37]. Although an inclusion test is still required for the visual hull, this is easier to compute than determining each voxel's photo-consistency. Prock and Dyer employ a coarse-to-fine reconstruction to avoid exhaustively testing the photo-consistency of empty space at high resolutions [18, 46]. Their approach carves a series of models at increasing resolutions with progressively tighter tolerances. The effective voxel resolution is iteratively increased by subdividing and eroding the free space. One of the challenges of this approach is that it requires careful control of the threshold at multiple resolutions.

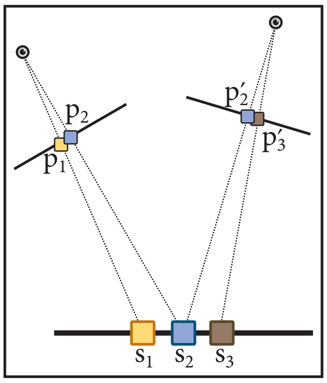
2.3.4 *Voxel sampling*

A voxel's observation is used to estimate its reflectance model and therefore evaluate its photo-consistency. Unfortunately, the use of volumes immediately compromises the assumptions underpinning the photo-consistency constraint because a voxel encompasses an infinite number of scene-points. Any number of the points within the voxel could be free space or surface points with different reflection properties. A voxel, by definition, must represent this space by a *single* material property. The problem, then, is to sample a voxel's projection in the set of reference images to define a single material property. Many approaches treat voxels as a geometric entity in scene space to simplify the sampling process. In the most straightforward of these approaches, only the projection of the voxel's centroid is considered. While straightforward to implement, the reliance on a single point means consistency evaluation can be affected by small errors in calibration and differences in the assumed reflectance-model.

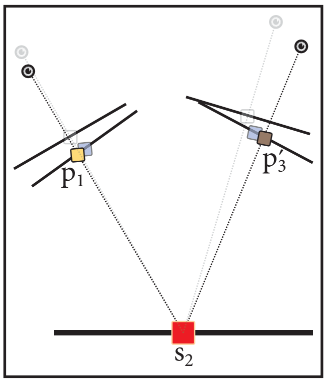
Rather than only using the projection of a single point, some approaches (Stevens, for example [61]) sample the volume using quadrilaterals aligned with each reference view. Other approaches include imposing a box (or nearest-neighbour) filter over the volume space. Broadhurst, for example, adopts a cube voxel model to sample the image, but only uses the projection of the front face [8]. This approach rasterises the front-face polygon and assigns a per-pixel weight indicating the extent to which the pixel samples the voxel. Steinback similarly uses a cube model, but measures photo-consistency over the cube's projection rather than just its front-face [59, 60]. Instead of sampling a voxel's projection, Horung integrates photo-consistency over a voxel's domain in 3D-space by computing the normalised cross correlation of a set of samples distributed within the voxel [25]. To minimise errors introduced by sampling free-space of voxels that only contain a subset of the surface, each voxel sub-sample is weighted by the angle between the voxel's normal and the reference view's optical axis.

These sampling approaches assume the camera is accurately calibrated so that points on the reference surface have are correctly aligned in the reference images. Kutulakos relaxes this assumption by sampling a single pixel from a disc around the projection of the voxel's centroid [28]. 'Shuffle transformations' are used to describe an arbitrary pixel rearrangement, such as the random relocation of a pixel within a disc of radius r.

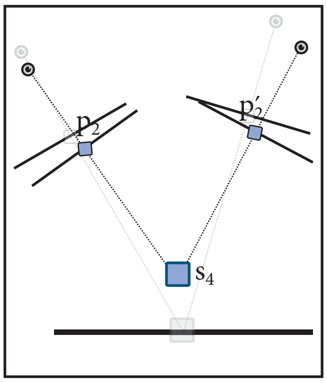
The disc radius controls the level of dispersion tolerated by the photo-consistency criterion. While $r = 0$ is the strict photo-consistency criterion used in most Space Carving algorithms, a surface consistent with a shuffle transform of r' implies the surface is also consistent at radius r if $r' > r$. These r-shuffles allow for some tolerance of photo-consistency error by recovering a one-parameter family of volumes that have an increasingly tighter super-set of the true surface. This approach is suitable for coarse-to-fine refinement of the voxel space with increasing bounds on the disc radius.



(a) The reference scene and correctly calibrated cameras.



(b) Inconsistency from calibration errors



(c) Extraneous voxels from calibration errors.

Figure 2.12: Camera calibration errors mean surface voxels can appear inconsistent while voxels not on the surface appear consistent.

2.3.5 *Consistency metrics*

Space carving algorithms rely on comparing the hypothesised projection of surface voxels. Small calibration errors can mean a point on the surface actually projects to observations of different surface points in the reference images. This is not a critical problem if the misalignment is small and the surface is relatively free of texture; but the misalignment can lead to erroneous inconsistency in the case of highly textured regions. Figure 2.12 illustrates the shape distortion arising from calibration errors. Attempts to reconstruct the surface in Figure 2.12(a) with calibration errors are illustrated in Figures 2.12(b) and 2.12(c). Voxel $\boldsymbol{s}_2$ in Figure 2.12(b) is on the reference surface, but the calibration errors indicate its projection is $\boldsymbol{p}_1$ and $\boldsymbol{p}'_3$ and therefore $\boldsymbol{s}_2$ is considered inconsistent. Similarly, the calibration error has shifted the projection of $\boldsymbol{s}_2$ from the surface, making voxel $\boldsymbol{s}_4$ appear consistent.

A voxel's material property must account for filtering errors over the camera's sensor and problems with representing the summary of a volume with a single reflectance model. A voxel can be represented by a 3D box[2] if a nearest neighbour filter is used to sample the volumetric space. The projection of this box defines an area in the reference image. One approach to determine the observed radiance of a diffuse voxel is to compute the average colour from its footprint. There are two problems with

[2]The 'box' model is *only* an artifact of the reconstruction filter [57].

this approach. Firstly, a voxel must represent a summary of its volume, but it may simultaneously contain textured surface points and free space. Secondly, the sensor design used in digital cameras means that the observed radiance contains filtering artifacts and noise, even for correctly hypothesised points.

Voxel colouring and Space Carving traditionally use the variance of the sampling process as a measure of the voxel's photo consistency. The variance is straightforward to compute, but it is sensitive to specular or strongly textured surfaces. Most implementations therefore require a consistency threshold to tolerate small variations in pixel intensity due to noise and slightly specular surface. This threshold must be chosen before the model is evolved, yet it is critical to the reconstruction's success. A threshold that is too aggressive can inadvertently carve too much of the surface and therefore break its conservative estimate of occlusion. This can lead to a cascading carving chain that destroys the reconstruction. A lenient threshold, on the other hand, tends to stops carving too early which leads to 'inflated' reconstructions.

Removing voxels is an irreversible process that compounds the problem of choosing a threshold. Since Space Carving uses the surface hypothesis to maintain a conservative model of visibility, erroneously removing a voxel can destroy significant sections of the reconstruction. The problem is one of undermining the occlusion model that Space Carving uses to determine which reference images can vote for a voxel's consistency. If the occlusion model is broken, then the reference cameras would be allowed to see *inside* the reference surface. These points would appear photo-inconsistent because they are not visible in the reference images. Removing these voxels would further destroy the occlusion model, causing a chain of voxel removals that would ultimately erode the entire model away.

Stevens proposed a photo-metric that does not use rely on a consistency threshold [61]. A colour histogram in RGB space is generated for each observation and used for consistency analysis. The voxel is considered inconsistent if the intersection of all observation histograms is empty. This approach is well suited for dealing with textured regions, particularly for large voxel foot-prints where the distribution of colours is likely to be multi-modal.

2.3.6 Carving as an optimisation problem

Slaubaugh treats carving as a Monte Carlo optimisation process which can remove *and add* voxels to maximise photo-consistency [54]. Given a photo-hull carved by their Generalised Voxel Colouring algorithm [13], the surface's photo-consistency is optimised by tentatively adding or removing random surface voxels. Tentative modifications are accepted according to a simulated annealing schedule. This optimisation process has two key advantages over Space Carving. Firstly, *speculative* changes can be made to the surface; and secondly, it treats photo-consistency as a metric over the entire surface rather than making a chain of independent decisions.

Rather than making random changes to a reconstruction, Isidoro refines a mesh representation of the visual hull by searching for consistent points along epipolar lines sampled from a distribution of photo-inconsistent points [26]. The surface's photo-consistency is evaluated by back-projecting the reference images onto the mesh and measuring the disparity from their synthetic image in the reference views. This displacement represents an inconsistency probability density over the image, where high inconsistency in the image indicates that the associated visible vertex is incorrectly positioned. Samples are randomly drawn from this image density and back-projected to yield epipolar lines in the other images (or 'strands' in 3D). These lines are sampled to find the point with minimum re-projection error.

While this approach models occlusion when determining photo-consistency, the views that are used are those in which a strand's starting point is visible. A distortion vector is created for each vertex coincident with the strand to push the mesh toward the point of maximum consistency. The vertex (and its neighbours, to some extent, according to a smoothness term) are distorted by this set of free-form distortion vectors. These vectors are used to update the mesh and the process is repeated until the surface is photo-consistent.

Isidoro extends this work by incorporating silhouette constraints and sampling inconsistency over its texture rather than its image [27]. A surface texture is created by unfolding the visual hull onto a plane. This texture image is estimated by back-projecting all reference images onto the surface and mapping the result into texture space. This process is repeated for all reference images to compute the distribution of inconsistent points over the surface. Points are sampled from this texture to yield strands joining the corresponding point in scene-space to each of the camera's optical centres. Like their image-based approach, these strands are sampled to find the point of maximum photo inconsistency, which are then used to find a new set of distortion vectors. Silhouette constraints are enforced by using the set of photo-consistent points on the visual hull to contribute deformation vectors for adjacent vertices. These *'frontier* *points*' are guaranteed to be on all photo-consistent hulls, and ensure that the mesh is not 'over-pushed' by finding a photo-consistent solution which takes short but inconsistent paths to minimise consistency error.

Tran poses volumetric reconstruction as a problem of finding the minimum cut of a weighted graph [65]. The graph is defined by assigning one node to each voxel in the visual hull. Silhouette constraints are enforced by using 'blocking curves' that connect frontier points to an *a priori* depth below the hull's surface. Weights associated with these edges are designed to heavily penalise a solution that includes them in the reconstruction. These weights allow the graph cut to explore concavities in the surface rather than making short but inconsistent paths which skip the concavity.

Vogiatzis also uses graph cut framework, but does not use silhouette constraints [68]. The error term for each node is given by the photo-consistency of a patch centered at the voxel; the patch's normal is defined by the closest point on the visual hull to the patch. The voxel's photo-consistency is given by the patch's average cross-correlation score between all pairs of cameras where the voxel is visible. Similar to Iisidoro's algorithm, a patch's normal and its visibility is determined with respect to the visual hull; the visibility of interior voxels is estimated by projecting the voxel onto the surface and using the surface point's visibility. Because this treatment of visibility can lead to unreliable NCC scores for interior voxels, carving is limited to within a certain depth of the visual hull. Projective warping is likely to make the NCC scores unreliable for wide baselines, so only camera pairs whose optical axes are within 45° of each other and within 60° of the surface normal are used to compute the average NCC score.

The optimal reconstruction is found by minimising the consistency integral over the surface. A disadvantage of this approach is its bias toward small, but highly photo-inconsistent surfaces rather than more correct surfaces that have a larger aggregate error due to integrating sensor noise over a larger surface. While Isidoro addressed this problem with silhouette constraints, Vogiatzis adds a 'ballooning' term to favour larger solutions that fill the visual hull. The combination of the ballooning term and the NCC score is used to weight edges in a graph. Graph cuts are used to find the solution that minimises the surface's consistency score.

2.3.7 *Probabilistic carving*

Space carving evolves a shape hypothesis by considering a single voxel at a time to determine its consistency. Voxels that fail the consistency test are irrevocably removed from the model. Rather than making a binary, binding, decision, a number of approaches have considered evolving an occupancy probability for each voxel. The advantage in this approach is that it directly models uncertainty and is therefore able to correctly handle the problem of committing to a carving decision.

Debonet uses responsibility weighted voxels (or '*roxels*') to model each voxel's contribution to the surface [6]. Voxels are not simply opaque cubes, but a mechanism for sampling 3D-space; only a subset of the voxel may actually contain the surface. This fractional contribution is modelled using α−weights per voxel in the same manner as Porter's [44] pixel-based α-contribution. The photo-consistency test accounts for the voxel's partial transparency by tracing rays through a given pixel and accumulating the transmission of each voxel that it intersects. The apparent colour of a voxel is given by α-proportion of the voxel's colour and weighting all voxels further along the ray by $(1-\alpha)$: that is, the apparent colour of a voxel's projection is the proportion it transmits, and the remainder is from the reflectance of voxels behind it.

The volume's colour and opacity are estimated via an iterative, multi-stage process that alternates between estimating each voxel's colour and then its responsibility. The voxel's view-specific responsibility is initialised such that each voxel contributes equally to its image. The voxel's colour is estimated as a weighted average of its observed colour, modulated by the voxel's estimated responsibility toward its image. The result is compared with each reference image to yield a per-view photo-consistency metric; the inverse of this metric is described as the voxel's 'agreement' with each reference image.

Each voxel's responsibility is estimated by normalising agreements along each viewing ray. Photo-consistent voxels have high agreements, indicating that the voxel is highly responsible for the image under its projection. The per-view opacity is generated for each voxel by inverting the responsibility equation and combining it into a view-*in*dependent opacity metric. Each voxel's responsibility is re-evaluated from its updated opacity estimate and the entire procedure is repeated until convergence.

Broadhurst also recovers a per-voxel probability using a Bayesian framework [9]. The likelihood that a voxel is on the reference surface is given by its photo-consistency metric conditioned on its visibility probability. The visibility probability is given as the joint probability that all potentially occluding voxels are free space. To make the problem tractable, each voxel in the scene is visited according to the ordinality constraint. The probability of all potentially occluding voxels is therefore estimated before considering the visibility of a given voxel. The event that the voxel is on the surface is given by marginalising over the photo-consistency of all surfaces that include the voxel. A voxel that is not on the surface is modelled by a set of independent colour models, one for each reference view. Bayes' learning rule is used to estimate which of these two events is more likely.

Agrawal proposed a probability framework that does not rely on the ordinality constraint, but it assumes that an a priori bound on the surface's visibility is known [1]. Their algorithm iteratively updates a per-voxel probability that the voxel is on the reference surface by collecting evidence along rays through the voxel and each reference view. The back-projection of an image point in the reference image r_i describes a ray in scene-space. A surface point $\boldsymbol{v}$ on this ray must obey three constraints:

- all voxels on the ray closer than $\boldsymbol{v}$ to the reference camera must be free-space if $\boldsymbol{v}$ is visible in r_i;
- if $\boldsymbol{v}$ is not visible in r_i, then there must be some surface voxel $\boldsymbol{v}'$ closer than $\boldsymbol{v}$ to occlude $\boldsymbol{v}$; and
- no voxel $\boldsymbol{v}''$ on the ray further than $\boldsymbol{v}$ can be visible in r_i.

These constraints are used as evidence to support the probability that a point is on the surface.

The probability that a voxel is on the surface is given by comparing its projection in two views and relying on the assumption that each surface voxel is visible in an a priori known number of views. The probability that a voxel is visible in two views is a function of its correlation score, where the probability is high for images that are very similar but decreases monotonically as the dissimilarity increases. If a voxel is visible in a subset of reference views, then the voxel must be visible in all pairs of views from this subset. Accordingly, the probability that a voxel is visible in a given subset of views is described by the *smallest* correlation score from all pairs of views of the subset. Finally, the probability that a surface voxel is visible in a *specific* image i is the maximum probability that it is visible in a subset of n views containing i, where n is from the constraint that every surface voxel is visible in at least n reference images.

The view-specific probability is used to identify candidate surface points. Evidence is collected for a voxel by scanning the rays spanning it and each reference view. The two voxels along these rays with the least probability of obeying these constraints are used as evidence to update the voxel's surface probability in a Bayesian framework. The process iteratively updates the voxel's surface probability until the system converges.

Smelyansky uses a Bayesian framework based on a photo-consistency likelihood to register reference images to a surface model [56]. Using a height-field surface model, Smelyansky infers the mesh height and albedo at each vertex, and makes small improvements to the camera parameters. Like the probabilistic carving algorithms, the surface's ability to recreate the reference views is used as a likelihood metric, but the metric considers the photo-consistency simultaneously over the entire surface.

The implementation of the global photo-consistency constraint is made tractable by ignoring global illumination changes such as occlusion and shadows. This allows synthetic images to be differentiated with respect to changes in the height-field, albedo and camera parameters. Render engines typically sample only a single triangle per pixel, yielding images that are not differentiable. Smelyansky proposed a custom render engine that accounts for the partial flux of *all* triangles that project to a given pixel. This allows the use of a gradient-based optimisation processes to minimise the disparity between the synthetically textured surface and the reference images.

2.3.8 Photo-consistency on graphics hardware

Determining photo-consistency ultimately compares the reference images with synthetic images of the hypothesised surface to make decisions about evolving the hypothesis. Generating synthetic images of models is very efficiently addressed by graphics hardware. Prock and Dyer use graphics hardware to back-project each reference image onto a plane swept through space, effectively registering each reference image in scene-space [46, 18]. While texture mapping hardware is used to back-project the reference images, it is not clear how this approach accounts for occlusion.

Sainz similarly projects the reference images onto the sweep plane, but occlusion is modelled by projecting the evolving surface onto the registration plane [48]. Colour-codes are used to identify voxels in the sweep plane (which is not unlike a voxel-space representation of Culbertson's item-buffer [13]) and Blinn's shadow projection [4] is used to remove identifiers of occluded voxels. This creates an identifier mask that correlates visible voxels with the projection of the reference images onto the sweep plane. This mask is then read back and used to identify visible voxels for consistency testing.

Yang uses graphics hardware to estimate depth maps while generating novel views [71, 72]. The variance of hypothesised surface points is approximated by measuring the average squared difference between each reference image and the reference image closest to the virtual camera. The correlation is measured by sweeping a plane parallel to the virtual camera's imaging plane through space. Each pair of reference views is back-projected onto the plane and compared in a fragment shader. The average colour corresponding to the lowest disparity score for each pixel is retained.

Yang does not model occlusion to filter erroneous correspondences and therefore relies on each scene point being visible in each reference view. The sweep plane only compares the colour disparity of a single pixel across all views. While this is acceptable for a large number of views to minimise mis-prediction, the use of SSD over a window, rather than a single point becomes more important as the number of views decreases. Yang extend their single pixel disparity approach by considering the sum of squared differences across a range of window sizes to increase matching accuracy [70].

Li uses graphics hardware to generate novel views by using a single sweep along the virtual camera's optical axis [33]. The graphics fragment processor is used to compute photo-consistency, and therefore limits the number of reference cameras to the number of texture units in the graphics pipe. Li notes that aliasing artifacts on the silhouette edge are likely to mis-predict the point's photo-consistency. To correct for this, a *'slope-map'* is introduced such that cameras with an oblique view of the surface have less influence on determining a point's consistency.

2.4 Summary

Feature-based approaches can only reconstruct parts of the scene that are recognised by a feature-detector. Such systems require a small base-line to assist in finding correspondences. Because these approaches work only with features, the recovered models are unsuitable for a number of applications that require 'photo-realistic' reconstructions. An alternative approach is to work directly in scene-space. Some of these approaches use models of the scene to assist with an underlying feature-based reconstruction, but an increasing number of approaches do not use features at all. Instead, a model is recovered by evolving a hypothesis until it recreates the reference images. These reconstructions are more *'complete'* than feature-based reconstructions and are more suitable for a variety of applications, including generating novel views of the scene and recovering dense depth-maps for integration with computer generated imagery.

While recovering generative models is a compelling idea, a number of approximations and assumptions are made in practice. These include using a surface model that either restricts the type of scenes to those containing pre-defined elements; limiting camera placement; or recovering scenes bounded by an initial scene envelope. The process of recovering the model has also shifted to probability-based and global optimisation approaches to try and overcome some of the limitations inherent in considering only one voxel at a time.

Parameterised Photo-Consistent Geometry

Chapter III

Evolving a reconstruction in scene-space confers a number of advantages over image-based approaches. In general, scene-space approaches evolve a hypothesis to find a 3D surface that recreate the reference images. Different approaches to this problem make different assumptions about the type of scene (and what constraints on the surface, if any); how shape uncertainty is modelled; and how its consistency is evaluated. Each scene-based approach has its own advantages and disadvantages which we believe can be combined into a hybrid system.

Our reconstruction process combines Space Carving's photo-consistency constraint with a geometric model similar to Facçade [15] with the aim of recovering a photo-consistent polyhedral mesh without requiring the user to manually identify correspondences between the 3D structure and the reference images. Because a scene-graph is used to specify the set of solutions, our system is capable of recovering spatially coherent photo-consistent geometry even if the geometry is partially occluded. Our approach is further characterised by its ability to run on graphics hardware. Graphics hardware is very adept at rendering triangle meshes, which is exploited to make tractable the problem of estimating the colour assignment of potentially millions of surface points for a given hypothesised surface.

3.1 Overview

Our reconstruction approach uses a scene-graph to constrain the reconstruction process. Like Façade, the scene-graph is a collection of *'building blocks'* related by parameterised transformations. The advantage of parameterised geometry is that large surfaces can be represented with very few parameters. Not only is the optimisation process made simpler through parameter reduction, but the scene-graph also allows prior knowledge to be readily incorporated into the reconstruction process through both the graph's hierarchy and using prior distributions for transformation parameters.

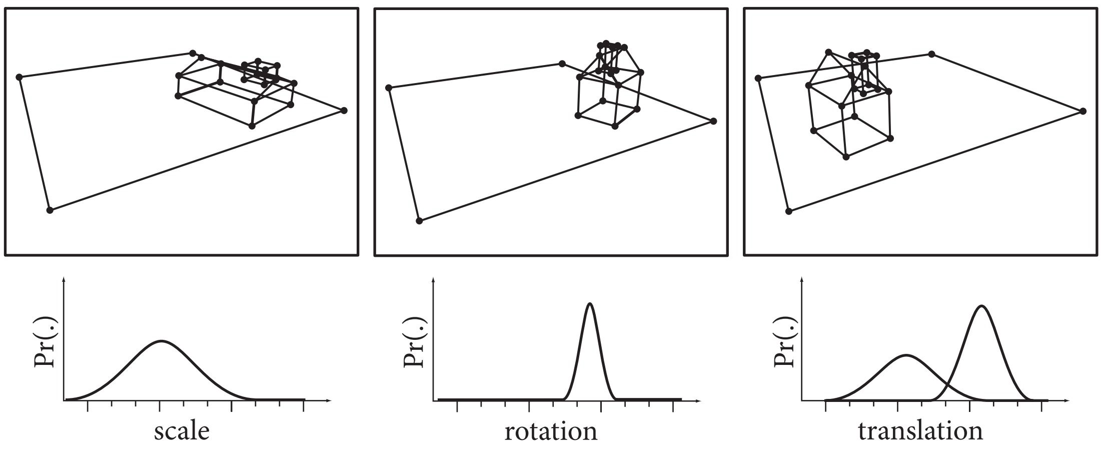
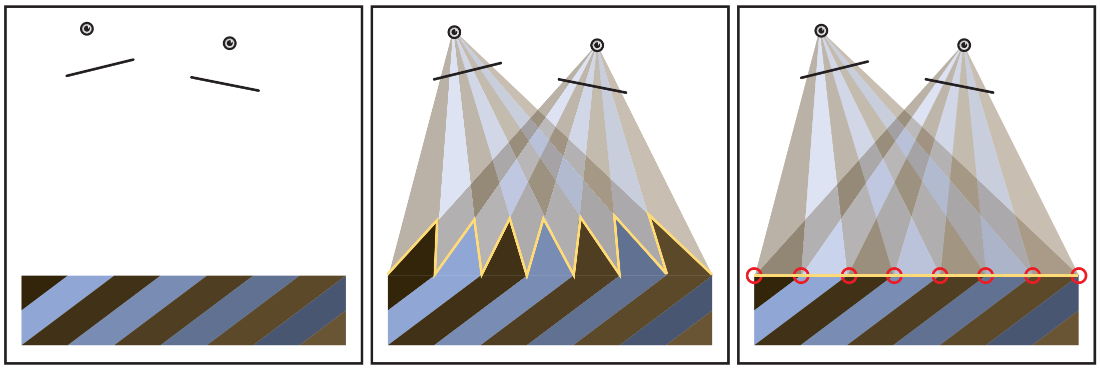


Figure 3.1: The surface can be radically transformed by changing only a few parameters, each with their own prior distributions.

The reconstruction does not need to be simply the largest photo-consistent surface because the space of hypothesised surfaces is constrained by the scene-graph. This is particularly relevant for surfaces that have regions of detail which could be used to guide the reconstruction in relatively untextured patches. The maximal photo-consistent shape in Figure 3.2(b), for example, has peaks of voxels within regions free of texture. Forcing the solution space to contain a single plane ensures the reconstruction intersects these textured points while the surface remains photo-consistent with the reference images. Including geometric constraints therefore partly overcomes the problem of carving's inherent ambiguity by fitting spatially coherent surfaces that the user *anticipates*, rather than finding only the largest shape consistent with the images.

The set of parameters characterises the hypothesised scene; depending on the scene-graph hierarchy, changing even a subset of the parameters can fundamentally change the hypothesised surface. The effect of modifying the scene structure is illustrated in Figure 3.1 where the same scene of a house is modified by scale, rotation, and translation parameters. Constraining the solution space to a user-supplied scene-graph turns the reconstruction problem into one of estimating the parameter values that transform the scene-graph into a surface that best recreates the reference images.

We use a Bayesian learning process [53] to infer the most probable parameter vector from the posterior given the reference images and user-supplied scene-graph. The posterior is characterised by the product of the priors and likelihood of generating the images given a hypothesised parameter vector. The likelihood of a hypothesised



(a) Reference surface **(b)** Maximal photo-consistent shape **(c)** Constrained shape

Figure 3.2: Photo-consistency with geometry constraints can be used to reduce surface ambiguity rather than finding only the largest consistent shape.

parameter vector is measured by generating synthetic images of the hypothesised surface with the reference camera parameters and comparing their similarity to the corresponding reference images.

The posterior is highly non-linear because occlusion dramatically changes a surface point's perceived photo-consistency. Because of this non-linearity, we use a Monte Carlo Markov Chain [38] to draw a large number of hypothesised surfaces from the posterior with the aim of finding the maximum a posteri (MAP) parameter vector. An advantage of this approach is that the parameter space can be explored without committing to bad decisions. A further advantage is that the camera parameters can also be a part of the optimisation process. This is not tractable in Space Carving because changing the camera parameters has a fundamental effect on how 3D points project to hypothesised point correspondences.

3.2 Fitting photo-consistent geometry

We treat scene-reconstruction as a problem of assigning a probability to hypothesised scene-graph parameter vectors. Suppose $\mathcal{S}$ is a user-supplied scene-graph that connects primitives by parameterised transformations whose values are taken from the parameter vector $\Psi = [\rho_1, \cdots, \rho_n]$. We are interested in assigning a probability to the vector ψ, denoted $\Pr(\Psi = \psi \,|\, \mathcal{I}, \mathcal{S})$, to reflect the degree of belief that the scene-graph

$\mathcal{S}$ represents the scene imaged in the set of reference images, $\mathcal{I} = \{\mathsf{r}_i\}$ when $\mathcal{S}$ is parameterised by ψ.

Assigning a probability for each hypothesised parameter vector $\Psi = \psi$ for each $\psi \in \mathbb{R}^n$, yields the probability distribution, $\Pr(\Psi \mid \mathcal{I}, \mathcal{S})$. This distribution is derived by rearrangement of the joint probability $\Pr(\Psi, \mathcal{I} \mid \mathcal{S})$. Using the product rule, the probability $\Pr(\Psi, \mathcal{I} \mid \mathcal{S})$ can be written as the conditional probability,

$$\Pr(\Psi, \mathcal{I} \mid \mathcal{S}) = \Pr(\Psi \mid \mathcal{I}, \mathcal{S}) \Pr(\Psi) \tag{3.1}$$

and, similarly,

$$\Pr(\Psi, \mathcal{I} \mid \mathcal{S}) = \Pr(\mathcal{I} \mid \Psi, \mathcal{S}) \Pr(\mathcal{I}). \tag{3.2}$$

Because these conditional probabilities are equivalent, they can be re-arranged to give

$$\Pr(\Psi \mid \mathcal{I}, \mathcal{S}) = \frac{\Pr(\mathcal{I} \mid \Psi, \mathcal{S}) \Pr(\Psi)}{\Pr(\mathcal{I})}. \tag{3.3}$$

$$\propto \mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S}) \Pr(\Psi), \tag{3.4}$$

which is the Bayesian posterior that encapsulates all information about the scene parameters given the reference images and scene-graph. The posterior (3.3) combines three distributions:

- $\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S})$, the likelihood of generating the reference images given a hypothesised parameter vector;
- $\Pr(\Psi)$, the prior distributions on the parameter space; and
- $\Pr(\mathcal{I})$, the marginal probability of observing a given set of reference images.

The prior distributions, $\Pr(\Psi)$, denote predictions on the scene-parameter values and are defined by the user as part of the process of describing the scene-graph. If the scene-parameters are assumed to be independent, the prior probability distribution is given by

$$\Pr(\Psi) = \Pr(\rho_1, \cdots, \rho_n) = \prod_i \Upsilon_i \tag{3.5}$$

where Υ_i is the prior for each scene-graph parameter. In practice, each scene-parameter prior is given by

$$\Upsilon_i = \mathcal{U}(\hat{\rho}_i - \varepsilon_i, \hat{\rho}_i + \varepsilon_i). \tag{3.6}$$

where $\hat{\rho}_i$ is an estimate of the scene-parameter, ε_i signifies uncertainty and $\mathcal{U}(a, b)$ denotes a uniform distribution in the domain $[a, b]$.

The likelihood of recreating the reference images $\mathcal{I}$ is given by the distribution $\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S})$. Given a hypothesised parameter vector, Ψ, the likelihood function generates a hypothesised surface $\mathcal{S}(\Psi)$ by transforming the primitives in $\mathcal{S}$ by the transformations parameterised by Ψ. The likelihood $\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S})$ is given by the similarity between the reference images, $\mathcal{I}$, and synthetic images of $\mathcal{S}(\Psi)$ imaged using the reference cameras. The specification of the image likelihood $\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S})$ is described in the remainder of this Chapter.

3.2.1 *The surface model*

In determining the likelihood of observing $\mathcal{I}$, the reference images are compared with synthetic images of the hypothesised surface imaged by the reference cameras. The surface model must therefore contain sufficient information about the hypothesised surface so that synthetic images of $\mathcal{S}(\Psi)$ are comparable to the reference images. A sparse point-cloud, for example, is an unsuitable surface model because synthetic images of the 3D point-cloud is a set of 2D points, and cannot be meaningfully compared with the reference images.

Triangle meshes are a ubiquitous form of surface representation [7] which can be used to generate photo-realistic images, and are very efficiently rendered on graphics hardware [3, 40]. A mesh is defined by a collection of vertices in $\mathbb{R}^3$ and adjacency information to connect these vertices into a set of triangular faces. The mesh describes a linear, piecewise surface approximation, but requires a large number of vertices, and therefore scene-parameters, unless an underlying structure is imposed on vertex position and adjacency.

Real scenes are *not* a collection of random vertices. Photographs of buildings, for example, exhibit a high degree of a patterned, regular, structure. The number of surface parameters can be significantly reduced in these cases by exploiting relationships between surface entities. A box, for example, requires six vertices in three-dimensions and therefore eighteen degrees of freedom unless the relationship between vertices is constrained. However, only nine parameters are required to represent the box's translation, scale, and rotation if right-angled constraints are imposed, and can be further reduced to seven parameters if a cube constraint is imposed on the scale parameters.

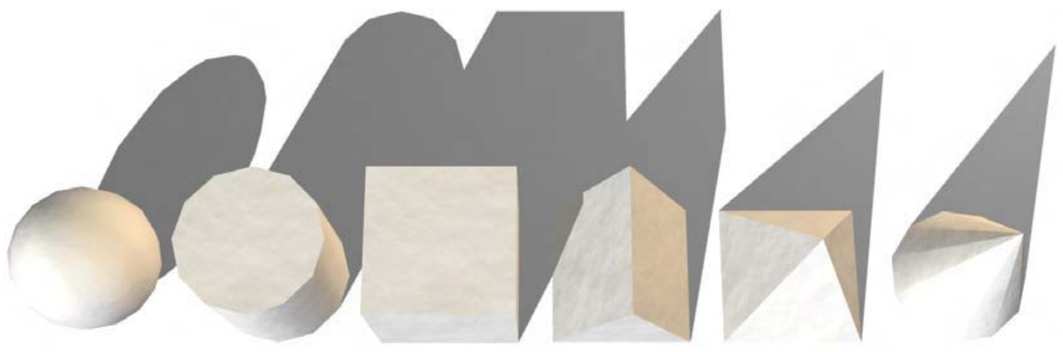


Figure 3.3: Examples of scene-graph primitives.

Geometric constraints between planar facets are represented in our system by defining a set of building blocks. These blocks, or *primitives*, are pre-defined rigid geometric entities. Examples of these primitives include spheres, cylinders, cubes, prisms, tetrahedrons and cylinders (see Figure 3.3). A primitive encodes constraints between its faces. The cube, for example, is defined such that opposite faces are parallel, adjacent faces are orthogonal, and all sides are of equal length. The primitives are defined in a 'normalised' coordinate system to facilitate stacking and alignment.

3.2.2 *Hierarchical transformations*

Geometric relationships are described by connecting primitives together in a directed acyclic graph. This *scene-graph* is a set of connection nodes, transformation links, and geometry leaves. Connection nodes are used to assemble a collection of directed links, which describe transformations, and therefore relationships, between connection nodes. Transform links hold symbolic references to parameters stored in the scene-graph's parameter vector; multiple transforms can reference the same parameter which reduces the required number of parameters. These transformations are described by an ordered list of parameterised 4×4 matrices $\boldsymbol{T}_\star$, $\boldsymbol{R}_\star$, and $\boldsymbol{S}_\star$ to represent translation, rotation, and scale, where $\star$ denotes which axes are parameterised. These transformations are accumulated as the graph is traversed.

Figure 3.4 is an example of a scene-graph illustrating how transformations are propagated through a hierarchical model. The house model is built-up from the ground plane by stacking primitives so that related transformations are inherited. The house walls, for example, are transformed by the link $1 \rightarrow 2$ from the scene co-ordinate frame.

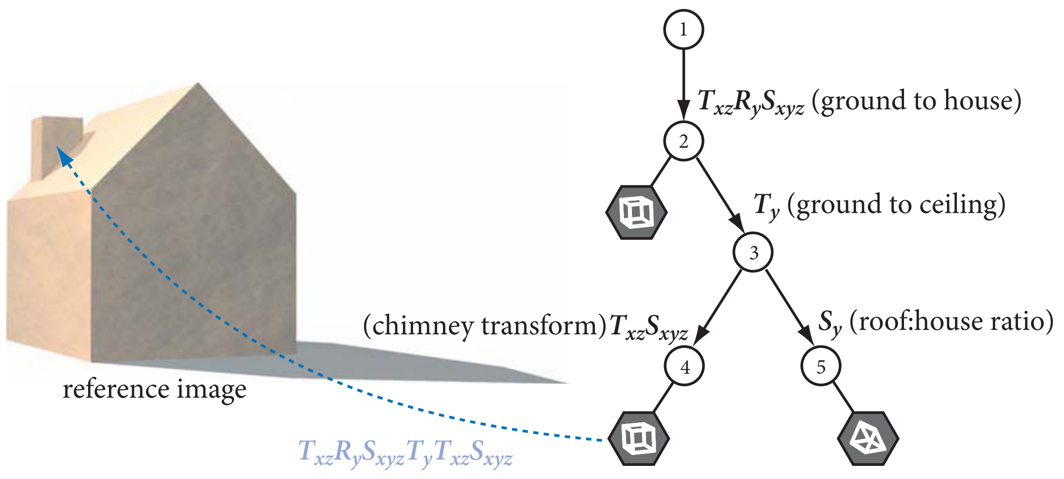

1
$T_{xz}R_yS_{xyz}$ (ground to house)
2
T_y (ground to ceiling)
3
(chimney transform) $T_{xz}S_{xyz}$
S_y (roof:house ratio)
4
5
reference image
$T_{xz}R_yS_{xyz}T_yT_{xz}S_{xyz}$

Figure 3.4: Scene graph construction for a house model illustrating how transformations are propagated along edges. Geometry leaves are denoted by hexagons; connection nodes are enumerated. Nodes are related by transformations along edges.

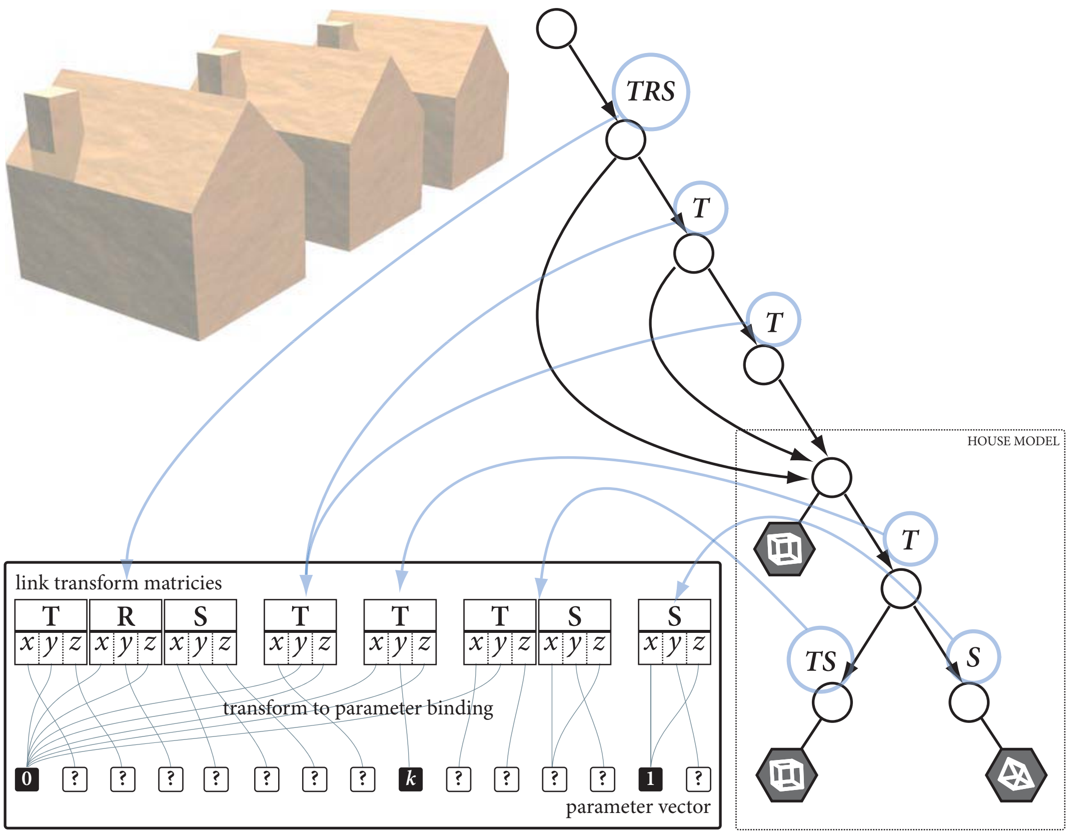


Figure 3.5: Binding scene-graph parameters to transformations and the transformations to links between scene-graph nodes.

Node 1's co-ordinate frame is undefined, but is assumed to be the identity matrix. In practice, this transformation defines the scene co-ordinate frame relative to the camera rig. The link 1 → 2 describes the five unknown parameters to transform the house relative to the co-ordinate frame imposed by the ground-plane. The rotation about the $x-$ and $z-$ axes can be omitted because the walls are orthogonal to the ground-plane; the translation along the vertical axis can also be omitted because the house does not float above the ground. This leaves three parameters to position and orientate the house relative to the scene-frame.

Transformations are propagated as the graph is traversed. The cube attached to node 2 (representing the four walls of the house), for example, is rendered with the transform

$$\boldsymbol{H}_H\boldsymbol{H}_K\boldsymbol{H}_C = \boldsymbol{T}_{xy}\boldsymbol{R}_y\boldsymbol{S}_{xyz}\boldsymbol{T}_y\boldsymbol{T}_{xz}\boldsymbol{S}_{xyz} \tag{3.7}$$

where

- $\boldsymbol{H}_H = \boldsymbol{T}_{xy}\boldsymbol{R}_y\boldsymbol{S}_{xyz}$, from link 1 → 2, orientates and defines the scale of the house;
- $\boldsymbol{H}_K = \boldsymbol{T}_y$, from link 2 → 3, is a constant transform in the cube's local co-ordinate system which aligns the chimney with the top of the ceiling; and
- $\boldsymbol{H}_C = \boldsymbol{T}_{xz}\boldsymbol{S}_{xyz}$, from link 2 → 4, is the transformation describing the chimney's position and scale relative to the house.

Some links are not parameterised because they are used to align objects. Link 2 → 3, for example, is a constant translation to shift the origin to the top of the cube. The translation is constant since it is pre-multiplied by the house scale transform in link 1 → 2. Shifting the origin in this manner means the chimney and roof primitives can be arranged independently of the house transform; the roof does not require an additional height parameter from the ground plane because this transformation is inherited by the link 1 → 2.

There are three classes of surface constraints imposed by the primitive geometry, transformation parametrisation, and scene-graph structure. Examples of these constraints are illustrated in Figure 3.5:

- *geometric constraints*—wall orthogonality is described by the box geometry;
- *similarity constraints*—the distance between the three houses are equivalent because the transformation separating them refers to the same parameter; and
- *template constraints*—the three houses share a common sub-graph.

The scene-graph is a mechanism to parameterise a triangular mesh. In order to recreate the reference images, the scene model requires a description of the geometry and surface reflectance. If only a single reflectance model could be assigned to each triangle, then an unmanageable number of triangles would be required to model even a slightly textured surface. The solution is to use *texture mapping* to associate a homography between a texture image and each surface triangle. This homography is used to map a detailed, point-sampled reflectance model that is mapped onto the hypothesised surface when the surface is rendered by a virtual camera.

As the texture is part of the surface model that we seek to estimate from the reference images, each point in the texture image *could* be a random variable from Ψ to denote the surface's estimated reflectance. The relationship between the hypothesised surface and its synthetic projection in each reference image, however, can be used as photo-metric constraints, turning the reconstruction problem into one of estimating the most likely texture given the hypothesised geometry.

3.2.3 *The rendering equation*

The surface parameters are estimated by deforming the scene-graph by a hypothesised parameter vector Ψ and computing the likelihood $\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S})$. The likelihood measures the similarity between the reference images $\mathcal{I}$ and the set of synthetic images generated by imaging $\mathcal{S}(\Psi)$ with the reference camera parameters. The set of synthetic images s_i is generated by back-projecting each point $\underline{\boldsymbol{p}} \in \mathsf{s}_i$ into scene-space, and finding the closest point to the camera's optical centre that intersects the surface. The closest point is then mapped into the surface's texture image to define the colour that is mapped back to $\underline{\boldsymbol{p}}$. We use here the notation $\underline{\boldsymbol{p}}$ to denote Euclidean points and $\boldsymbol{p}$ the corresponding point in homogeneous co-ordinates, such that

$$\underline{\boldsymbol{p}} = [\frac{x}{w}, \frac{y}{w}, \frac{z}{w}]^\top \equiv \boldsymbol{p} = [x, y, z, w]^\top. \tag{3.8}$$

Images are synthesised by a pin-hole camera model described by an imaging plane and optical centre. Suppose Π_i is the image plane and $\boldsymbol{o}_i$ is the optical centre of the camera hypothesised to generate r_i, and let $\boldsymbol{E}_i$ be the homography to map world points to normalised camera co-ordinates (that is, $\boldsymbol{E}_i\boldsymbol{o}_i$ is the origin and $\boldsymbol{E}_i^{-\top}\Pi_i$ is the plane $z = -1$ in the camera co-ordinate frame). Let $\{\mathsf{t}_j = \{\boldsymbol{v}_j^1, \boldsymbol{v}_j^2, \boldsymbol{v}_j^3\}\}$ be the set of hypothesised scene triangles in the i^{th} camera's co-ordinate frame with corresponding texture images, $\mathcal{C} = \{\mathrm{t_j}\}$. Given this normalised co-ordinate space, the point $\underline{\boldsymbol{p}} \in \boldsymbol{E}_i^{-\top}\Pi_i$ forms the ray

$$\lambda \boldsymbol{p} = \begin{bmatrix} \underline{\boldsymbol{p}} \\ \lambda \end{bmatrix} \tag{3.9}$$

through $\underline{\boldsymbol{p}}$ on the imaging plane and the camera's optical centre. The ray (3.9) intersects the plane coincident with triangle t_j when

$$\boldsymbol{p}^\top \begin{bmatrix} \boldsymbol{n}_j \\ \boldsymbol{n}_j^\top \boldsymbol{E}_i \boldsymbol{v}_j^1 \end{bmatrix} = 0 \tag{3.10}$$

where

$$\boldsymbol{n}_j = (\boldsymbol{E}_i \boldsymbol{v}_j^3 - \boldsymbol{E}_i \boldsymbol{v}_j^1) \times (\boldsymbol{E}_i \boldsymbol{v}_j^2 - \boldsymbol{E}_i \boldsymbol{v}_j^1) \tag{3.11}$$

is the plane's normal. The point $\boldsymbol{p}_j = \lambda_j \underline{\boldsymbol{p}}$, the solution to (3.10) for the j^{th} plane, is within the triangle t_j if

$$\|\boldsymbol{H}_j \boldsymbol{E}_i^{-1} \boldsymbol{p}_j\|_1 \leq 1, \tag{3.12}$$

where the homography

$$\boldsymbol{H}_j = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3]^{-1} \tag{3.13}$$

maps the triangle to its texture space. In practice, the map to the triangle's image space is arbitrary, but this abstraction is used for convenience because it also simplifies intersection testing.

The colour $\mathrm{s}_i(\boldsymbol{p})$, the point $\underline{\boldsymbol{p}}$ in the synthetic image s_i, is defined by the texture of the corresponding point $\boldsymbol{p}_j$ in the texture image t_j. Because the hypothesised surface is transformed to camera-coordinates, then $\lambda > 0$ for all points in front of the camera and, accordingly, the colour of $\underline{\boldsymbol{p}}$ is given by

$$\mathrm{s}_i(\boldsymbol{p}) \overset{def}{=} \mathrm{t}_k(\boldsymbol{H_k} \boldsymbol{E}_i^{-1} \lambda_k \underline{\boldsymbol{p}}), \tag{3.14}$$

where $\mathrm{t_k}$ is the texture image of the k^{th} triangle and k is the index denoting triangle that simultaneously satisfies Equations 3.10 and 3.12. The process of mapping texture from the triangle's texture space into the synthetic image is illustrated in Figure 3.6.

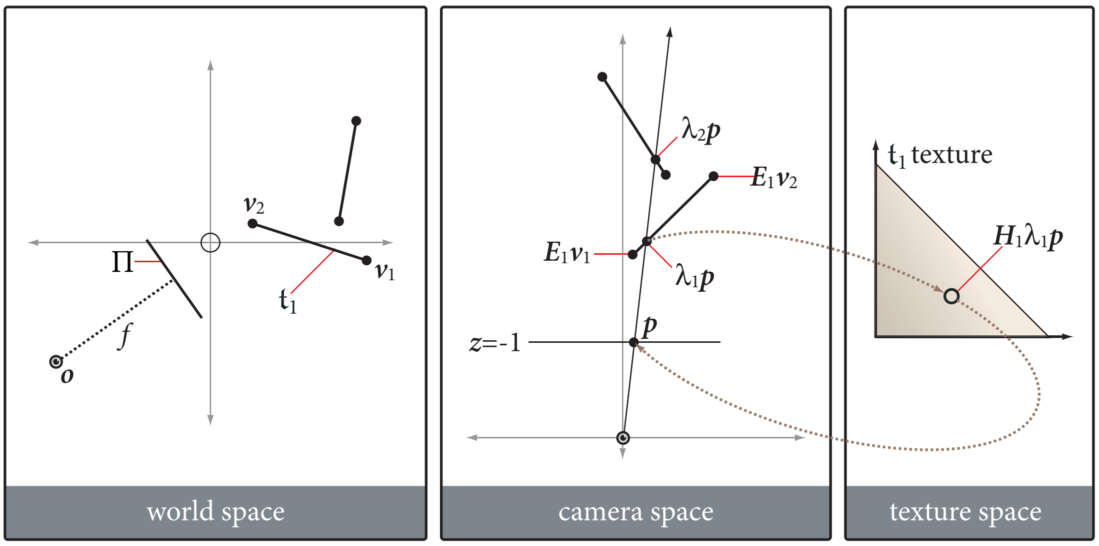


Figure 3.6: Projectively mapping texture to image-space by finding the nearest surface point to a camera and mapping it to its corresponding texture image.

3.3 The re-projection likelihood

Given s_i synthesised by rendering $\mathcal{S}(\Psi)$ using the reference cameras, the likelihood of generating $\mathcal{I}$ given Ψ is measured by pixel-wise comparisons between $\mathcal{I}$ and s_i. Assuming that the camera's sensor has a Gaussian noise model with standard deviation σ, then the likelihood of a point $\boldsymbol{p}$ in reference image is given by

$$\mathscr{L}(\mathsf{r}(\boldsymbol{p}) \mid \mathsf{s}(\boldsymbol{p})) = \frac{\exp\left(-\frac{(\mathsf{s}(\boldsymbol{p}) - \mathsf{i}(\boldsymbol{p}))^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}}, \tag{3.15}$$

which uses the model that each reference image is a noisy observation of the hypothesised surface reflectance that was used to generate synthetic images. We assume here that the difference between two colours can be meaningfully reduced to a scalar: the Euclidean distance in RGB space, for example. If every pixel is assumed to be independent, the likelihood with respect to the synthetic images is given by

$$\mathscr{L}(\mathcal{I} \mid \mathsf{s}_1, \cdots, \mathsf{s}_n) = \prod_j \prod_{\boldsymbol{p}} \mathscr{L}(\mathsf{r}_j(\boldsymbol{p}) \mid \mathsf{s}_j(\boldsymbol{p})). \tag{3.16}$$

This likelihood evaluating the similarity between the reference images and synthetic images of $\mathcal{S}(\Psi)$. Because this involves rendering the hypothesised surface using the reference cameras, the image likelihood from (3.4) is therefore given by

$$\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S}) \equiv \mathscr{L}(\mathcal{I} \mid \mathrm{s}_1, \cdots, \mathrm{s}_n). \tag{3.17}$$

This likelihood is based on the image formation model described in Section 3.2.3. Because the rendering equation is a function of the hypothesised surface's geometry in $\mathbb{R}^3$ and the surface texture image, the likelihood is therefore a distribution over the scene-graph parameters *and* texture reflectance. Suppose, however, that the scene-graph parameter vector is given by $\Psi = \Gamma \cup \Theta$, where Γ is the set of transformation parameters in $\mathcal{S}$, and Θ is the set of reflectance parameters in $\mathcal{C}$. Given this parametrisation, the posterior from (3.3) becomes

$$\Pr(\Psi \mid \mathcal{I}, \mathcal{S}) = \frac{\Pr(\mathcal{I} \mid \Theta, \Gamma, \mathcal{S}) \Pr(\Theta, \Gamma)}{\Pr(\mathcal{I})}. \tag{3.18}$$

Applying Bayes' rule to the image likelihood,

$$\Pr(\mathcal{I} \mid \Theta, \Gamma, \mathcal{S}) = \frac{\Pr(\Theta \mid \Gamma, \mathcal{I}, \mathcal{S}) \Pr(\mathcal{I})}{\Pr(\Theta)} \tag{3.19}$$

and substituting into (3.18) gives

$$\begin{aligned} \Pr(\Psi \mid \mathcal{I}, \mathcal{S}) &= \frac{\Pr(\Theta \mid \Gamma, \mathcal{I}, \mathcal{S}) \Pr(\mathcal{I}) \Pr(\Gamma) \Pr(\Theta)}{\Pr(\mathcal{I}) \Pr(\Theta)} && (3.20) \\ &= \Pr(\Theta \mid \Gamma, \mathcal{I}, \mathcal{S}) \Pr(\Gamma), && (3.21) \end{aligned}$$

which relates the likelihood of generating the reference images with the likelihood of the texture Θ. Rather than having a distribution over both texture and geometry, the posterior (3.3) can be described by the probability of the most likely texture given the hypothesised geometry.

3.3.1 *Estimating the surface texture*

The image formation model (3.14) depends on both surface geometry and texture. Instead of hypothesising both the geometry and its texture, suppose that the geometry was fixed and the problem was to estimate the most likely texture given the surface. Fixing the geometry defines the closest point to a given pixel from (3.10) and (3.12), which, in turn, defines the relationship between surface texture and synthetic image

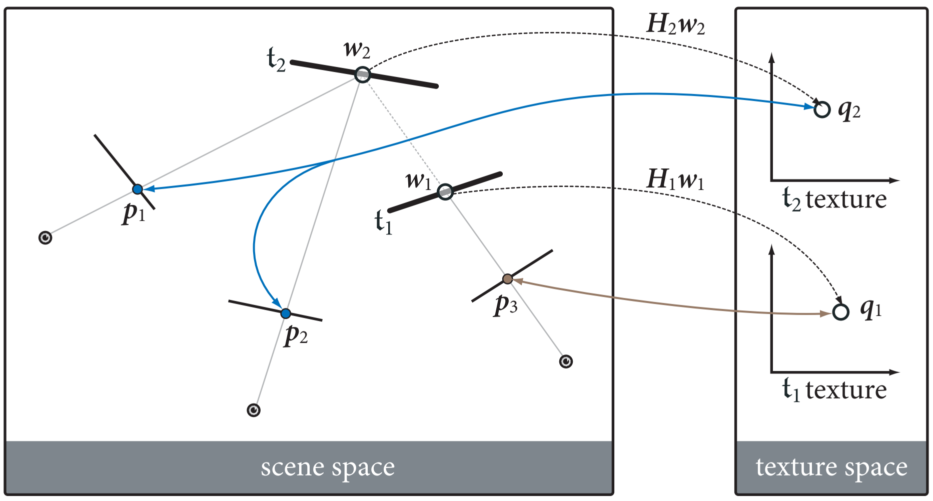


Figure 3.7: The photo-consistency constraint.

from (3.14). Because the corresponding point in the reference image i is a hypothesised observation of the surface texture, inverting the assignment in (3.14) yields

$$\mathrm{t}_k(\boldsymbol{H}_k\boldsymbol{E}_i^{-1}\lambda_k\underline{\boldsymbol{p}}) \overset{def}{=} \mathrm{r}_i(\boldsymbol{p}) \tag{3.22}$$

where, again, k denotes the triangle index belonging to closest surface-point to a given point $\underline{\boldsymbol{p}}$ in the reference image.

Equation (3.22) represents the photo-consistency constraint. If $\lambda_k\underline{\boldsymbol{p}}$ is visible by only one reference view, then the hypothesised surface colour is simply the assignment given by (3.22). In the case where a scene-point is visible by more than one camera, however, the photo-consistency constraint (3.22) must hold for *all* observations. That is, this constraint for a diffuse surface $\underline{\boldsymbol{q}} = \boldsymbol{H}_j\boldsymbol{E}_i^{-1}\lambda_i\underline{\boldsymbol{p}}_i\ \forall\ i$ is expressed as

$$\mathrm{t}(\underline{\boldsymbol{q}}) = \mathrm{r}_1(\boldsymbol{p}) = \mathrm{r}_2(\boldsymbol{p}) = \cdots = \mathrm{r}_n(\boldsymbol{p}). \tag{3.23}$$

The photo-consistency constraint is illustrated in Figure 3.7. Here, two surface-points, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are visible by one and two cameras respectively. Because $\boldsymbol{w}_1$ only projects to $\boldsymbol{p}_3$, then the reflectance $\mathrm{t}_1(\boldsymbol{q}_1)$ of the corresponding point $\boldsymbol{q}_1$ in texture space is given only by $\mathrm{r}_3(\boldsymbol{p}_3)$. In contrast, the surface-point $\boldsymbol{w}_2$ is visible by *two* surface-points, and therefore $\mathrm{t}_2(\boldsymbol{q}_2)$ must be consistent with both $\mathrm{r}_1(\boldsymbol{p}_1)$ and $\mathrm{r}_1(\boldsymbol{p}_3)$ if the hypothesis is correct.

Given that a surface-point has a number of observations across the set of reference images, the problem of estimating the most likely texture given $\mathcal{S}(\Psi)$ involves minimising the colour disparity of the surface's re-projection in all views. In the case of a diffuse reflectance model, a point's re-projection error is given by the distance in colour-space between the texture point's colour and the corresponding point's projection in all reference images. This re-projection error is minimised if the point's colour is defined by the average colour under its projection in all reference images. The surface texture for t_j is therefore constructed by

$$\mathsf{t}_j(\boldsymbol{w}) \overset{def}{=} \frac{\sum_i \mathsf{r}_i(\boldsymbol{H}_j^{-1}\boldsymbol{w})v_i(\boldsymbol{w})}{\sum_i v_i(\boldsymbol{w})}, \tag{3.24}$$

which computes the average observed colour of every surface point $\boldsymbol{w} \in \mathsf{t}_j$ on the surface of $\mathcal{S}(\Psi)$. The function

$$v_i(\boldsymbol{w}) = \begin{cases} 1, \text{if} \boldsymbol{E}_i\boldsymbol{w} \equiv \lambda_k\boldsymbol{P}_i\boldsymbol{w} \\ 0, \text{otherwise} \end{cases}, \tag{3.25}$$

where λ_k is the positive minimum that simultaneously satisfies Equations (3.10) and (3.12), is the visibility constraint such that only points in front of the camera contribute to the estimated reflectance of the scene-point $\boldsymbol{w}$.

3.4 Surface consistency on graphics hardware

Determining a point's photo-consistency involves projecting the point into each reference image; testing for occlusion; and measuring the variance of its image across the set of reference images in which it is visible. The surface's photo-consistency must be measured for each randomly sampled hypothesised parameter vector. A large number of surface hypothesis must be generated so that inferences about the scene-graph's parameter distribution can be made. This process is a mathematically straightforward but computationally intensive task.

The photo-consistency framework from Section 3.2 describes a method of computing a surface point's photo-consistent colour in a process not unlike multiple-pass rendering, a task well suited to graphics hardware. It would seem reasonable, then, that this paradigm could be adapted to graphics hardware. Graphics processors have evolved into highly parallel processors that are optimised for vector operations and capable of projecting and texturing millions of triangles per second [40]. Part of the

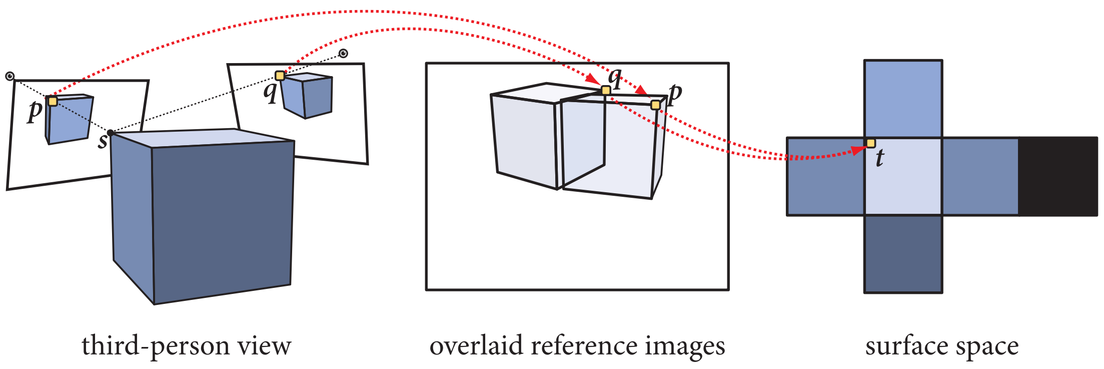


Figure 3.8: Because corresponding scene points are unlikely to align in image-space, they must be mapped to a common space for comparison.

motivation behind our reconstruction process is that it can exploit this dedicated, rendering hardware by using its forward mapping process to reason about the *inverse* problem of inferring the surface's parameters from the reference images.

One way of accessing graphics hardware is through the OpenGL API [50]. The key challenge with OpenGL's paradigm, however, is that it has a very specialised and unconventional programming model: the only mechanism for storing data, for example, is by writing fragments[1] from the frame-buffer. Furthermore, because OpenGL is designed for generating synthetic images, it is allowed to covertly manipulate data according to its preferred storage model. This may, for example, include clamping values, precision loss from type conversion, culling vertices and fabricating new vertices with interpolated attributes.

The key challenge in developing a photo-consistency likelihood on graphics hardware lies in adapting the hardware to collect the surface's hypothesised correspondences across the set of reference images. A point's observation across all reference images must be aligned in a common space to facilitate the process of estimating the surface reflectance model. In Figure 3.8, for example, a surface point $\boldsymbol{s}$ projects to non-overlapping points $\boldsymbol{p}$ and $\boldsymbol{q}$ in two reference images. These points must be mapped (either explicitly or implicitly) to a common point $\boldsymbol{t}$ for comparison.

Aligning corresponding points in a common co-ordinate frame is complicated by OpenGL's rigid pipe-line programming model. Graphics hardware can only project

[1]Fragments are point-samples over the rendered surface. More details concerning OpenGL's pipe-line is described in Appendix B

points via a single projection matrix at a time, and the result must be written into the frame-buffer to the pixel associated with the point's projection. In order to correlated hypothesised correspondences, however, surface points must be re-mapped to the frame-buffer in a manner independent on the point's projection in a given reference image. For this remapping process to work, there must be an invertible transform from the 3D surface to a plane so the frame-buffer can represent unique points *on the surface* rather than the point's projection *in the image*.

Unfortunately, although the vertex shader is free to arbitrarily remap points, much of OpenGL's functionality relies on data indexed by a fragment's projection in the frame-buffer. This means that if the point does not correspond to its projection in the synthetic image, then either the fragment shader must account for occlusion itself or the depth-buffer must be somehow distorted to compensate. The challenge, then, is to write the fragment data that is independent of the camera's projection matrix so that observations may be collated, while at the same time relying on data that can be only accessed only if the fragment *is* projected by the virtual camera parameters.

3.4.1 Adapting the likelihood

We have developed two approaches to compute the photo-consistency of the hypothesised surface to overcome OpenGL's limitations. The first likelihood, described in Chapter 4, does not explicitly estimate the surface's maximum likelihood texture image. Instead, new views are synthesised by using the projection of the hypothesised surface in two images to warps one reference image into the space of another. This process aligns the hypothesised projection of a surface point in two images, facilitating the comparison between the synthetic and reference images. The surface's photo-consistency is measured over the set of reference views by repeating this process for all pairs of reference images. The error metric is therefore a measure of the surface's *projection* rather than a measure of the surface photo-consistency.

The second likelihood described in Chapters 5—7 is a more direct representation of the likelihood described by Section 3.2. The hypothesised surface's texture is estimated by warping and compositing the reference images via the hypothesised surface. This likelihood is able to check the surface's visibility by directly identifying occluded scene elements. We make use of this additional information to ignore trivially photo-consistent solutions which are visible only in one view, thereby addressing an important limitation of the image-based approach.

Image-Space Photo-Consistency

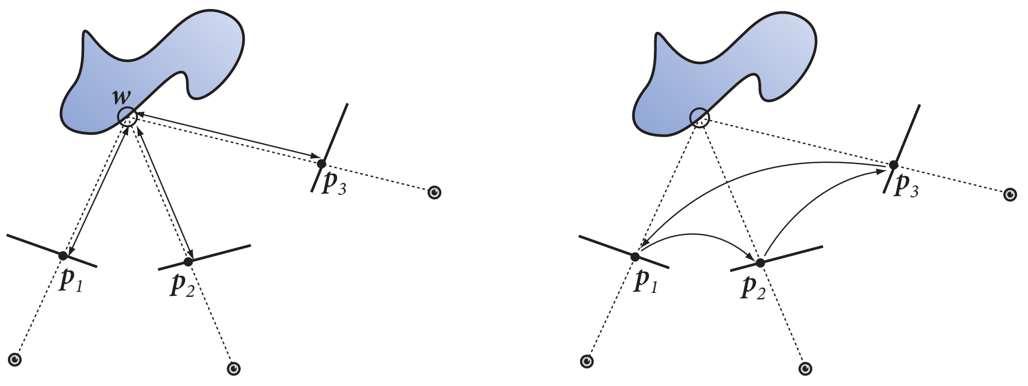
Chapter IV

The likelihood of generating the reference images is given by the disparity between the reference images and synthetic images of the hypothesised surface. Because the hypothesised surface is modelled by a triangle mesh and surface texture, the image likelihood is therefore a distribution over all transformation parameters and all possible textures. The relationship described in Section 3.2.3 between the surface texture and reference images can be exploited, turning the image likelihood into a problem of estimating the maximum likelihood texture given the reference images and hypothesised geometry.

Finding the maximum likelihood texture is a problem of minimising the variance of the surface's projection in all reference images. Instead of explicitly representing the texture image, however, our approach compares the hypothesised surface's projection in two views by texturing the hypothesised surface using a 'source' reference image and rendering the result using a 'target' camera. This process effectively registers one reference image into the space of another, facilitating the process of computing pixel-wise comparisons. Repeating this process for all pairs of reference images yields a photo-consistency metric which approximates the variance of the surface's reflectance across the set of reference images. This metric is used to find the most likely photo-consistent surface.

4.1 Stereo photo-consistency

Under a purely diffuse reflectance model, a point's photo-consistency is measured by the variance of its projection in each reference image. Rather than explicitly compute the average observation for each scene point, variance can be approximated by measuring the disparity between pairs of observations; this process is illustrated in Figure 4.1. Although not a direct measure of variance, the comparison of all stereo-pairs is a function that monotonically increases as the variance of the set of hypothesised



(**a**) Direct representation of the surface's variance

(**b**) Stereo photo-consistency

Figure 4.1: The variance of $\boldsymbol{w}$ can be approximated by pairwise comparison of its projection in all views.

observations also increases. The advantage of this approach is that it only relies on comparing two observations, rather than comparing an observations with a texture that has to be computed and stored.

The comparison between hypothesised point-correspondences is facilitated by distorting one reference image into the space of another. A synthetic image is generated by back-projecting one reference image (the 'source') while rendering $\mathcal{S}(\Psi)$ using a 'target' reference camera. If a point is visible in the source image, then the colour denoted by the point's projection in the first image is mapped to the point's projection in the second. Otherwise, the point's projection in the target image is tagged to indicate that the pixel back-projects to a scene-point visible only by the destination camera. The result is a synthetic image of the surface seen by both cameras, but with the surface colour defined using the source reference image.

The process of generating a synthetic view by imaging the hypothesised surface using one reference camera, while implicitly texturing the surface with another, is illustrated in Figure 4.2. Two cameras in this example observe three world points $\boldsymbol{w}_i$ that project to two unique points in each image. Because the scene point $\boldsymbol{w}_2$ is visible by both cameras, the colour denoted by its projection $\boldsymbol{p}_2$ in the first image is mapped to its projection $\boldsymbol{p}'_2$ in the second image. The projection of $\boldsymbol{w}_3$ is also coincident with $\boldsymbol{p}_2$, but the image colour corresponding to $\boldsymbol{p}_2$ is not mapped to $\boldsymbol{p}'_3$ ($\boldsymbol{w}_3$'s projection in the second image) because $\boldsymbol{w}_3$ is occluded by $\boldsymbol{w}_2$ with respect to the source camera. Instead, the pixel at $\boldsymbol{p}'_3$ is tagged to indicate that $\boldsymbol{w}_3$ is occluded.

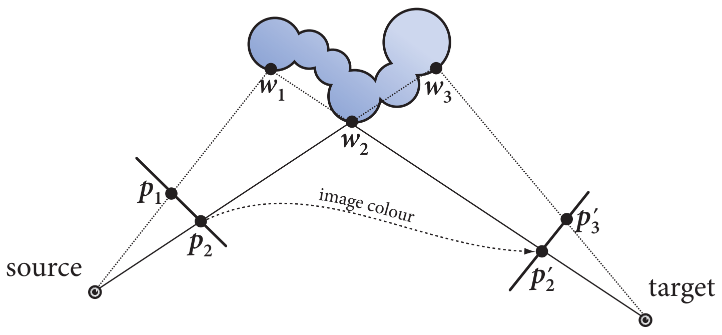


Figure 4.2: The colour under a point's projection in the source image is mapped to its projection in the target image if it is visible in both images.

4.2 Image mapping via the hypothesised surface

Synthetic images of the scene are generated by projectively mapping triangles from the source camera to the virtual target camera via homographies defined by the hypothesised surface's projection in both images. Points in the triangle's projection in the target image are assigned colour from the source image through this homography; but this mapping is valid only if the corresponding surface point is visible in both views. Accordingly, the fragment of every point rasterised in the synthetic image must be tested for occlusion in *both* cameras before projectively mapping colour from the source to target images.

Given the projection matrices $\boldsymbol{P}'$ and $\boldsymbol{P}$ from the '*source*' and '*target*' cameras respectively, the homography $\boldsymbol{H}_i$ relating two views of the surface triangle $\mathfrak{t}_i \in \mathcal{S}(\Psi)$ is defined by

$$\boldsymbol{H}_i \boldsymbol{P} \boldsymbol{v}_j = \boldsymbol{P}' \boldsymbol{v}_j \tag{4.1}$$

$$\boldsymbol{H}_i = \begin{bmatrix} \boldsymbol{v}_1^\top \boldsymbol{P}^\top \\ \boldsymbol{v}_2^\top \boldsymbol{P}^\top \\ \boldsymbol{v}_3^\top \boldsymbol{P}^\top \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{v}_1^\top \boldsymbol{P}'^\top \\ \boldsymbol{v}_2^\top \boldsymbol{P}'^\top \\ \boldsymbol{v}_3^\top \boldsymbol{P}'^\top \end{bmatrix}^{-\top} \tag{4.2}$$

where $\boldsymbol{v}_j$ are the 3D vertices of $\mathfrak{t} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$.

Let the triangles $\mathfrak{u} = \{\boldsymbol{P}'\boldsymbol{v}_j\}$ and $\mathfrak{v} = \{\boldsymbol{P}\boldsymbol{v}_j\}$ be the projection of the triangle $\mathfrak{t}$ in the source and target cameras respectively. The synthetic image $\mathfrak{s}_{s,t}$ is given by mapping every point $\boldsymbol{p} \in \boldsymbol{u}$ from the source reference image r into $\boldsymbol{v}$ via

$$\mathsf{s}_{s,t}(\boldsymbol{p}) \leftarrow \mathsf{r}_s(\boldsymbol{H}_i\boldsymbol{p}), \tag{4.3}$$

provided that the corresponding surface point $\boldsymbol{q} \in \mathfrak{t}$ whose projection is $\boldsymbol{Pq} = \boldsymbol{p}$ is visible with respect to $\mathcal{S}(\Psi)$ in both source and target cameras.

4.2.1 Image mapping with occlusion

Testing whether a surface point is occluded from the view-point of a particular camera involves determining whether there exists a point with a coincident projection that is closer to the camera's optical centre. Rather than traversing the scene for every occlusion query, this test is greatly simplified by storing a *depth-map* of the hypothesised surface for both the source and target views. The depth-map stores the distance of the closest surface point in front of the camera for each pixel in the image. If the projection matrix $\boldsymbol{P}$ of camera c is decomposed into the intrinsic and extrinsic matrices $\boldsymbol{K}$ and $\boldsymbol{E}$, then the depth-map z_c with respect to the surface $\mathcal{S}$ and camera c is defined as

$$\mathsf{z}_c(\boldsymbol{p}) \leftarrow \min\{d_c(\boldsymbol{q}) \mid \boldsymbol{q} \in \mathcal{S}(\Psi), \boldsymbol{KEq} = \boldsymbol{p}, f(\boldsymbol{Eq}) < 0\}, \tag{4.4}$$

where $d_c(\cdot)$ is a distance function from $\boldsymbol{s}$ to the optical centre of camera c. The constraint $f(\boldsymbol{q}) < 0$, where

$$f(\boldsymbol{q}) = \frac{[0,0,1,0]\,\boldsymbol{q}}{[0,0,0,1]\,\boldsymbol{q}}, \tag{4.5}$$

ensures that only the distance of points in front of the camera are stored in the depth-map.

If the depth-maps z_s and z_t are generated by rendering the hypothesised surface from the source and target cameras to generate respectively, then the point $\boldsymbol{p}' = \boldsymbol{P}'\boldsymbol{s}$, $\boldsymbol{p}' \in \mathfrak{u}$ is mapped to the corresponding point $\boldsymbol{p} = \boldsymbol{Ps}$, $\boldsymbol{p} \in \mathfrak{v}$ in the synthetic image via (4.3) i.f.f.

$$d_t(\boldsymbol{s}) \le \mathsf{z}_t(\boldsymbol{p}) \text{ and} \tag{4.6a}$$

$$d_s(\boldsymbol{s}) \le \mathsf{z}_s(\boldsymbol{p}') \tag{4.6b}$$

holds.

The occlusion test requires the distance from $\boldsymbol{s}$ to the synthetic and source images

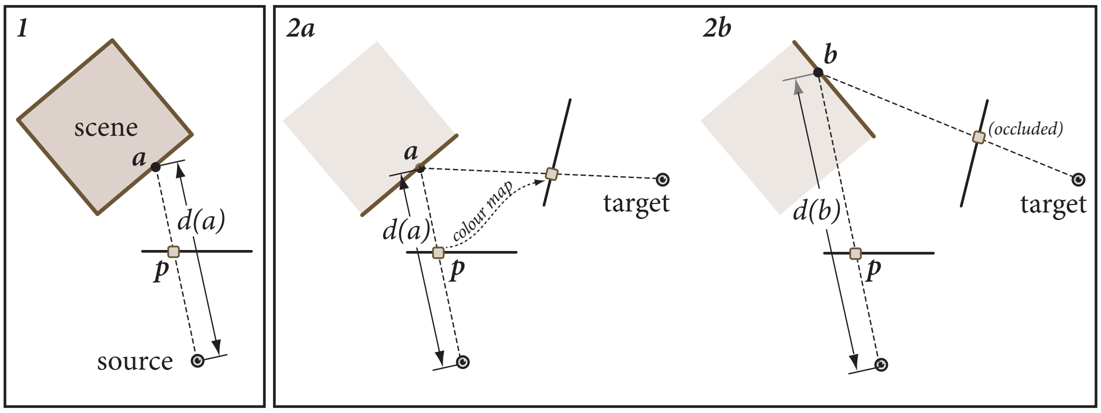


Figure 4.3: A depth map is generated by rendering the hypothesised surface using the 'source' reference camera *(1)*. The colour under a point's projection in the source image is mapped into the target image if the 3D point passes the depth test with respect to the source view *(2a)*, otherwise its projection is marked as occluded *(2b)*.

for its projection $\boldsymbol{p}$ and $\boldsymbol{p}'$ respectively. The distance $d(\boldsymbol{s})$ can be inferred by linear interpolation of the projected triangle's vertices. Provided $\mathfrak{v}$'s vertices are not collinear, the triangle's edges can be treated as basis vectors spanning every point in the triangle's image. Representing a point $\boldsymbol{p} \in \mathfrak{v}$ by two scalars a and b such that

$$\boldsymbol{p} = a(\boldsymbol{P}\boldsymbol{v}_3 - \boldsymbol{P}\boldsymbol{v}_1) + b(\boldsymbol{P}\boldsymbol{v}_3 - \boldsymbol{P}\boldsymbol{v}_1). \tag{4.7}$$

suggests the depth of the corresponding surface point $\boldsymbol{s} \in \mathfrak{t}$ such that $\boldsymbol{p} = \boldsymbol{P}\boldsymbol{s}$ is given by

$$d_t(\boldsymbol{s}) = (1 - a - b)d_t(\boldsymbol{v}_1) + bd_t(\boldsymbol{v}_2) + ad_t(\boldsymbol{v}_3). \tag{4.8}$$

The same process can be used with $\mathfrak{u}$ to infer the depth $d_s(\boldsymbol{s})$ of the corresponding point $\boldsymbol{p}' = P'\boldsymbol{s}$ in the source image.

4.2.2 *Generating synthetic images*

The synthetic image $\mathfrak{s}_{s,t}$ is generated by mapping the projection of every hypothesised surface triangle from the reference image associated with camera s into a new view image by the camera t. Each point in a triangle's projection in the target image is assigned the colour from the corresponding point's projection in the reference image, provided that the surface point is visible in both images. A pixel $\boldsymbol{p} \in \mathfrak{s}_{s,t}$ is undefined if it corresponds to a scene-point that is not visible in both images is undefined. Accordingly, *two* synthetic images are generated: $\mathsf{s}_{s,t}$ stores the colour mapped from r_s, and $\alpha_{s,t}$ is a 'visibility map' to denote which pixels in $\mathsf{s}_{s,t}$ correspond to points visible in both cameras.

The process of mapping the pixels from the source image into the target image via the hypothesised surface is illustrated in Figure 4.3. The depth map z_s is generated by rendering $\mathcal{S}(\Psi)$ using the source camera. The hypothesised surface is then rendered using the target camera. If the surface point's distance is equal to the depth stored in z_s, then the colour under the point's projection in the source image is mapped to $\mathsf{s}_{s,t}$, and the corresponding point in $\alpha_{s,t}$ is tagged as 'visible' (*2a*). If the distance of a scene-point is greater than the distance in z_s, then the projection of the point in the target image $\mathsf{s}_{s,t}$ is undefined, but the corresponding point in $\alpha_{s,t}$ is marked as 'occluded' (*2b*).

4.3 Graphics implementation

The rendering model described in Section 4.2 generates synthetic images of the hypothesised surface by mapping triangles from the source to target views. This task can be readily accomplished in graphics hardware. Configuring OpenGL's projection matrix to correspond to the target camera will generate $\mathfrak{v}$ as $\mathfrak{t}$ is sent to the graphics pipe-line. The source image is mapped to $\mathfrak{v}$ by rendering $\mathfrak{t}$ into the target image and using OpenGL's texture co-ordinate generator to map $\mathfrak{t}$'s projection in the source image into the frame-buffer.

The fragment processor automatically generates the linear interpolation from (4.8) to map the reference image into the synthetic image. Occlusion with respect to the target camera is handled by the depth buffer while *shadow mapping* is used to handle occlusion with respect to the source camera. This implements the two visibility tests from (4.6) so that points visible by both cameras are correctly mapped to the target image.

4.3.1 Stereo occlusion

Correctly transferring a surface point's colour from the source to the target image requires two occlusion tests. Although depth-testing can only implement occlusion with respect to the target image as the hypothesised surface is rendered, occlusion with respect to the source reference image is implemented with OpenGL's support

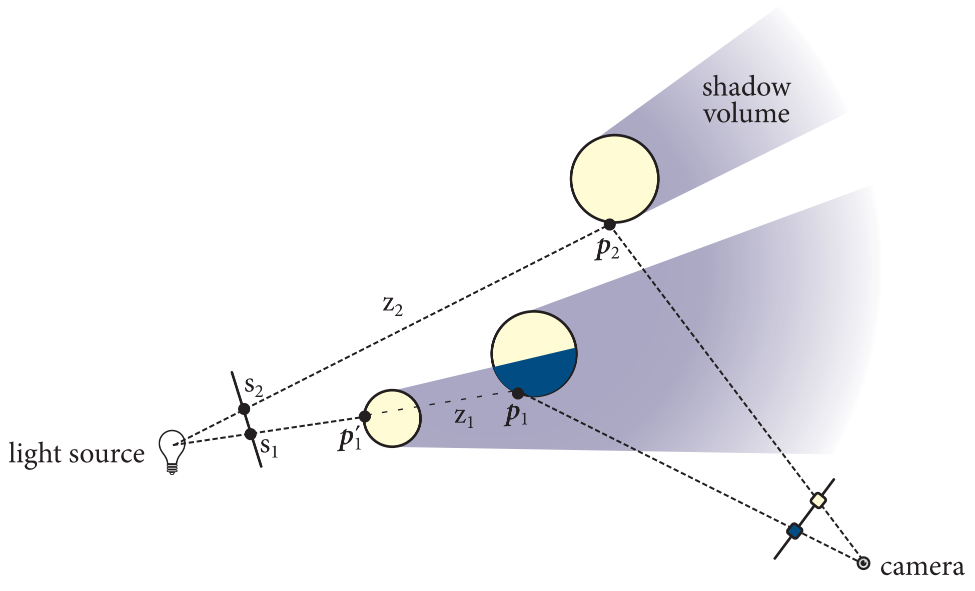


Figure 4.4: Shadows are rendered by testing the depth of a candidate point with its projection in the shadow-map. In this case, $\boldsymbol{p}_1$ is in shadow since $\{z_1 = d(\boldsymbol{p}_1)\} > \{s_1 = d(\boldsymbol{p}_1')\}$.

for shadow rendering. OpenGL supports shadow by conditionally applying effects to fragments as triangles are rendered by the virtual camera.

In the context of rendering shadows, a given surface is first rendered from a light's perspective to generate a depth map of every lit surface-point. The scene is then rendered from the virtual camera's perspective with this depth map bound as a *shadow map*. Scene-points are occluded with respect to the light-source if they fail a depth-test with respect to the light's projection matrix and shadow-map. In Figure 4.4, for example, the depth of the point $\boldsymbol{p}_1$ would be stored as s_1, and therefore rendered in shadow because $z_1 > s_1$. Shadows maps are bound as textures and referenced through texture co-ordinates. Unlike depth testing, shadow mapping cannot remove fragments from the fragment stream. Instead, the fragment's α-channel is modified depending on whether it succeeds the shadow-test.

The source reference image is conditionally transferred to the target image, subject to both occlusion tests from Equations (4.6), by combining the shadow-test *and* reference image colour into a single RGBα fragment. The depth-map from (4.4) is computed by rendering the hypothesised surface using the source camera. The depth-buffer is then stored as a shadow map. The hypothesised surface is then rendered using the target camera and textured by combining the source reference image and shadow-test. A fragment's colour is then always defined by its corresponding projection into the source image, but its α-channel is a function of its occlusion test with respect to the source camera. Accordingly, pixels in the target image with $\alpha = 1$ correspond to scene-points visible in both cameras, whereas pixels with $\alpha = 0$ correspond to points visible in the target camera but occluded with respect to the source camera.

4.3.2 *Configuring the graphics pipe*

Generating the synthetic image of the hypothesised scene requires two render passes. The source camera's depth map is generated in the first render pass. Since the hypothesised scene is rendered from the source camera's point-of-view, then the graphics projection ($\boldsymbol{P}_{\mathcal{G}}$) and model-view ($\boldsymbol{M}_{\mathcal{G}}$) matrices are defined as $\boldsymbol{P}_{\mathcal{G}} \leftarrow \hat{\boldsymbol{K}}'$ and $\boldsymbol{M}_{\mathcal{G}} \leftarrow \boldsymbol{E}'$ respectively. The matrix $\hat{\boldsymbol{K}}'$ is an OpenGL-class 4×4 projection matrix that projects vertices to the graphics hardware's 3D co-ordinate system. If $\boldsymbol{K}$ is a 3×4 projection matrix, then $\hat{\boldsymbol{K}}$ is in the form

$$\hat{\boldsymbol{K}} = \begin{bmatrix} K_{1,1} & K_{1,2} & K_{1,3} & K_{1,4} \\ K_{2,1} & K_{2,2} & K_{2,3} & K_{2,4} \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ K_{3,1} & K_{3,2} & K_{3,3} & K_{3,4} \end{bmatrix}, \tag{4.9}$$

which projects homogeneous surface points into homogeneous projective space, but includes a function of depth for testing against the depth-buffer.

The hypothesised surface's depth map z_s with respect to the source camera is generated by rendering $\mathcal{S}(\Psi)$ with the source camera. The depth-map is copied and bound as a shadow map and the pipe reconfigured to render the surface using the target camera. Dual texturing is used to map the source reference onto to the surface subject to a point's visibility in the source camera. The texture generation matrix is defined by

$$\boldsymbol{T}_{\mathcal{G}} = \boldsymbol{B}\hat{\boldsymbol{K}}'\boldsymbol{E}', \tag{4.10}$$

where $\boldsymbol{B}$ is

$$\boldsymbol{B} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.11}$$

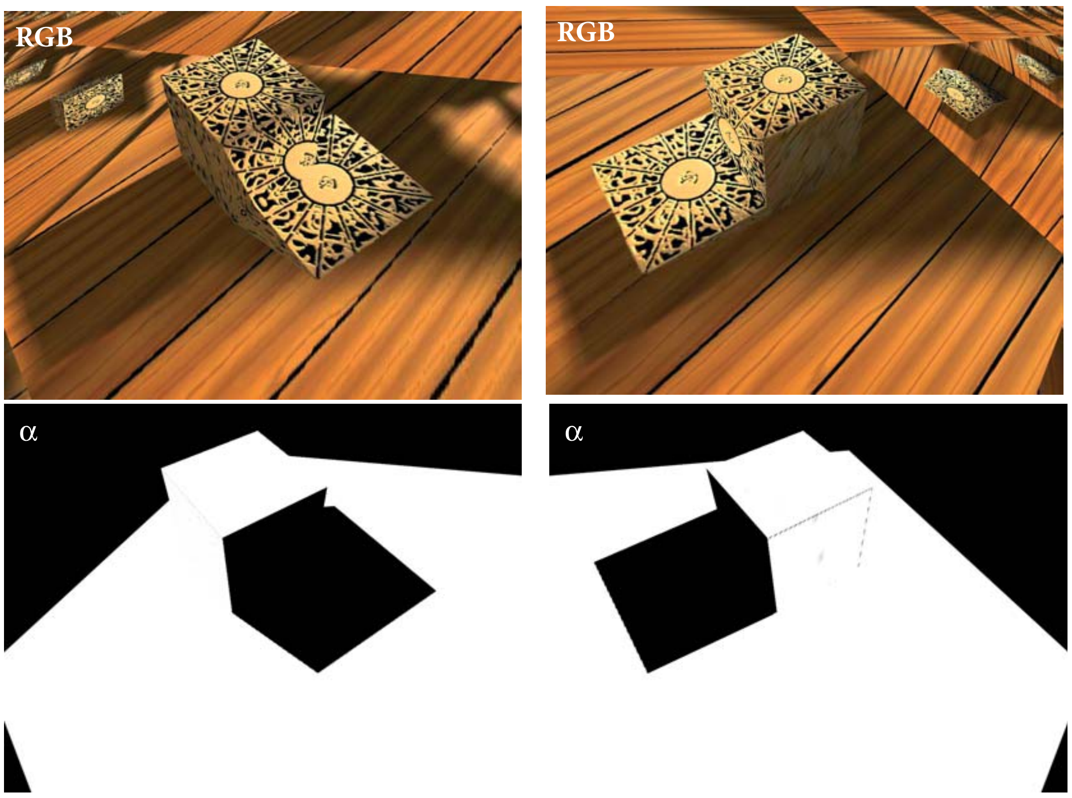
Algorithm 1: Warping a single image into a new synthetic view

```
Input : $\{\mathfrak{v}_j\}$—the set of surface triangles in the world co-ordinate system;
        $\mathsf{r}_s$—the source reference image; $\boldsymbol{P}' = \boldsymbol{K}'\boldsymbol{E}'$ and $\boldsymbol{P} = \boldsymbol{KE}$—the source
        and target projection matricies.
Output: $\mathsf{s}_{s,t}$ and $\alpha_{s,t}$—the target colour and $\alpha$-channel images generated by
        warping $\mathsf{r}_s$.
/* First pass: generate the depth-map for the source camera */
1  $\boldsymbol{P}_\mathcal{G} \leftarrow \hat{\boldsymbol{K}}'$; $\boldsymbol{M}_\mathcal{G} \leftarrow \boldsymbol{E}'$
2  render($\{\mathfrak{v}_j\}$)
3  $\mathsf{z}_s \leftarrow$ depthbuffer
/* Second pass: render the surface from the target view */
4  $\boldsymbol{P}_\mathcal{G} \leftarrow \hat{\boldsymbol{K}}$; $\boldsymbol{M}_\mathcal{G} \leftarrow \boldsymbol{E}$
5  $\boldsymbol{T}_\mathcal{G} \leftarrow \boldsymbol{B}\boldsymbol{P}'_\mathcal{G}\hat{\boldsymbol{K}}'$
6  bind $\mathsf{z}_s$, set its coordinate generation mode to GL_CLAMP_TO_BORDER, its
   compare mode to ≤GL_COMPARE_R_TO_TEXTURE and shadow application
   mode to GL_ALPHA;
7  bind $\mathsf{r}_s$
8  reset the depth-buffer to $+\infty$
9  enable clipping plane source image clipping plane
10 render($\{\mathfrak{v}_j\}$)
```

the 'bias' matrix to map OpenGL's device coordinate system into its texture space [1]. Clipping against the source camera's frustum is implemented by clamping texture co-ordinates to the shadow texture's boundary.

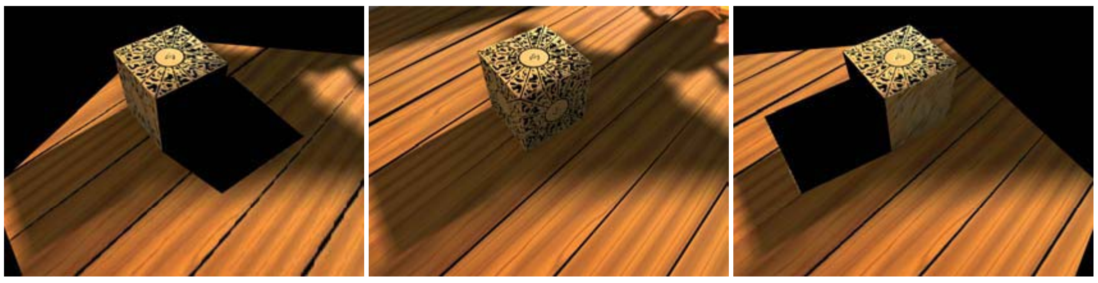
The algorithm to generate synthetic images of the hypothesised surface is listed in Algorithm 1. Figure 4.5 illustrates two synthetic images of the hypothesised scene at truth. The target image in both cases is the first reference image from Figure 4.10 using, as the source texture, the second and third reference views respectively. The top pair of images is the colour map from (4.3). The image below is the α-channel which effectively implements (4.6b): the occlusion from the source camera. White pixels indicate image points that back-project to surface points visible by both cameras. Black pixels indicate $\alpha = 0$ where fragments are occluded with respect to the surface and the source view. The α-channel combined with the colour image defines the view of the hypothesised scene in the target reference image while textured by a source reference image.

[1]The texture domain is within [0,1] along all three principal axes.



(a) Using reference image 1 as the source. **(b)** Using reference image 2 as the source.

Figure 4.5: Generating synthetic views of the surface at truth. The target view in both cases is the second reference image using the images from 4.10. The α image is the surface's visibility map.



synthetic view 1 reference image synthetic view 2

Figure 4.6: The juxtaposition of two synthetic views with its corresponding target reference image at truth.

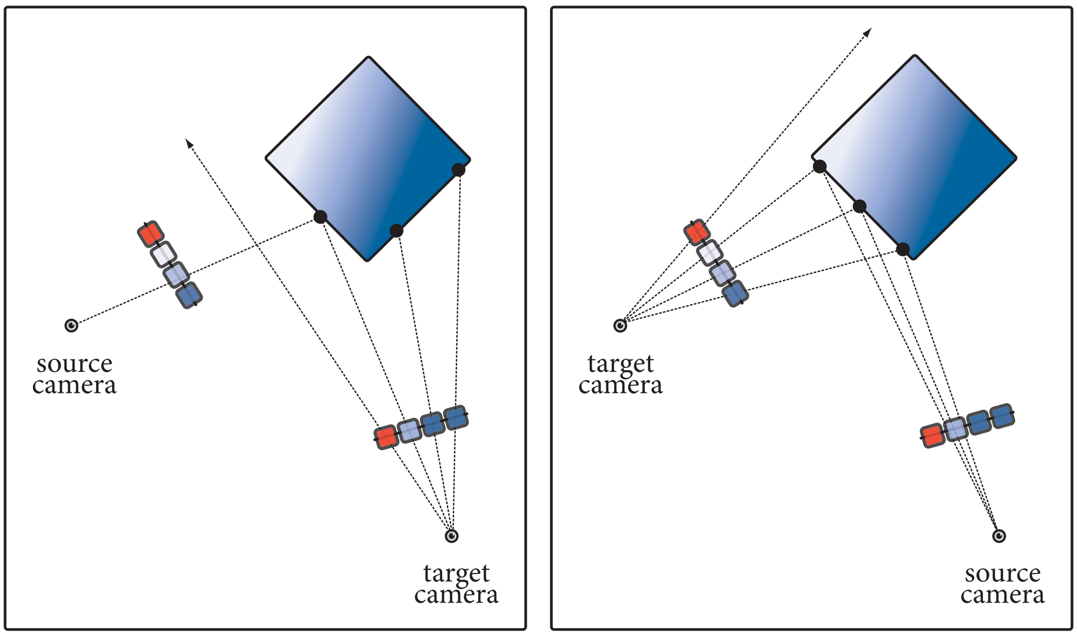


Figure 4.7: The surface's sampling rate is driven by the target image's resolution.

4.4 Shape inference with photo consistency

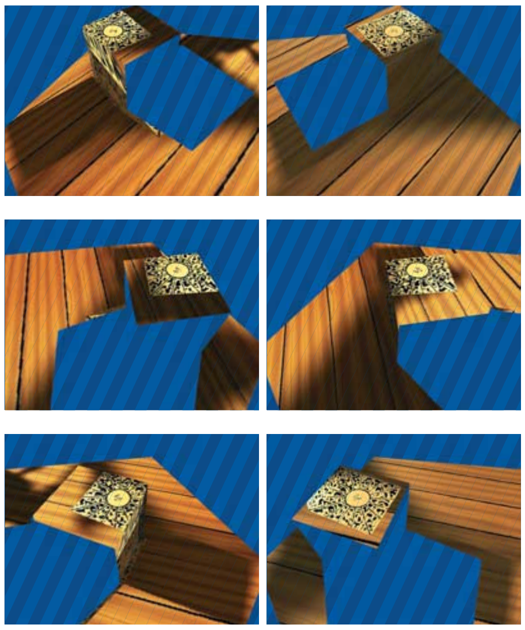
The synthetic image facilitates testing photo-consistency by aligning hypothesised corresponding points in the target camera's image space. Figure 4.6 illustrates the similarity between synthetic and target images of a hypothesised scene at truth. In this example, pixels that back-project to scene-points occluded in the source image are black, but are marked accordingly in the image's α-channel.

The likelihood of generating the target image from $\mathcal{S}(\Psi)$ is based on pixel-wise comparison of the target and synthetic images. In this stereo configuration, the estimated texture of an observed texel with colours $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ is $\bar{\boldsymbol{c}}$. The photo-consistency metric from (3.15) gives

$$\epsilon = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\boldsymbol{c}_1 - \boldsymbol{c}_2\|}{4\sigma^2}\right) \tag{4.12}$$

which is a function of the colour disparity between the two images.

The likelihood relies on pairwise comparison of hypothesised correspondences projected in the target image, and it is not clear how to define the likelihood of a pixel that back-projects to a scene-point that is occluded in the source camera. The problem of defining a suitable occlusion likelihood is a key limitation of this image-


(a) RGB images.

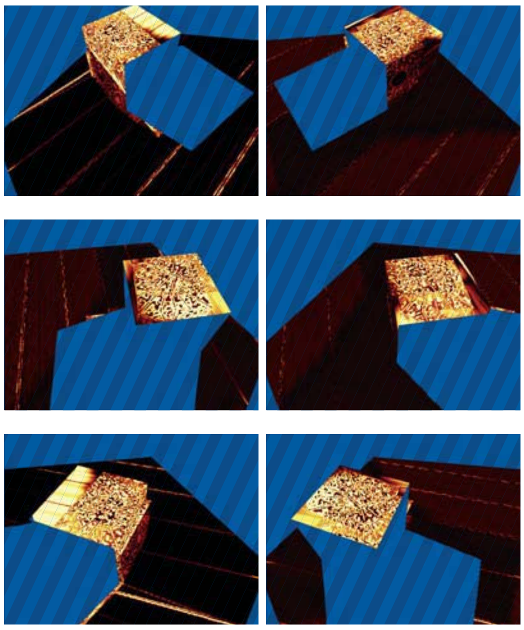
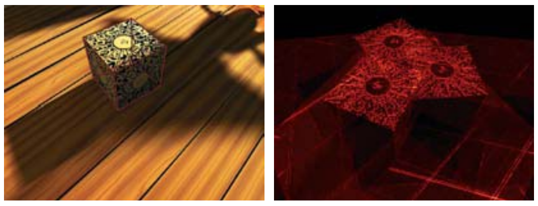
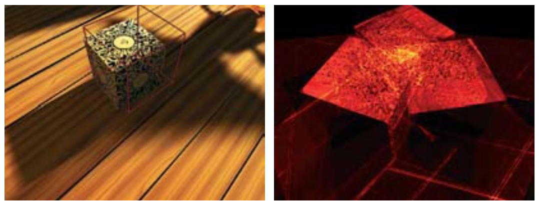
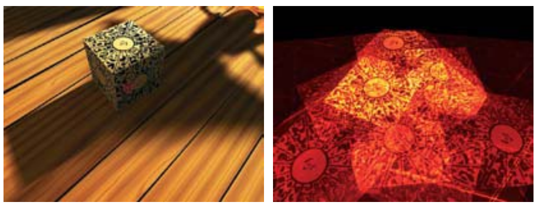

(b) Consistency images.

Figure 4.8: The six synthetic and consistency images for the hypothesised parameter vector illustrated in Figure 4.9(b). Image points where $\alpha = 0$ are indicated in blue. The three synthetic reference images used for this example are illustrated in Figure 4.10.


(a) *truth*


(b)


(c)

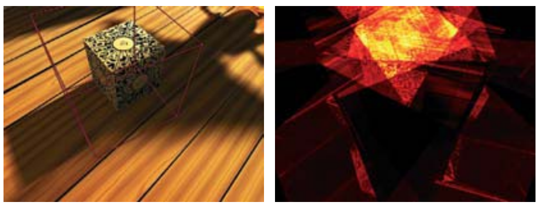

(d)

Figure 4.9: Visualisation of a hypothesised parameter vector *(left)* and the 'accumulated consistency image' *(right)*.

space likelihood. The term for zero disparity from (4.12),

$$\mathscr{L}(\alpha_{s,t}(\boldsymbol{p}) = 0) = \frac{1}{\sigma\sqrt{2\pi}}, \tag{4.13}$$

is used as the occlusion likelihood.

Given the target reference image r_t, the occlusion map $\alpha_{s,t}$, and synthetic image $\mathsf{s}_{s,t}$, then the likelihood of generating the target reference image r_t given $\mathcal{S}(\Psi)$ is

$$\mathscr{L}(\mathsf{r}_t \mid \mathsf{s}_{s,t}, \alpha_{s,t}, \Psi, \mathcal{S}) = \prod_{\boldsymbol{p}} \mathscr{L}(\mathsf{r}_t(\boldsymbol{p}) \mid \mathsf{s}_{s,t}(\boldsymbol{p}), \alpha_{s,t}(\boldsymbol{p}), \Psi, \mathcal{S}). \tag{4.14}$$

where the per-pixel likelihood

$$\mathscr{L}(\mathsf{r}_t(\boldsymbol{p}) \mid \mathsf{s}_{s,t}(\boldsymbol{p}), \alpha_{s,t}(\boldsymbol{p}), \Psi, \mathcal{S}) = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} & \text{if } \alpha_{s,t}(\boldsymbol{p}) = 0 \\ \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\mathsf{s}_{s,t}(\boldsymbol{p}) - \mathsf{r}_t(\boldsymbol{p})\|}{4\sigma^2}\right) & \text{otherwise} \end{cases} \tag{4.15}$$

assumes a perfectly diffuse scene viewed by a sensor with a Gaussian noise model whose standard deviation is σ.

The hypothesised surface is verified against all reference images by considering all pairs of source and target views, giving the likelihood

$$\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S}) = \prod_{u} \prod_{v, v \neq u} \mathscr{L}(\mathsf{r}_v \mid \mathsf{s}_{u,v}, \alpha_{u,v}, \Psi, \mathcal{S}). \tag{4.16}$$

All pairs of reference images must be considered because the surface's sampling rate is controlled by its projection in the target camera. The surface in Figure 4.7, for example, is sampled at different frequencies depending on both the scene occlusion and orientation of the imaging plane, and therefore

$$\mathscr{L}(\mathsf{r}_t \mid \mathsf{r}_s, \alpha_{s,t}, \Psi, \mathcal{S}) \neq \mathscr{L}(\mathsf{r}_s \mid \mathsf{s}_{t,s}, \alpha_{t,s}, \Psi, \mathcal{S}). \tag{4.17}$$

The permutation of all pairs of reference images must be considered to ensure that all hypothesised correspondences are included in the likelihood.

The image-space photo-consistency metric is illustrated in Figures 4.8 and 4.9. The six source/target image pairs of hypothesised parameter vector for the sequence in Figure 4.10 are illustrated in Figure 4.8(a). The corresponding photo-consistency metric for these images is visualised in Figure 4.8(b), where black pixels indicate zero photo-consistency error between the synthetic and reference images, and increasing

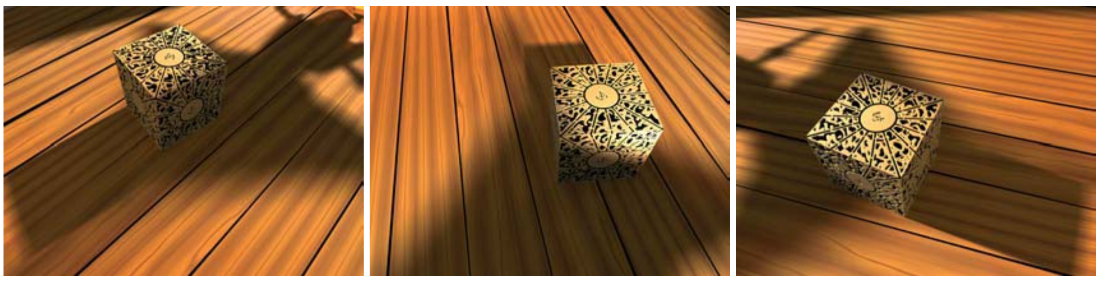


Figure 4.10: The three synthetic reference images used for testing the image-space likelihood.

red/yellow indicates increasing levels of photo-inconsistency. In both cases, blue pixels indicate the projection of surface points that are occluded in the source camera. The 'accumulated consistency image', generated by accumulating the six consistency images for a given hypothesised surface, is illustrated in Figure 4.9 for four parameter vectors.

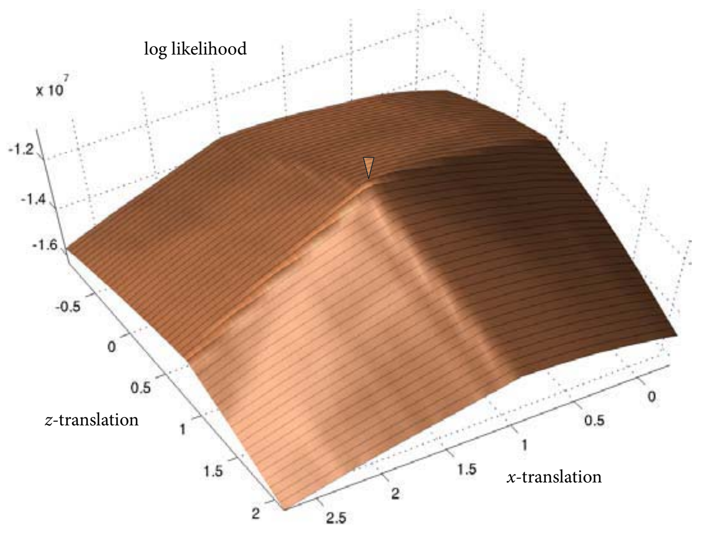
4.5 Experiments

The image-space likelihood (4.16) was used to estimate the scene parameters of the synthetic sequence illustrated in Figure 4.10. This sequence, consisting of a cube visible by three cameras, has 25 parameters if constant camera intrinsics and zero skew are assumed: three intrinsic parameters for focal length and optical centre; six extrinsic parameters per camera for rotation and translation; and four parameters to describe the cube's rotation, translation and uniform scale.

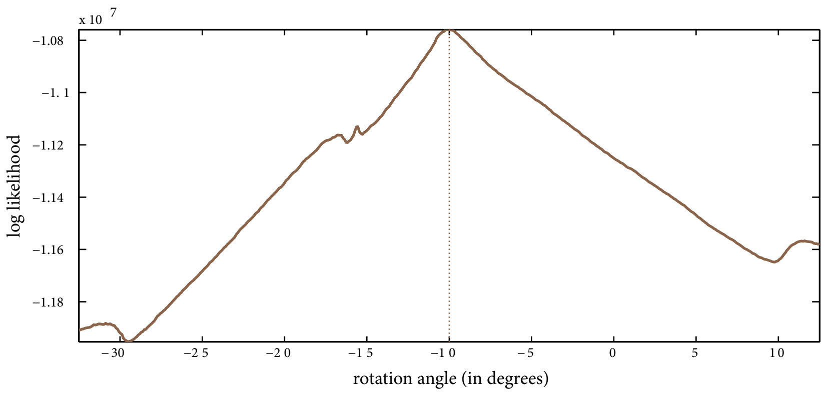
The image-space likelihood was tested by varying a subset set of scene-parameters while holding the remainder at truth. The graphs are illustrated in Figures 4.11(a)—4.12(b); the true parameter is at the centre of the x-axis in each case:

- *Figure 4.11(a)*—the cube's translation is varied along the ground plane;
- *Figure 4.11(b)*–the cube is allowed to rotate about the y–axis;
- *Figure 4.11(c)*–the cube's scale is varied;
- *Figure 4.12(a)*–the optical centre is changed while assuming constant intrisincs;
- *Figure 4.12(b)*–the focal length shared by all cameras is changed.

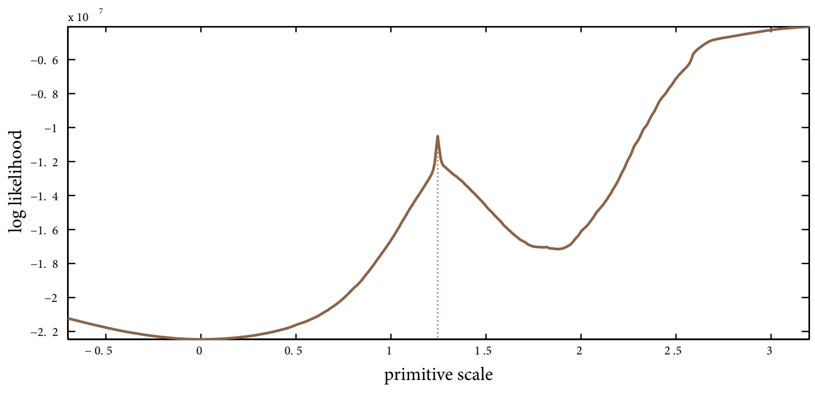
The tests illustrate a clear local maxima for each respective test. The scale (Figure 4.11(c)) and optical centre (Figure 4.12(a)), however, appear to have likely solutions



(a) The cube's translation over the ground plane.

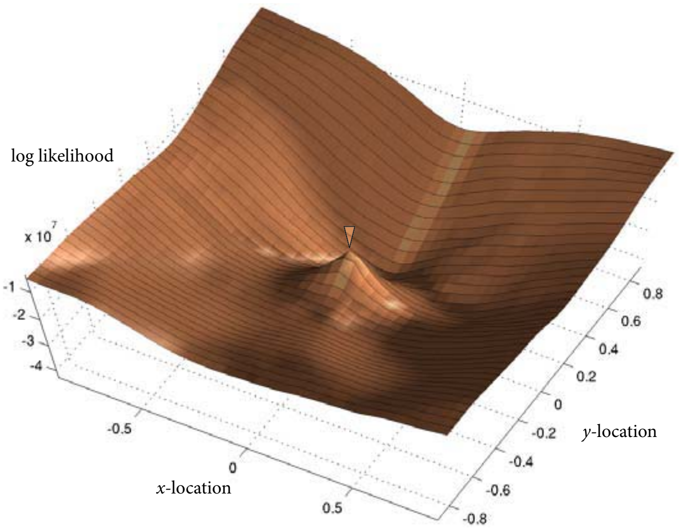


(b) The cube's rotation about its vertical axis.

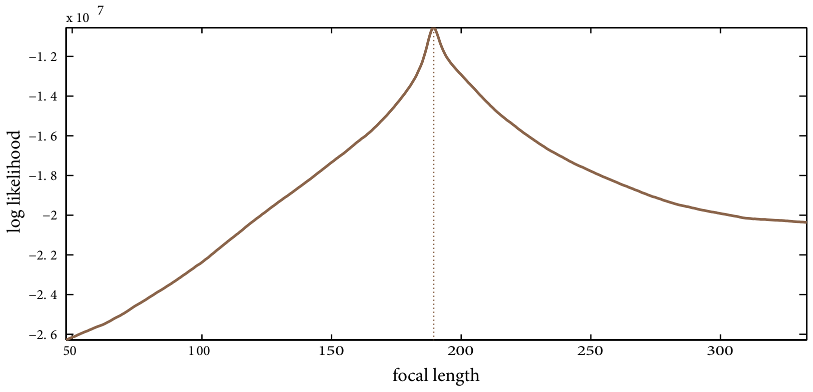


(c) The cube's scale relative to the primitive's local co-ordinate system.

Figure 4.11: The image-space likelihood when varying a subset of the scene's transformation parameters while the others are kept at truth.



(a) The likelihood as the shared principal-point is adjusted. The image-plane is bound by the lines $x = \pm 1$ and $y = \pm 1$



(b) The likelihood as the shared focal length is adjusted.

Figure 4.12: The image-space likelihood when varying the camera parameters while the surface is kept at truth.

elsewhere in the parameter space. The likelihood is maximised in these cases because the image-space likelihood (4.16) relies on the occlusion likelihood (4.13), and therefore cannot distinguish between photo-consistent and occluded pixels. The impact of occlusion on the image-space likelihood is described in Section 4.6.

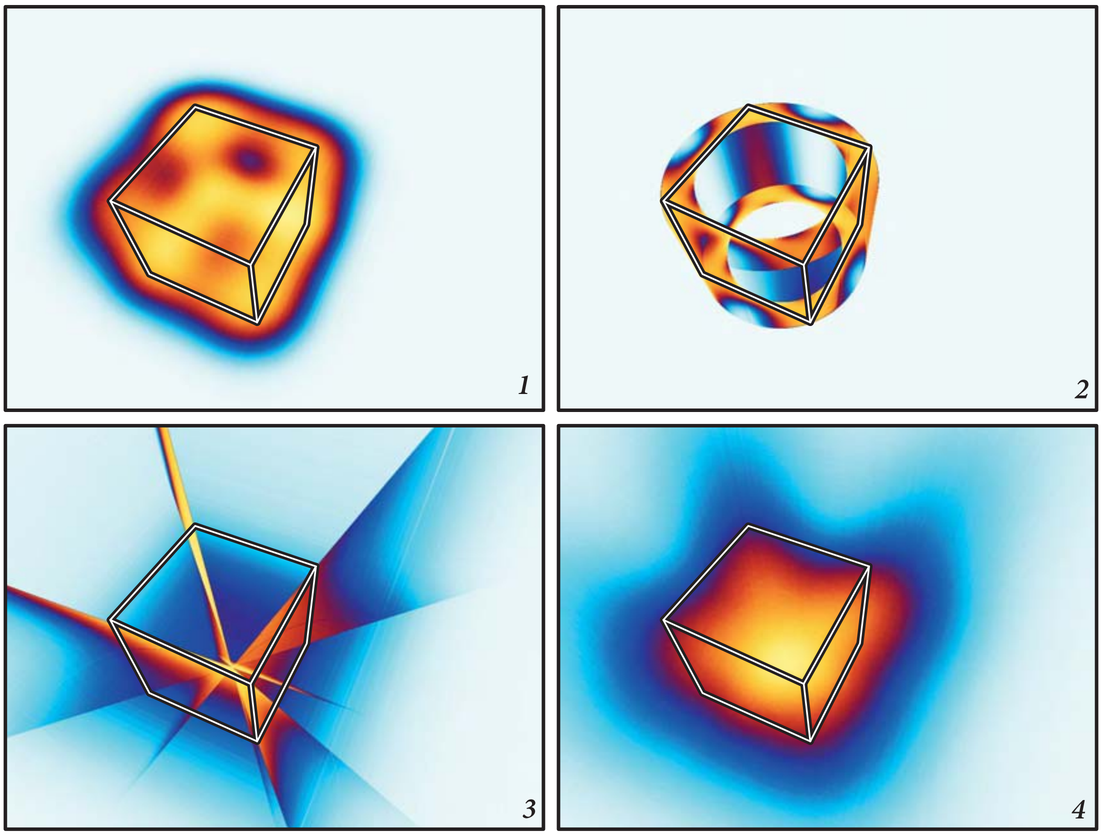


Figure 4.13: An illustration of the surface priors generated by overlaying random samples for 1) translation, 2) rotation, 3) scale, and 4) joint distribution of all three priors.

4.5.1 *Noise testing*

The effect of image noise on the reconstruction was tested by monitoring the MAP estimate of the posterior (3.4) after adding random Gaussian noise to the reference images. Priors on the cube's parameters were defined to be as uninformative as possible while omitting the incorrect solutions identified in Figure 4.11(c). The scene-parameter priors used for this test were defined as:

- the translation prior: $\Pr(\boldsymbol{t}) = \mathcal{U}(-0.25, 2.75)$;
- the rotation prior (in degrees): $\Pr(\theta) = \mathcal{U}(-55.0, 35.0)$; and
- the scale prior: $\Pr(s) = \mathcal{U}(0.5, 2.0)$,

where $\mathcal{U}(a, b)$ is the uniform distribution in the range $[a, b]$. The scale of these priors is relative to the cube's synthetic dimensions of $2 \times 2 \times 2$ world units.

Figure 4.13 is a visualisation of the variability represented by these priors. These images were generated by randomly sampling 100,000 surface configurations from each of the respective prior distributions and rendering a wire-frame model of the surface. The wire-frame images were accumulated to effectively generate a per-pixel histogram representing the frequency of an edge projecting to a given pixel, where the pixel colour corresponds to increasing frequency from pale blue to red. The aim of these images is to illustrate that while the priors restrict the set of hypothesised cubed, the priors allow sufficient variability to make the test meaningful.

We ran a series of simulated annealed Monte Carlo Markov Chains (MCMC) [38] to estimate the cube's position relative to the world co-ordinate frame. The MCMC algorithm is a method for sampling a probability distribution by randomly walking through parameter space. A MCMC chain is defined by a starting point in the distribution and a set of *jumping* distributions. The jumping distributions, $J(\eta_{i+1} \mid \eta_i)$, define the probability of choosing state η_{i+1} given the current state η_i.

New states are drawn from the jumping distributions and are added to the chain according to an acceptance test. This test is based on the ratio between the probability of the proposed state and the probability of the last accepted state: a proposed state with a higher probability than the current state is always accepted, but a proposal that is less probable is randomly rejected. Accepting new states is a random 'walk' that mimics a series of fair samples drawn from the probability distribution.

Drawing a set of samples from the posterior (3.4) is one method of characterising the distribution, but we are usually only interested in the posterior's maximum. To this end, the jumping distributions and acceptance test is 'cooled' over time, such that the chain converges to the global maximum. The cooling schedule in this case was defined by

$$T_i = \frac{1}{2}(1 - 10^{-3})\left(1 + \cos\left(\frac{i\pi}{n}\right) + 1\right) \tag{4.18}$$

where T_i is the temperature at the i^{th} iteration and n is the length of the Markov chain. Gaussian distributions, whose standard deviation cools over time and is initially half the range of the respective priors, were used as jumping distributions:

- $J(\boldsymbol{t}_{i+1} \mid \boldsymbol{t}_i) = \mathcal{N}(\boldsymbol{t}_i, 0.75^{T_i})$;
- $J(\theta_{i+1} \mid \theta_i) = \mathcal{N}(\theta_i, 22.5^{T_i})$; and
- $J(s_{i+1} \mid s_i) = \mathcal{N}(s_i, 0.375^{T_i})$,

where $\mathcal{N}(\mu, \sigma)$ is the normal distribution with mean μ and standard deviation σ.

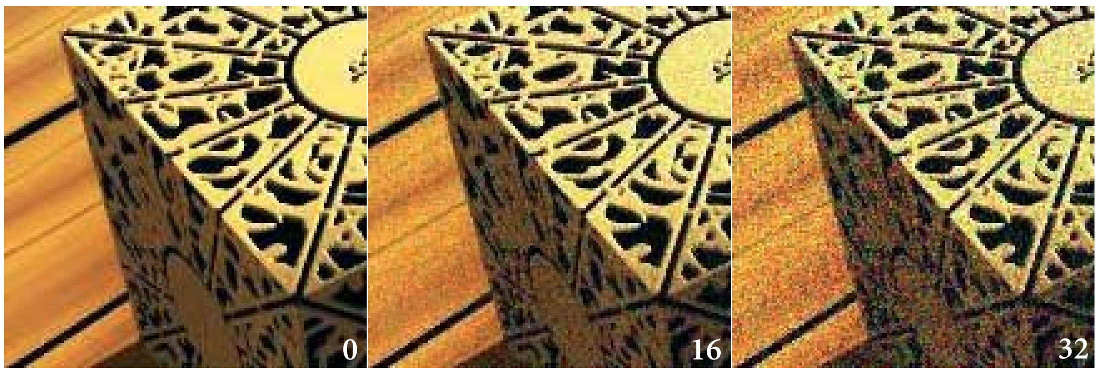


Figure 4.14: The effect of additive Gaussian noise on an 24-bit RGB image with $\mu = 0$ and σ at 0, 16 and 32.

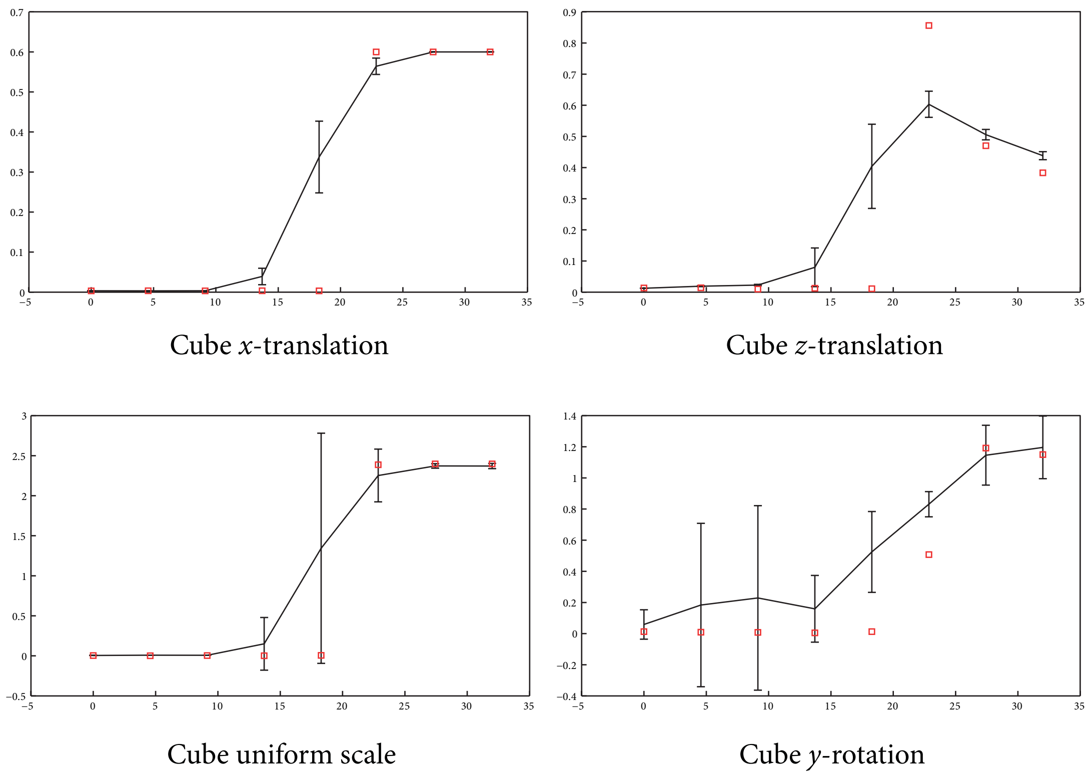


Figure 4.15: The average relative scene parameter variance over 50 iterations for increasing levels of image noise. Vertical bars indicate variance and red markers denote the MAP estimate for a given noise level.

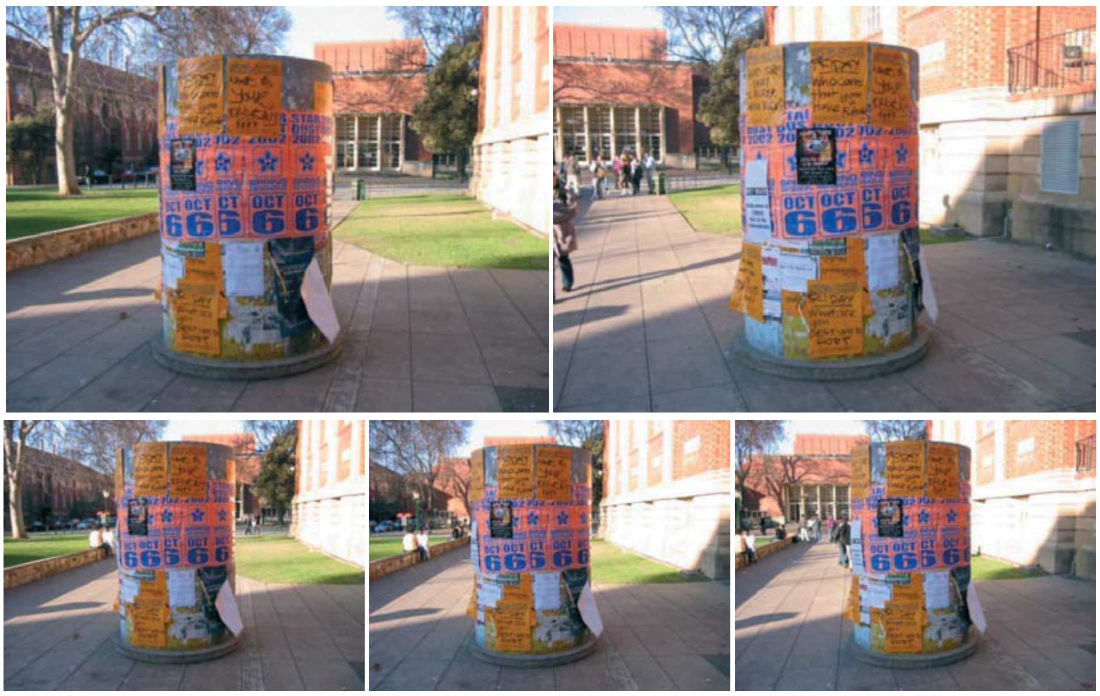


Figure 4.16: The reference images of the the poster cylinder sequence.

Eight tests were conducted against increasing levels of noise applied to 24-bit RGB images. The level of noise ranged from $\sigma = 0$ to $\sigma = 32$; Figure 4.14 illustrates the additive Gaussian noise with $\mu = 0$ and $\sigma = 0$, $\sigma = 16$, and $\sigma = 32$ used in three of these tests. Fifty chains over $25,000$ states were generated for each test. The relative error is graphed against increasing noise in Figure 4.5.1. This graph plots the average relative error of the MAP estimates for each trial; the vertical bar denotes variance. The red marker indicates the best estimate of the 50 trials per noise level. The results demonstrate that the chains converge near truth for trials up to $\sigma = 16$. Beyond $\sigma = 16$, however, the chain converges to the limits of the respective prior distribution, indicating that the consistency likelihood is unstable at excessive noise levels.

4.5.2 University poster sequence

The image-space likelihood was tested on the image sequence taken by a digital camera; the five reference images are illustrated in Figure 4.16. The cameras were calibrated with respect to the ground-plane using the Icarus camera tracker [21]. Although the reference images contain many scene-elements, including people moving between

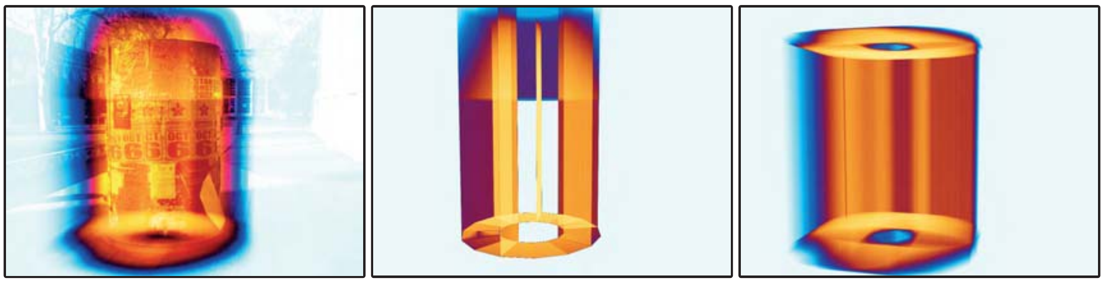


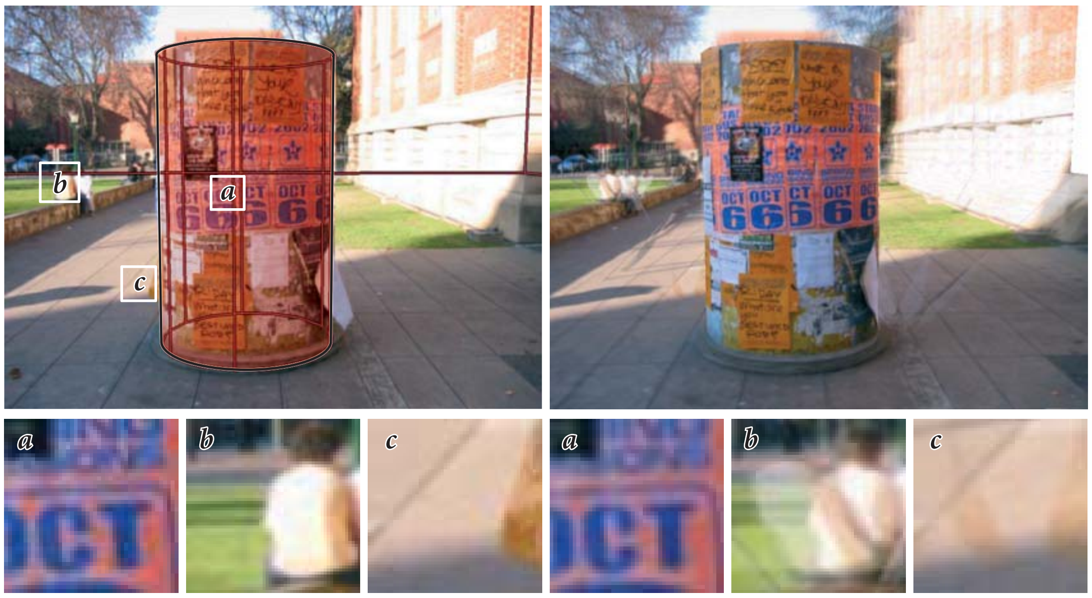
Figure 4.17: The translation and scale priors for the poster sequence projected using the first reference camera. The translation and scale prior visualisation is blended with the corresponding reference image for illustration.

frames, the objective of this experiment was to estimate the geometry of the 'poster cylinder' relative to the calibrated cameras.

A scene-graph consisting of a transformable cylinder, static ground-plane, and back-drop was used for this experiment. This scene-graph required four parameters: two parameters to translate the cylinder relative to the ground-plane, and two parameters for the radius and height of the cylinder. Priors for these parameters were chosen by experimenting with different distributions and examining the prior visualisation images. The priors used in the experiment are visualised in Figure 4.17.

Although reconstructing the cylinder was the focus of this experiment, the background elements were required so that the posterior is well conditioned. Because the scene-graph is used to map hypothesised correspondences between reference images, the image-space likelihood critically relies on the projection of the hypothesised surface in each reference image. Accordingly, the maximum likelihood is given by minimising the hypothesised surface's projection in each view. The purpose of the background elements is therefore to ensure that the hypothesised surface maintains a relatively static footprint in each reference image so that the likelihood is only maximised by minimising the photo-consistency error-term.

The back-drop was defined to be a vertical plane, placed approximately parallel to the set of reference views. It was manually positioned by experimenting with different values and visualising the re-projection of the back-drop. Only the scene-graph ground-plane was accurately measured with respect to the reference cameras by virtue of the camera calibration. The scale of the ground-plane quadrilateral and



(a) The wire-frame projection of the hypothesised surface super-imposed with the reference image. **(b)** Re-projection of all reference images back-projected onto the surface.

Figure 4.18: Re-projection of the university poster sequence using one of the reference cameras, including enlargements of three regions of interest. The silhouette of the hypothesised surface is shaded in the wire-frame image.

the position, orientation, and scale of the back-drop plane were manually estimated by experimenting with the scene-graph's wire-frame projection in each reference image. The static back-drop geometry was known to be incorrectly matched against the reference images. The building on the right side was not modelled, for example, and the ground-plane extended beyond the stone fence in the left of the reference images and into building on the right. The distance and orientation of the back-drop was known to not correspond with the building in the distance.

A simulated annealed MCMC chain converged to the MAP estimate illustrated in Figures 4.18, 4.19 and 4.20. Figure 4.18(a) is the composite of the reference image with the wire-frame projection of the hypothesised surface, illustrating that the edges of the hypothesised cylinder are correctly aligned with the edges of the cylinder in the reference image. Figure 4.18(b) is generated by back-projecting each reference image onto the hypothesised surface. This process is illustrated in more detail in Figure 4.19, where each of the five reference images is back-projected onto the hypothesised surface

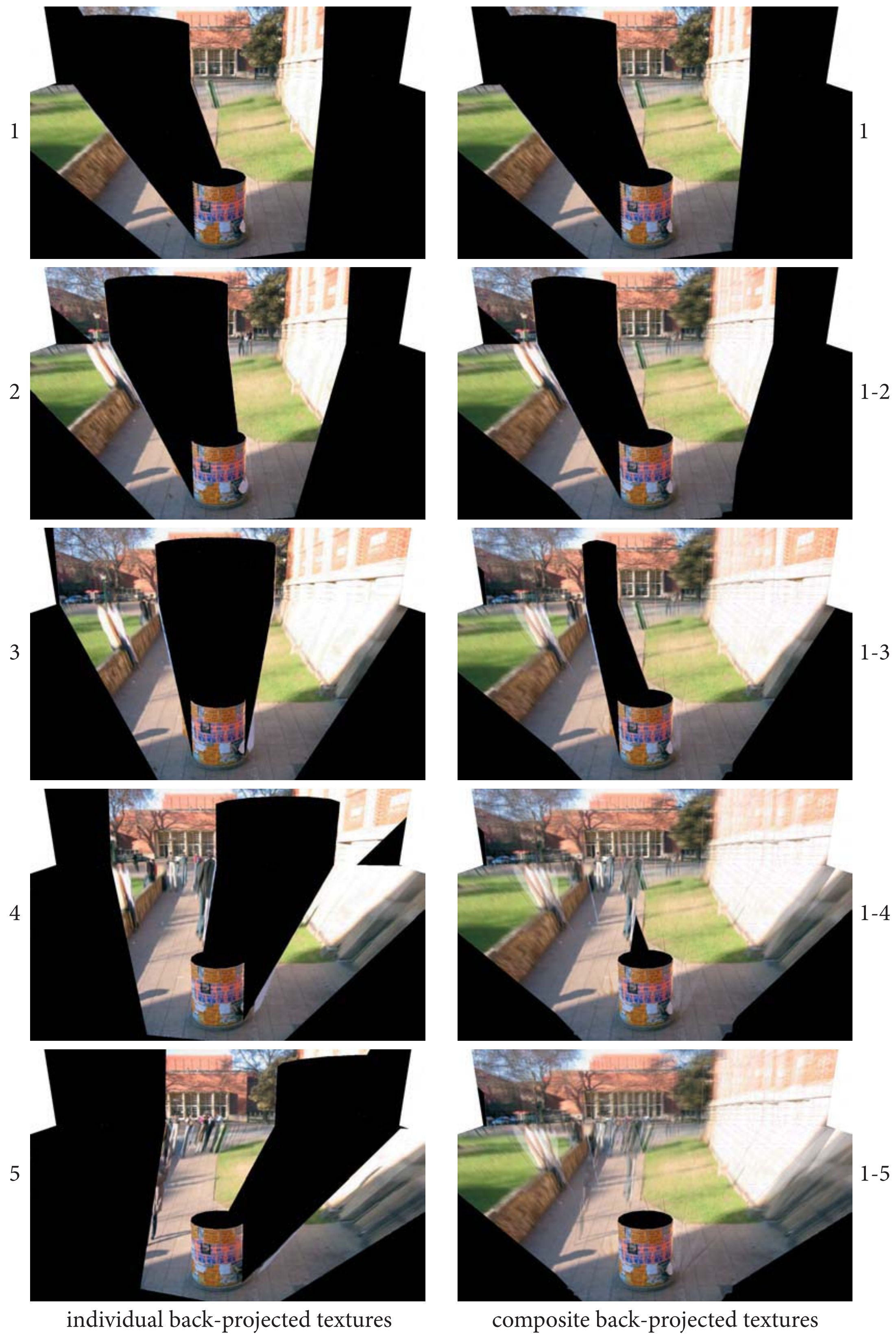

1
1
2
1-2
3
1-3
4
1-4
5
1-5
individual back-projected textures
composite back-projected textures

Figure 4.19: Back-projecting the reference images onto the reconstructed surface (the left column) and the composite results (the right column).

NOTE: This figure is included on page 72 of the print copy of the thesis held in the University of Adelaide Library.

Figure 4.20: Novel views of the poster-cylinder reconstruction (1).

and the result imaged by a novel virtual camera. The left column is the back-projection of a single reference image; the cumulative image is given in the right column.

The image illustrated in Figure 4.18(b) is generated from the average back projection of the five reference images onto the reference image, while the surface is imaged using the same reference camera as Figure 4.18(a). Three regions of interested are enlarged for comparison between one reference image (Figure 4.18(a)) and the average re-projection (Figure 4.18(b)):

- **Frame *(a)*** corresponds to the cylinder, which is part of the model optimised by the image-space likelihood. The synthetic image is a close facsimile of reference image, and has less noise than the reference image because the corresponding image patch is averaged across the five reference images.
- **Frame *(b)*** is blurred in the synthetic image because the person was not modelled in the scene-graph.
- **Frame *(c)*** contains artifacts in the synthetic image because the hypothesised surface does not correctly model the reference images. In this case, the edge of a poster extends beyond the hypothesised cylinder's surface; the poster is back-projected onto the ground-plane because the scene-graph assumes a perfect cylinder. As the correspondence from the poster's edge with the ground-plane depends on the reference image, the edge of the poster is blurred over a relatively large area of the ground.

The image-space likelihood succeeds despite the hypothesised scene-graph incorrectly modelling the reference images. The correct alignment of the map surface

NOTE: This figure is included on page 73 of the print copy of the thesis held in the University of Adelaide Library.

Figure 4.21: Novel view of the poster-cylinder reconstruction (2).

dominated the hypothesised projection in each reference image mitigating the inconsistency caused by poster edges and the incorrectly matched, static, back-drop. Novel views of the map surface in Figures 4.20 and 4.21 illustrate the photo-inconsistency inherent in the hypothesised scene, where scene-elements that are not part of the poster cylinder or ground-plane are 'smeared' across the ground.

4.5.3 Cardboard box sequence

A sequence, captured by a digital camera, of a cardboard box on a concrete ground-plane was used to repeat the synthetic experiment described in Section 4.5.1. The four images from the sequence are illustrated in Figure 4.23. The images were calibrated using Tsai's method of camera calibration [67], which involved correlating a set of known 3D points with their projection in each reference image. The point cloud's projection was manually identified in each reference image. Although the physical box is moderately distorted, a 3D point cloud representing points on the box's surface was defined under the assumption that the cardboard box was a perfect cube.

This experiment used a scene-graph which comprised a cube and ground-plane. The scene was parameterised by the cube's 2D position $[t_x , t_z]^T$ over the ground-plane, the cube's rotation θ (in degrees) about its vertical axis, and the cube's scale s. An initial

Parameter	Prior	Calibration	Estimated
$\boldsymbol{t}_x$	$\mathcal{U}(-1, 1)$	0	-1.406×10^{-5}
$\boldsymbol{t}_z$	$\mathcal{U}(-1, 1)$	0	6.729×10^{-4}
θ	$\mathcal{U}(-45, 45)$	0	0.142
s	$\mathcal{U}(0.5, 1.5)$	1	1.006

Figure 4.22: The priors probabilities, the parameter values corresponding to the calibration cube, and the MAP estimate for the cardboard box test.

estimate of the cube's transformation parameters is given by $t_x = 0$, $t_z = 0$, $\theta = 0$, and $s = 1$, which corresponds to the 3D point model that was used to calibrate the cameras.

Verifying that the posterior is able to recover the cube's transformation parameters was the first stage of this experiment. A simulated annealed MCMC chain of 5000 states was initialised from the surface parameters that were used to calibrate the cameras. The prior[2] probabilities and the MAP estimate for this experiment is given in Figure 4.22. The re-projection of the wire-frame model over the reference image (Figure 4.24(a)) indicates that the model's edges are strongly correlated with the cardboard box's edges in the reference images. The MAP estimate, with *all* reference images back-projected onto the surface, is imaged by the reference cameras in Figure 4.24(b) and by novel cameras in Figure 4.25. The images of the textured surface strongly accord with the reference images, and novel views of the MAP surface indicate that the box's texture has been successfully segmented from the ground-plane texture.

The box's checkered pattern has been successfully registered over most of the surface, but is unfocused on the side illustrated in the third frame on the top row of Figure 4.25. This defocus is caused by parallax error between the hypothesised surface, which is assumed to be planar, and the cardboard box's concave surface. The discord between the hypothesised model and the surface in the reference images is emphasised when all reference images are back-projected onto the surface, where the average colour of coincident points from different views is given. Despite this photo-inconsistency, the posterior has successfully recovered a model that best recreates the reference images.

[2]The cube primitive is defined by the vertices $[\pm 1, 1 \pm 1, \pm 1]$, and therefore the translation prior allows sufficient freedom to translate the centroid of the hypothesised cube within the footprint of the calibration model.

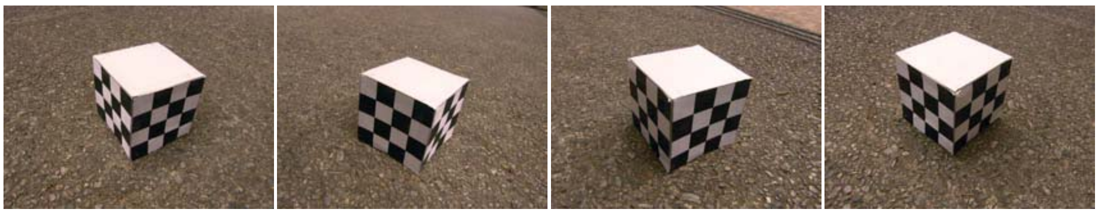
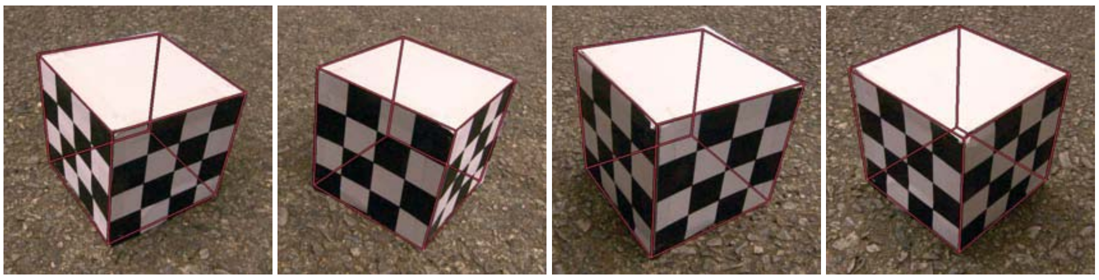
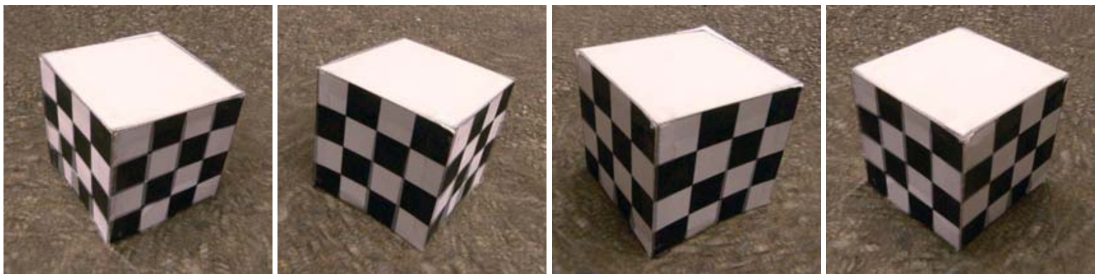


Figure 4.23: Reference images for the cardboard cube sequence.



(a) Wire-frame re-projection of the MAP surface over the reference images.



(b) All reference images back-projected onto the MAP surface.

Figure 4.24: Re-projection of the MAP surface using the reference cameras.

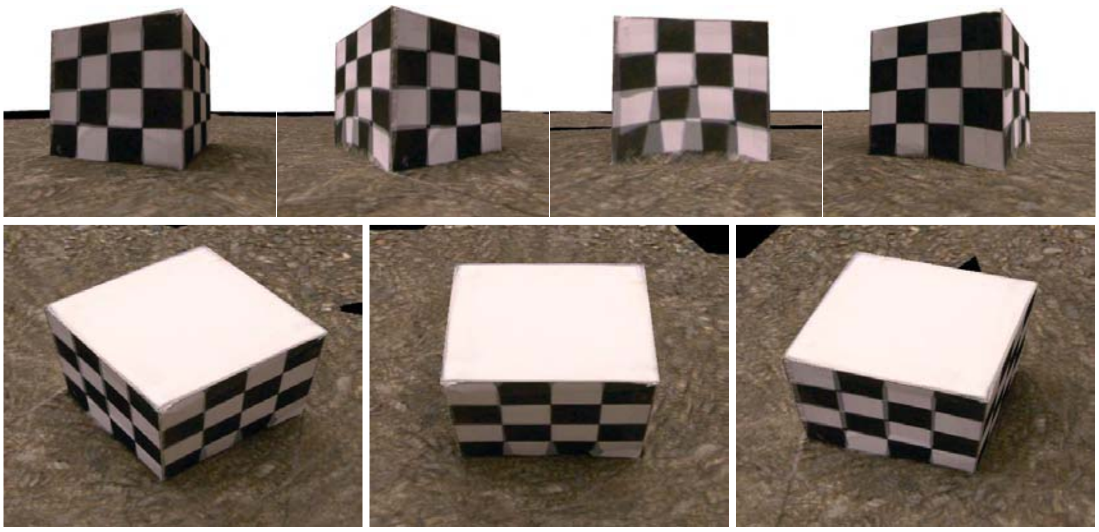


Figure 4.25: Novel views of the cardboard box reconstruction.

Rotation prior	**Translation prior**	**Relative likelihood**	**Re-projection error**
$\mathcal{U}(-15, 15)$	$\mathcal{U}(-1.125, 1.125)$	2.800	351.5180
$\mathcal{U}(-5, 5)$	$\mathcal{U}(-0.75, 0.75)$	1.020	20.8774
(initial solution)		1	0 *(see note)*

Figure 4.26: The relative log likelihoods of the MAP estimate for different priors. The re-projection error is given in pixels when assuming a 1024×768 resolution image. *Note:* The re-projection error is measured with respect to the projection of the Tsai-estimated solution. The re-projection error between the 3D point-cloud and manually identified correspondences used for calibration was approximately 3.5 pixels.

Optimising camera parameters

Having demonstrated that the cardboard box model can be recovered from calibrated cameras, the second part of the experiment aimed to recover the camera parameters while holding the scene-graph cube to the 'calibration model' parameters. The aim of this experiment, then, was to estimate the cameras relative to the box, rather than estimate the box relative to the calibrated cameras.

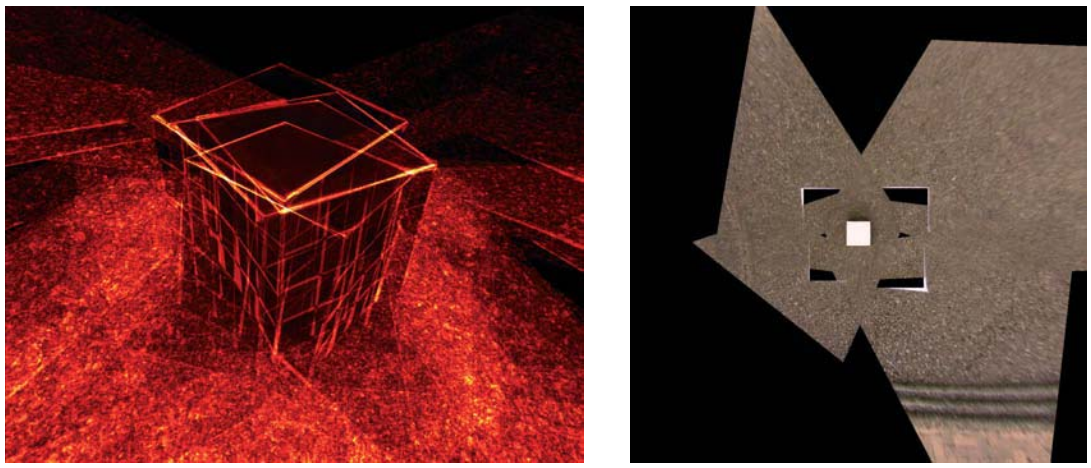
The test configuration allowed the cameras to change their intrinsic and extrinsic parameters relative to the parameters estimated by the Tsai calibration. In all tests that follow, the following priors on the focal length f and principal point $[o_x, o_y]^\top$ were used:

- $\Pr(f) = \mathcal{U}(0.85, 1.15)$ as a multiplier of the initial focal length; and
- $\Pr(\boldsymbol{o}) = \mathcal{U}(-0.01, 0.01)$ as an offset to the initial principal point.

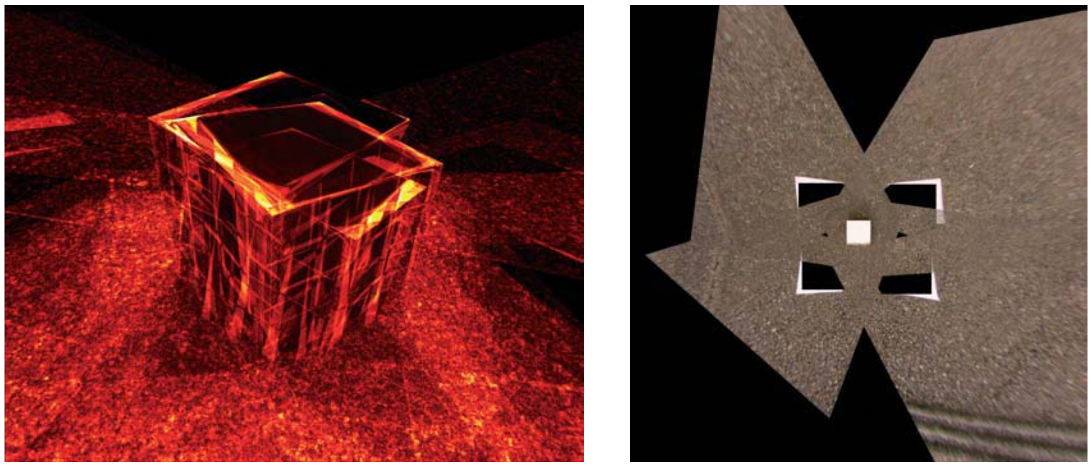
The priors on the extrinsics were described by rotation $\theta = [\theta_x, \theta_y, \text{ and } \theta_z]^\top$ and translation vector $\boldsymbol{t}$ relative to the co-ordinate frame of the calibrated cameras. The results of two particular trials are included here, using the priors

- $\Pr(\boldsymbol{t}) = \mathcal{U}(-1.25, 1.25)$ and $\Pr(\theta) = \mathcal{U}(-15, 15)$ in the first trial; and
- $\Pr(\boldsymbol{t}) = \mathcal{U}(-0.75, 0.75)$ and $\Pr(\theta) = \mathcal{U}(-5, 5)$ in the second.

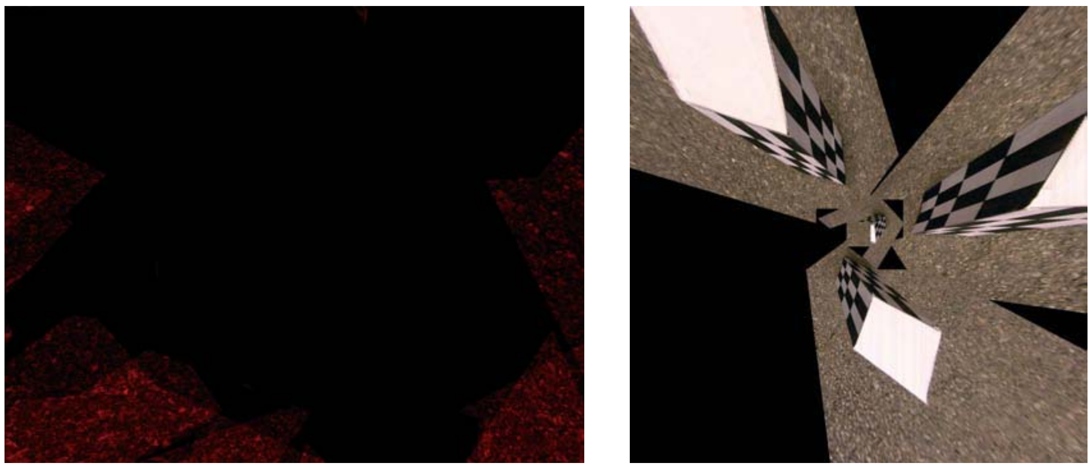
For each of these two trials, a number of simulated annealed MCMC chains of $15,000$ states were generated from the initial, calibrated solution. Unlike the previous experiment, no MAP estimate converged to a solution where the edges of the hypothesised scene-graph aligned with the edges in the reference images. The results from the two trials compared with the initial solution is tabled in Figure 4.26. The re-projection error given in the table by the ℓ^2-norm between the projection of the cube's vertices from the initial configuration and each MAP estimate.



(a) Initial solution from the Tsai calibration.



(b) $\theta \sim \mathcal{U}(-5, 5)$, $\boldsymbol{t} \sim \mathcal{U}(-0.75, 0.75)$



(c) $\theta \sim \mathcal{U}(-15, 15)$, $\boldsymbol{t} \sim \mathcal{U}(-1.125, 1.125)$

Figure 4.27: The initial Tsai estimate and two MAP estimates with different priors for the camera extrinsics. The left column indicates accumulated photo-inconsistency and the right column illustrates the reference images back-projected onto the hypothesised scene.

These results show that the likelihood increases as the priors on the camera extrinsics are relaxed, but the re-projection error also increases. This problem is caused by the image-space likelihood's bias toward occlusion. Because the likelihood is based on the comparison of hypothesised correspondences, the likelihood is maximised when the surface is occluded in all views. This bias is illustrated in Figure 4.27, where the left column is the 'consistency image' generated by the accumulated inconsistency from stereo comparisons.

The right column in Figure 4.27 is an image of the MAP estimate as the reference images are back-projected onto the surface. This figure illustrates that as the hypothesised cameras are relaxed, the image-space likelihood arranges the cameras to minimise the surface area which is visible across the set of cameras. The scene can be occluded in a number of ways, including increasing the focal length, translating the cameras closer to the ground plane, and rotating their optical axis to minimise the surface area visible in all stereo pairs. This bias is evident by comparing the relatively low photo-consistency error in Figure 4.27(c) with the consistency image from the initial solution 4.27(a).

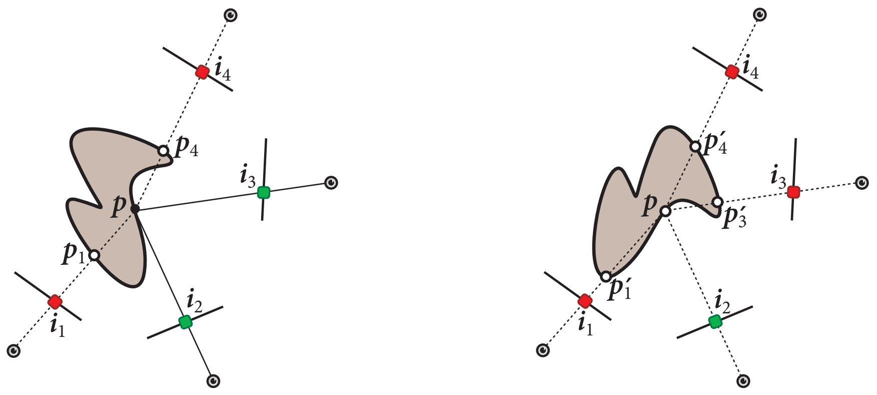
4.6 Limitations of image-space occlusion

The image-space approach defines a posterior distribution over the scene-graph's parameter space by measuring the likelihood of generating the reference images given the hypothesised surface. This score is inversely proportional to the surface's observed colour variance and therefore critically depends on the surface's visibility. Because rendering involves projection with occlusion, correlation measurements are only performed on the *closest* hypothesised surface points to each reference image. Only cameras 2 and 3 from Figure 4.28(a), for example, can determine the consistency of the point $\boldsymbol{p}_1$ because it is occluded in cameras 3 and 4. Although the dissimilarity of point's projection is a good indication that it is not on the reference surface, a similar colour *does not* suggest that the occlusion hypothesis is incorrect.

The problem caused by occlusion leads to two ways of maximising the surface's likelihood, by either:

- finding a visible solution that minimises colour variance; or
- minimising visibility so that inconsistency is also minimised.

The likelihood function is biased toward occlusion because the error term for visible points is greater than the error term for occluded pixels. Even when the



(**a**) $\boldsymbol{p}$'s consistency can be determined (**b**) $\boldsymbol{p}$'s consistency cannot be determined.

Figure 4.28: A surface point must be visible in multiple views in order to determine its photo-consistency.

hypothesised surface represents the true scene, a point's photo-consistency is likely to be non-zero due to filtering and sensor artifacts. The four pixels $\boldsymbol{i}_1, \cdots, \boldsymbol{i}_4$ illustrated in Figure 4.28(a), for example, would be considered less likely than the same points from the transformed surface illustrated Figure 4.28(b). The transformation is likely to affect the correlation of other pixels in the reference image because the surface was fundamentally modified to occlude $\boldsymbol{p}$. In this case, the transformation *may* yield a less photo-consistent solution over the entire surface, despite improving the likelihood of $\boldsymbol{p}$.

4.6.1 Image-space visibility constraints

The image-space likelihood readily finds trivial solutions because it is unable to identify low variance due to significant occlusion. Could a visibility metric encourage photo-consistent but visible scenes by penalising occlusion? An occlusion likelihood might be applied by modifying (4.13) to

$$\mathscr{L}(\alpha(\boldsymbol{p}) = 0) = \kappa \tag{4.19}$$

where $\kappa < (\sigma\sqrt{2\pi})^{-1}$ is one way of suggesting that occlusion is less likely than zero inconsistency. Alternatively, an occlusion likelihood could be a function of the total number of occluded pixels in a given stereo pair. The image-space likelihood (4.14)

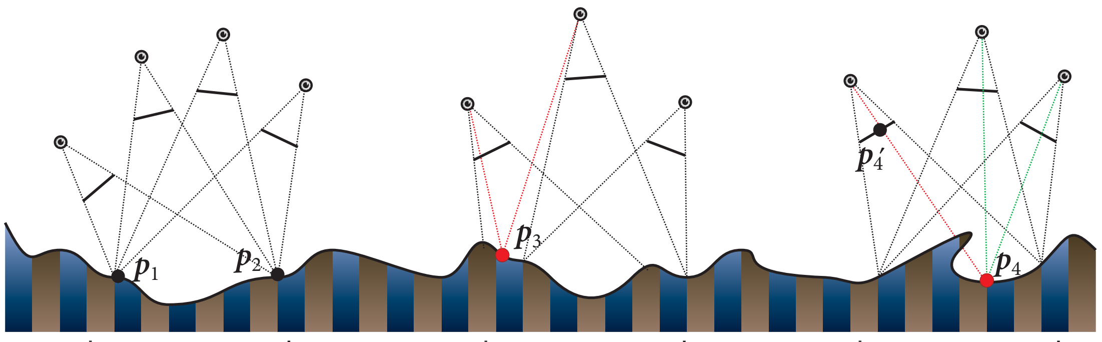


Figure 4.29: Three classes of scene visibility ranging from total visibility, partial visibility and total occlusion.

would become, in this case,

$$\mathscr{L}(\mathsf{r}_t, \alpha \mid \mathsf{r}_s, \Psi) = \mathscr{L}(\alpha \mid \mathsf{r}_s) \prod_{\boldsymbol{p}} \mathscr{L}(\mathsf{r}_t(\boldsymbol{p}) \mid \mathsf{r}_s, \Psi). \tag{4.20}$$

where

$$\mathscr{L}(\alpha \mid \mathsf{r}_s) = \begin{cases} 1 & \text{if } \sum_{\boldsymbol{p}} \alpha(\boldsymbol{p}) < \tau \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

Choosing appropriate κ and τ constants is the key difficulty with both occlusion likelihoods because they require prior knowledge about the reference surface's visibility. A per-pixel occlusion likelihood would be applicable only if the surface spanned the entire image for all views. An example of such a configuration is illustrated in Figure 4.29 where all visible scene-points between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are unoccluded and within the four reference camera frustums. The camera's position and focal length must be carefully controlled to ensure constant visibility, making this configuration exceptionally limited. In more general cases, penalising occlusion would adversely affect the reconstruction because a surface point that is visible in one camera is unlikely to be visible in *all* cameras.

The image-space algorithm is not able to correlate surface points across the entire set of reference images because it is only capable of dealing with *stereo* occlusion. The point $\boldsymbol{p}_3$, for example, have an occlusion penalty in Figure 4.29 because it is not within all three of the configuration's cameras. Similarly, $\boldsymbol{p}_4'$ would be penalised because its corresponding point $\boldsymbol{p}_4$ is occluded. The likelihood model 4.16 would not be aware

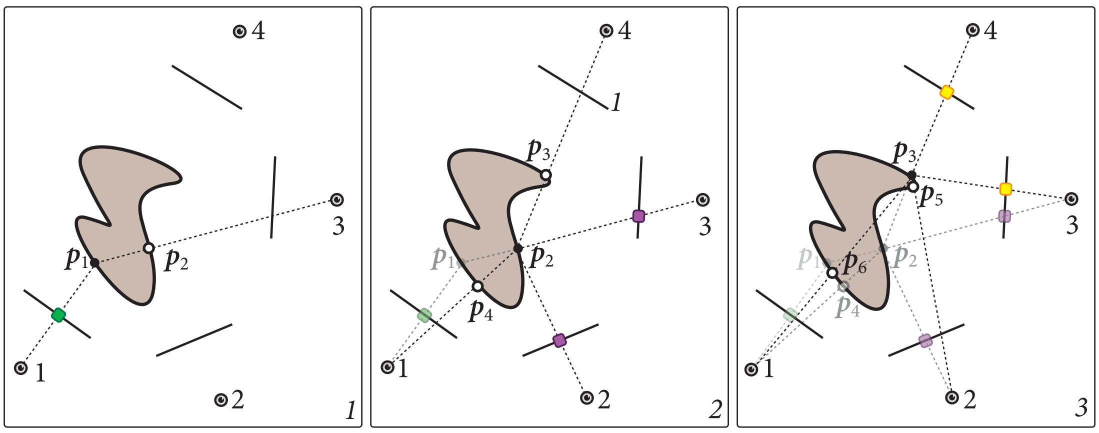


Figure 4.30: A conditional visibility chain over three iterations.

that $\boldsymbol{p}_4$ is visible in the two other reference views and therefore a conditional occlusion penalty cannot be applied. These occlusion penalties would cause cause the likelihood function to tolerate photo inconsistency in order to maximise visibility, thereby leading to undesirable reconstructions.

4.6.2 Toward a surface-space occlusion likelihood

Could the likelihood of the occlusion scene-point be used to define the likelihood of a pixel being marked as occluded? Recasting a pixel's likelihood as a function that involves other surface points in the scene would would create a recursive chain of dependencies because each occlusion would require determining the validity of ever-increasing numbers of surface points. An example of this recursive dependency chain is illustrated in Figure 4.30. Here, the point $\boldsymbol{p}_1$ is occluded by the point $\boldsymbol{p}_2$ in the third reference view. Determining whether $\boldsymbol{p}_2$ is on the hypothesised surface is a function of its photo-consistency in the second and third views and the likelihood that it is occluded in the first and fourth views by the points $\boldsymbol{p}_4$ and $\boldsymbol{p}_3$ respectively. Similarly, the occlusion likelihood for $\boldsymbol{p}_3$ requires determining the surface likelihood of points $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$ in the first and second images respectively.

Adopting an occlusion likelihood that considers the probability of hypothesised surface points would requires traversing the entire scene. This recursion is limited only if the scene can be discritised into a finite set of points. As described in Chapter 2, iteratively updating surface probabilities has been considered by Debonet

[6] and Agrawal [1] to reconstruct the surface. In contrast, a similar approach in this circumstance would contribute to the likelihood of $\mathcal{S}(\Psi)$ rather than as an approach to estimate Ψ directly. Furthermore, including a likelihood metric would involve *directly* representing the surface rather than its implicit form by considering only its projection and is therefore not amendable to the image-space likelihood metric.

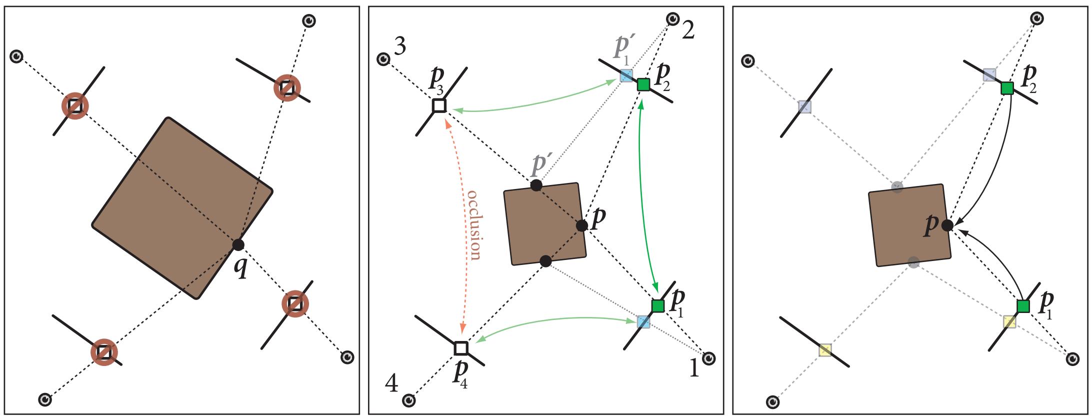
Towards a Surface Occlusion Metric

Chapter V

The image-space likelihood described in Chapter 4 is adversely affected by occlusion. Although this likelihood is able to distinguish between consistent and inconsistent points, it cannot distinguish between a surface that is photo-consistent and one that has limited photo inconsistency caused by occlusion. This motivates a new likelihood which includes an occlusion prior which guarantees that the surface is sufficiently visible so that photo-consistency can be measured.

The image-space likelihood cannot impose visibility constraints because hypothesised correspondences cannot be correlated across the set of stereo pairs. The problem of stereo-occlusion is illustrated in Figure 5.1. Figure 5.1(a) is an example of a undesirable solution where *all* correspondences are occluded. Stereo occlusion cannot be used to reject a shape hypothesis, however: the surface in Figure 5.1(b), for example, yields a set of image-space likelihoods that includes both photo-consistency measurements and occlusion. Unfortunately, the occlusion likelihood must be determined independently of the back-projected surface point's consistency in any other stereo pair. Hypothesised correspondences cannot be correlated because the result of pair-wise comparisons is stored in image-space of the target image, and is therefore a function of the surface's projection by the particular camera.

The surface's consistency must be measured in a space which is independent of the surface's projection so that a point's projection in each reference image can be considered as a set, rather than a series of pair-wise comparisons. This shift in measurement space would allow the use of an occlusion prior to guarantee the surface was sufficiently visible. In the example illustrated in Figure 5.1, a surface-space likelihood could apply a penalty to the point $\boldsymbol{q}$ from Figure 5.1(a) because its visibility set is empty.



(a) Surface occlusion. **(b)** Image-space likelihood. **(c)** Surface-space likelihood.

Figure 5.1: The image-space likelihood can only consider stereo occlusion and is unable to track visibility across different pairs. In contrast, the surface-space likelihood is able to monitor the number of views of each surface point.

5.1 Surface-based visibility constraints

Representing consistency in surface-space requires an explicit model of the surface texture, rather than an implicit representation used by the image-space likelihood. To this end, three texture images are associated with each scene-graph primitive. The first texture represents the surface's estimated reflectance; the second is a 'visibility map' which denotes the number of images in which a texel is visible; and the third is a 'consistency' image which represents the per-texel surface-space likelihood. Given these three texture spaces, the surface-likelihood is a problem of estimating the surface texture given $\mathcal{S}(\Psi)$ and deriving a likelihood using the surface's photo-consistency and visibility metrics.

A *texture map* is used to relate a triangles in the texture image with triangles on the surface. The map from surface to texture space is used in conjunction with the surface's projection by a particular camera to define the projective relationship between surface-space and the particular reference image. A *view-specific* surface texture is generated with respect to the particular camera by mapping the projection of each surface triangle from the reference image into the surface's texture space. An example of a view-specific texture is illustrated in Figure 5.2.

View-specific textures *exactly* recreate the corresponding reference image when

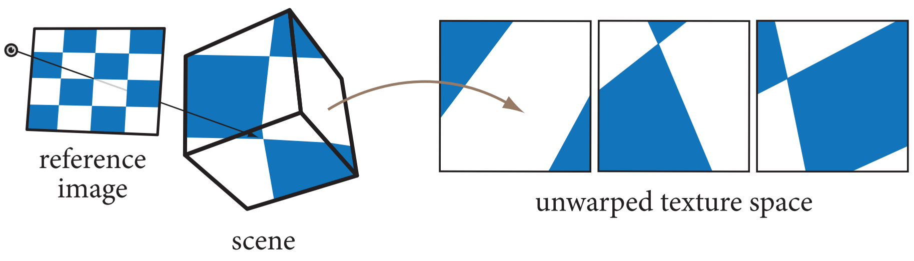


Figure 5.2: View-specific textures are generated by back-projecting the reference image onto the surface and mapping the result to the surface's texture space.

the texture is applied to the hypothesised surface and rendered with the same reference camera parameters. This property holds irrespective of whether the hypothesised surface is correct, but a texture from one reference image is *unlikely* to recreate all reference images unless the hypothesised surface is correct. This property is exploited to derive a surface-space photo-consistency metric.

The surface photo-consistency is measured by comparing each view-specific texture against the average view-specific texture. Inconsistencies between view-specific textures and the average texture suggests that the scene parameters are incorrect. This photo-consistency metric, described by the variance of the view-specific textures, is used to drive an optimisation process which finds scene-parameter values that maximise consistency with respect to all images. Figure 5.3 illustrates the process of 'unfolding' the reference images to create a set of view-specific textures that are compared to yield a photo-consistency metric.

5.2 Generating view-specific textures

A texture image is used to represent the surface's estimated reflectance. This image is mapped to the surface via a set of texture co-ordinates known collectively as a texture map; two example texture maps for a cube are illustrated in Figure 5.4. The texture map is defined by a set of correspondences between the primitive's triangles in $\mathbb{R}^3$ and the primitive's 'texture triangles' in texture space. These correspondences therefore define a set of piece-wise, affine homographies defining the relationship between geometry and surface-space.

The texture map is used to distort a given reference image into texture-space

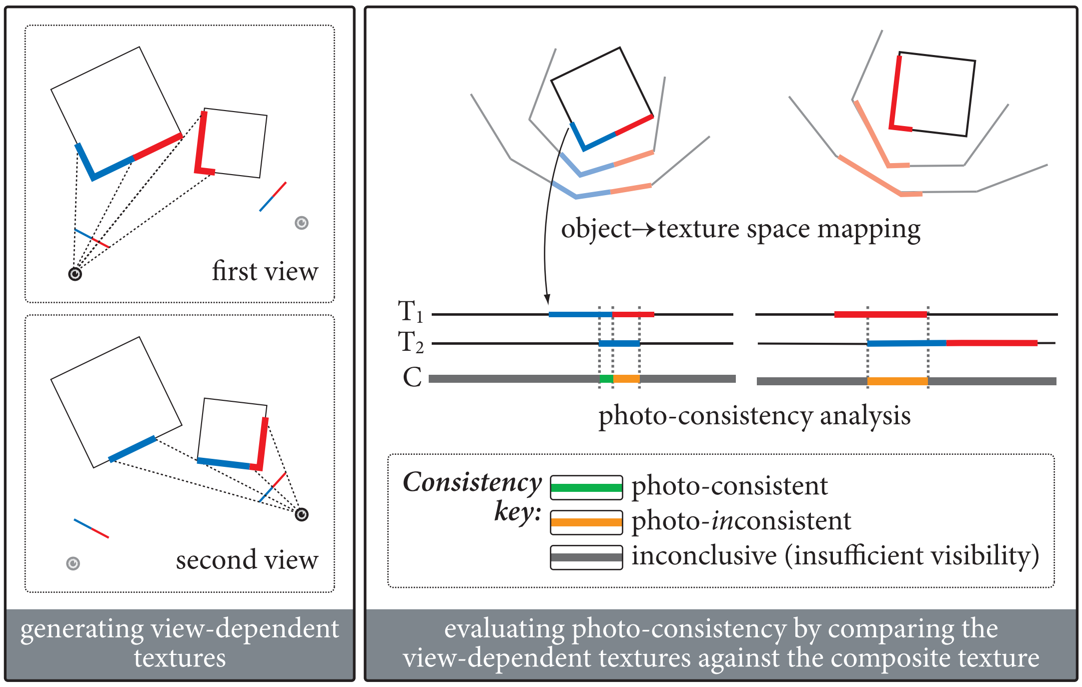

first view
second view
generating view-dependent textures
object→texture space mapping
T1
T2
C
photo-consistency analysis
Consistency key:
photo-consistent
photo-inconsistent
inconclusive (insufficient visibility)
evaluating photo-consistency by comparing the view-dependent textures against the composite texture

Figure 5.3: Photo-consistency is determined by constructing a set of view-specific textures and comparing them against the 'composite' texture.

via the surface's projection. Suppose the matrices $\hat{\boldsymbol{K}}$ and $\boldsymbol{E}$ are the projection and extrinsics matrices respectively for a particular reference camera, where $\hat{\boldsymbol{K}}$ is an OpenGL projection[1] matrix which maps $\mathbb{R}^3 \mapsto \mathbb{R}^3$. The projection of the surface triangle $\mathfrak{w} = \{\boldsymbol{v}_i\}$ gives the triangle

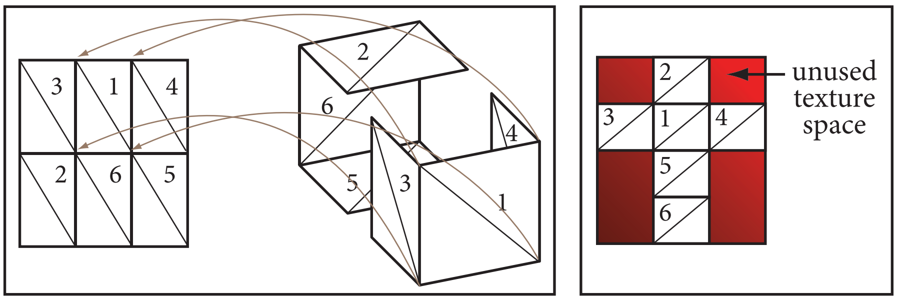
$$\mathfrak{w}' = \{\hat{\boldsymbol{K}}\boldsymbol{E}\boldsymbol{v}_i\} = \{\boldsymbol{v}'_i\}. \tag{5.1}$$

in the reference image. This triangle is mapped to the surface's texture space by the homography $\boldsymbol{H}_\mathfrak{w} : \boldsymbol{H}_\mathfrak{w}\boldsymbol{v}'_i \mapsto \underline{\boldsymbol{t}}_i$,

$$\boldsymbol{H}_\mathfrak{w} = \begin{bmatrix} \boldsymbol{v}'_1 & \boldsymbol{v}'_2 & \boldsymbol{v}'_3 \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{t}}_1 & \underline{\boldsymbol{t}}_2 & \underline{\boldsymbol{t}}_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}, \tag{5.2}$$

where $\underline{\boldsymbol{t}}_i$ are the co-ordinates in texture space associated with the vertices $\boldsymbol{v}_i$. Points in scene-space are similarly mapped to texture space by the homography $\boldsymbol{G}_\mathfrak{w} : \boldsymbol{G}_\mathfrak{w}\boldsymbol{v}_i \mapsto \underline{\boldsymbol{t}}_i$,

[1]The OpenGL projection matrix maps scene-space into camera co-ordinates via the pin-hole camera model, but projected points maintain a distance term so that the depth-buffer can be used.



(a) Full utilisation. **(b)** Partial utilisation

Figure 5.4: Texture co-ordinates define a map from the surface's geometry to an image representing the surface's texture.

given by

$$G_{\mathfrak{w}} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{t}}_1 & \underline{\boldsymbol{t}}_2 & \underline{\boldsymbol{t}}_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}. \tag{5.3}$$

The homographies $\boldsymbol{H}_{\mathfrak{w}}$ and $\boldsymbol{G}_{\mathfrak{w}}$ relate a triangle's projection in the reference image to the corresponding triangle in scene-space and in texture space, respectively. Note that these are projective homographies because they are defined by the projection $\boldsymbol{v}_i$ in projective space. Each point $\boldsymbol{p}' \in \mathfrak{w}'$ therefore has the perspectively correct corresponding points $\boldsymbol{p} = \boldsymbol{G}_{\mathfrak{w}}^{-1}\boldsymbol{p}'$ in $\mathfrak{w}$ in scene-space and $\underline{\boldsymbol{t}} = \boldsymbol{H}_{\mathfrak{w}}\boldsymbol{p}'$ in $\mathfrak{w}$'s texture space. Figure 5.5 illustrates the relationship between $\boldsymbol{G}_{\mathfrak{w}}$, $\boldsymbol{H}_{\mathfrak{w}}$ and a surface point's correspondence in the texture, scene and image spaces.

The surface's view-specific texture is generated by mapping each pixel $\boldsymbol{p}' \in \mathfrak{w}'$ to $\mathfrak{w}$'s texture space, provided the corresponding scene point $\boldsymbol{p} = \boldsymbol{G}_{\mathfrak{w}}^{-1}\boldsymbol{p}'$ is visible with respect to the particular reference camera. Using the depth-map z defined by 4.4 to check occlusion, $\mathfrak{w}$'s texture v_i in the i^{th} reference view r_i is defined by

$$\ddot{\mathsf{v}}_i(\boldsymbol{p}') = \mathsf{r}_i(\boldsymbol{H}_{\mathfrak{w}}\boldsymbol{p}') \tag{5.4}$$

and

$$\dot{\mathsf{v}}_i(\boldsymbol{p}') = \begin{cases} 1 & \text{if } d(\boldsymbol{G}_{\mathfrak{w}}^{-1}\boldsymbol{p}') \leq \mathsf{z}(\boldsymbol{p}') \\ 0 & \text{otherwise} \end{cases}. \tag{5.5}$$

where $\ddot{\mathsf{v}}_i$ and $\dot{\mathsf{v}}_i$ denotes the colour and α-channels of image v_i, respectively. The image v_i is an RGBα surface texture defined by the back-projection of r_i, where the RGB channels

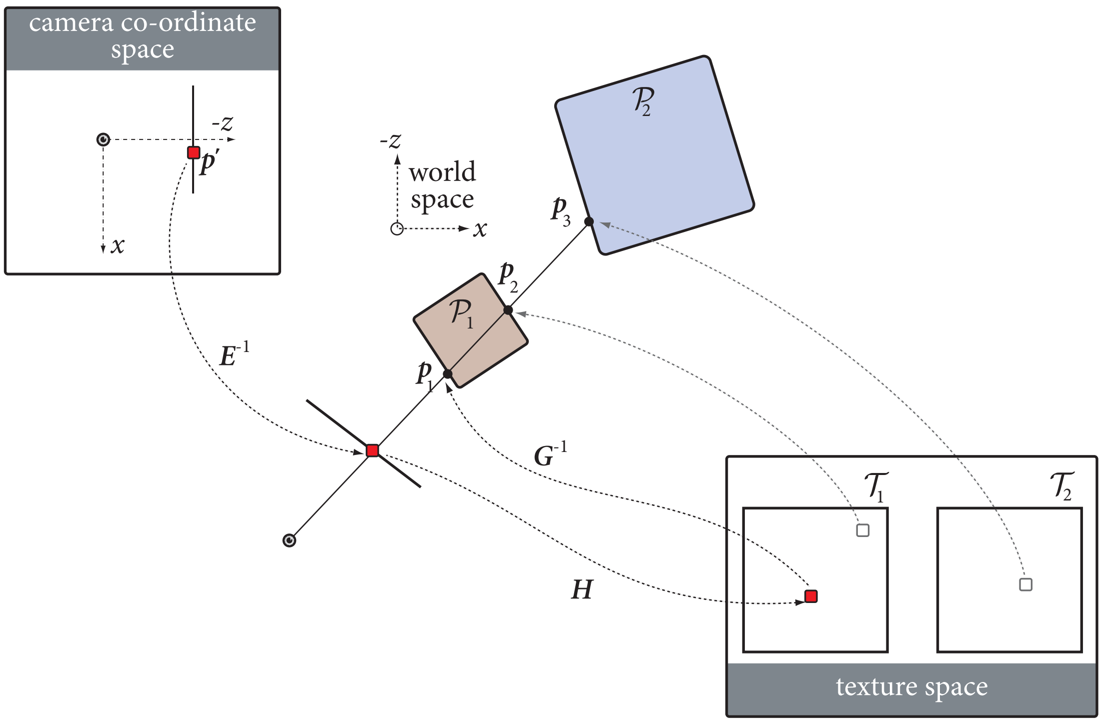


Figure 5.5: The relationship between image pixels and surface texels is defined by the surface texture-map and its projection in the image.

describes the surface reflectance and the α-channel denotes visibility in r_i. The texture image is constructed by initialising all points in $\dot{\mathsf{v}}_i$ to 0 and mapping all triangles $\varpi \in \mathcal{S}(\Psi)$ into texture space via the projection of $\mathcal{S}(\Psi)$ in the particular reference camera.

5.3 Generating surface textures with graphics hardware

View-specific textures are computed by iterating over every surface triangle and conditionally mapping the reference image into the texture image. A texel's colour is only defined if the corresponding scene point is unoccluded in the particular reference camera. This process creates a view-specific texture which guarantees that the hypothesised surface recreates the given reference image when the texture is applied to the surface and rendered with the reference camera. This property is not necessarily true for one view-specific texture imaged in other reference cameras, unless the hypothesised surface has been correctly estimated. A view-specific texture is created for each reference image so that the surface's photo-consistency can be measured.

Although mapping the reference image to texture space is a straight-forward process, it is also computationally expensive. Furthermore, a set of view-specific textures must be generated for each hypothesised surface, and a large number of samples must be drawn from the parameter space to make inferences about the posterior. This strongly motivates developing an efficient process for computing view-specific textures. Generating view-specific textures ultimately relies on conditional projective texture mapping, which is a task well suited to graphics hardware. It would seem reasonable, then, that graphics hardware could be used to generate the hypothesised surface's set of view-specific textures.

Our hardware accelerated algorithm uses graphic hardware to map the reference images into the surface's texture space. Graphics hardware, however, is designed for generating synthetic images from a geometry description, and not for generating textures from images. OpenGL cannot write into texture images. Our approach therefore stores the reference image in the graphics pipe-line's texture unit and uses the frame-buffer to represent the surface texture. While this frame-work addresses the problem of writing the results, it introduces a number of problems with regards to testing the hypothesised surface's visibility.

The key difficulty is that OpenGL can only test occlusion with respect to its projection matrix and a fragment's projection into the frame-buffer. OpenGL has only limited scope for computing visibility in spaces other than against its camera model. Frustum clipping is the most relevant example of this, although others include clipping-planes and back-face culling. These stages follow primitive assembly, [2] where incoming primitives are clipped before the results are presented to the fragment processor.

The surface's occlusion must be determined so that only points visible to the reference camera are transferred to the texture space. Determining the set of points visible to the reference camera depends on two occlusion tests:

- the surface triangles must be clipped against the reference camera's frustum, including the left/right/top/bottom planes[3] and the imaging plane; and
- of the remaining surface inside the camera's field of view, only the projection of the closest points are mapped to texture space.

[2] The order of graphics operations is described in Appendix B.

[3] The left/right/top/bottom planes are coincident with the image's edges.

Although these tests are readily performed by the graphics pipe-line, these operations only work with triangles in OpenGL's camera-space, and not against triangles in OpenGL's texture space, for example. Because the frame-buffer is used to represent the surface texture, the triangles rendered by OpenGL must leave the vertex processor in the surface's texture space, not in scene-space. As the graphics pipe-lines projection must be configured to map surface-space into the frame-buffer, *all* triangles that enter OpenGL's clipping stage will therefore be visible with respect to the OpenGL camera. Consequently, OpenGL's triangle clipping mechanics cannot be used to clip triangles against the reference camera's frustum.

Storing the reference image in texture space suggests that shadow maps could be used to model occlusion in a similar manner to that used by the image-space likelihood. While shadow maps do have a rôle in the surface-space likelihood, they cannot solve the problem of clipping against the reference camera frustum because the OpenGL pipe-line cannot clip triangles based on their texture co-ordinates.

The process of generating the surface texture must ensure that only image pixels corresponding to visible scene points are mapped into texture space. Although these requirements are not unlike visibility testing when rendering, OpenGL's clipping mechanism cannot be used because the pipe-line must be reinterpreted in a manner inconsistent with OpenGL's frame-buffer and camera model. If left unchecked, surface points not visible to the reference camera will be considered visible and therefore textured.

Two algorithms to map reference images into texture space via the scene-graph's geometry have been developed to handle surface occlusion in the re-interpreted graphics pipe. The first can be implemented against the fixed pipe-line OpenGL; but this algorithm cannot handle visibility on the graphics card, and instead must use the processor to cull and clip the surface against the reference camera's imaging plane. Clipping against the left, right, top and bottom frustum planes, however, is done on hardware by the texture clamping mechanism.

A vertex shader is used in the second algorithm to combine the triangle in texture space with depth information from the corresponding vertex in scene-space. Combining these two different spaces means OpenGL's near-plane clipping can be used to clip surface triangles against the reference camera's imaging plane while mapping the result into the surface's texture space. Like the first algorithm, texture clamping is used to clip against the frustum's edges. The advantage of the second algorithm is that it

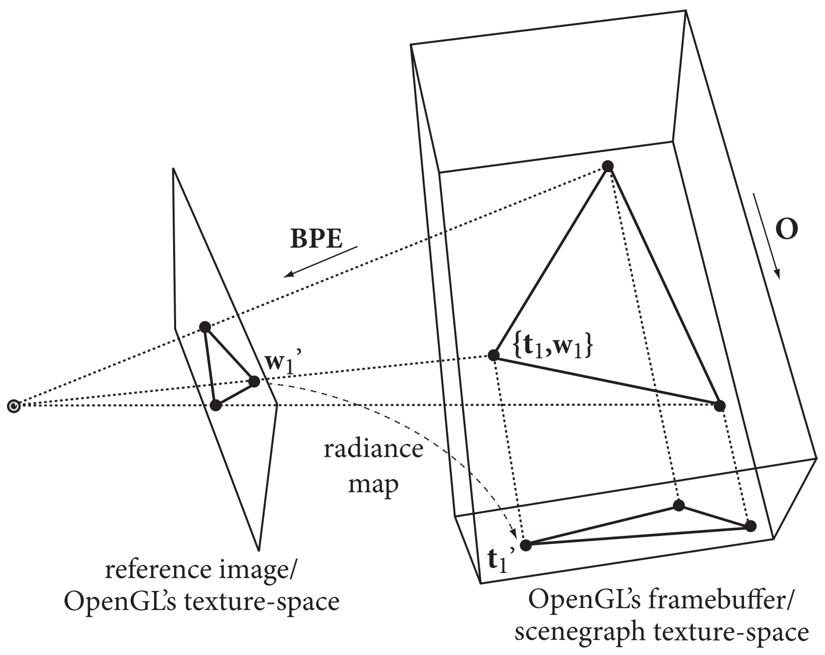

BPE
O
$\{t_1,w_1\}$
w_1'
radiance map
t_1'
reference image/ OpenGL's texture-space
OpenGL's framebuffer/ scenegraph texture-space

Figure 5.6: The fixed-pipe back-projection algorithm orthogonally projects texture triangles onto the frame-buffer while applying texture from the reference image.

makes better use of the graphics hardware: the scene-graph primitives can be compiled into display lists and clipped by graphics hardware.

5.4 Fixed-pipe back-projection

The first of the algorithms for computing the view-specific texture represents the surface's texture space in the frame-buffer and the reference image in OpenGL's texture unit. The surface's texture triangles are sent to OpenGL and orthogonally projected to the frame-buffer. The surface triangle's co-ordinates in scene-space are bound as OpenGL's texture co-ordinates; the surface triangle's projection into the reference image is computed by the texture unit. The vertex processor computes the distance between the reference camera's optical centre and the corresponding point in scene-space. This distance is compared with the closest distance along the ray between the point and the optical centre and is used to reject texels that are occluded by the rest of the scene. An overview of this process is illustrated in Figure 5.6 where points $\boldsymbol{w}_i$ represent the surface triangle's world co-ordinates and $\boldsymbol{t}_i$ are the associated texture co-ordinates.

Scene-graph triangles are rendered to the frame-buffer by combining the point's world co-ordinate and texture co-ordinate into a single vertex entity that is sent to the graphics pipe. The surface point's world co-ordinate is encoded in OpenGL's texture space and its texture co-ordinate is encoded in OpenGL's vertex space. Because an OpenGL vertex is in the surface's texture space, OpenGL's projection matrix must therefore be

$$\hat{\boldsymbol{P}} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

to map from texture space into the frame-buffer co-ordinate system. As OpenGL's texture matrix is used to project the world co-ordinate frame into image space using the reference camera's projection, it is defined as

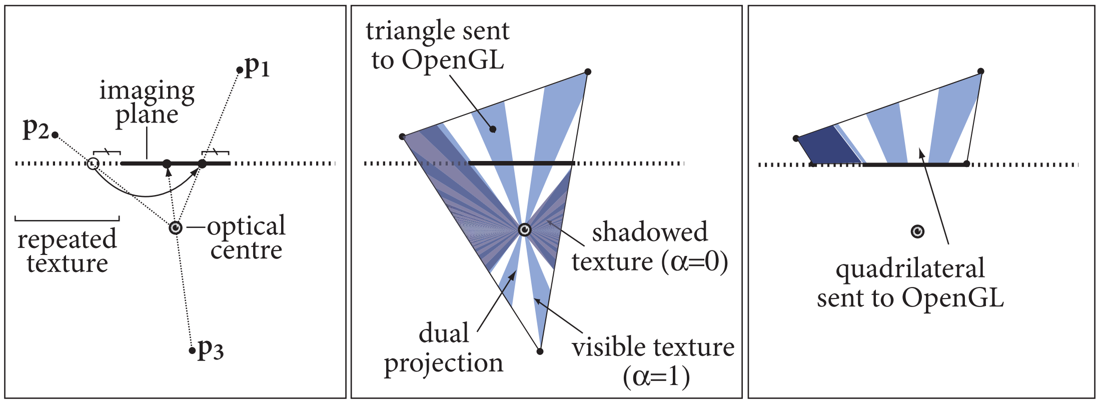
$$\boldsymbol{T}_{\mathcal{G}} = \boldsymbol{B}\hat{\boldsymbol{K}}\boldsymbol{E}, \tag{5.7}$$

where $\boldsymbol{B}$ is the 'bias' matrix from (4.11) that maps OpenGL's clip to texture co-ordinates. Per-vertex texture co-ordinates are generated in the pipe for each vertex by (5.7). These co-ordinates are in the form

$$\boldsymbol{t}_i = \boldsymbol{T}_{\mathcal{G}}\boldsymbol{v}_i = \begin{bmatrix} u \\ v \\ s \\ t \end{bmatrix}, \tag{5.8}$$

where $(\frac{u}{t}, \frac{v}{t})$ represents the projection of the vertex in the reference camera's image plane and $\frac{s}{t}$ represents the distance of the world vertex to the reference camera. These texture coordinates $\boldsymbol{t}_i$ define the homography from the reference image into the surface's texture space as the triangle is rendered.

The RGBα fragment written to the frame-buffer is a combination of the RGB fragment from the reference image and α-channel from the shadow-map. The fragment's RGB components, taken from the reference image, represents the surface texel's colour on the assumption that the corresponding point is visible. The α component models the point's occlusion using shadow-maps. Before constructing the view-specific texture, the hypothesised scene is rendered by the reference camera and the depth buffer bound as a shadow-map. Shadow maps implement depth-based occlusion in OpenGL's texture



(a) All texture references resolve to a valid texel. **(b)** Texture is applied to triangle extending behind the camera. **(c)** The triangle must be clipped so points behind the camera are not textured.

Figure 5.7: OpenGL's texture semantics treat textures as continuous entities. Triangles must be clipped before sent to the graphics pipe so that points within them, but behind the reference camera, are marked as occluded.

space. Points *behind* the camera can also project within the image domain; but the distance of these points cannot be distinguished from the distance of points in front of the camera.

OpenGL can clip triangles based only on the vertices which are defined in clip co-ordinates. In the context of generating view-specific textures, however, OpenGL's scene-space is used to represent triangles in the surface's texture space and therefore the triangles in clip co-ordinates are in fact the surface's triangles in texture space. These triangles are always visible with respect to OpenGL's clipping semantics. Because OpenGL's clipping mechanism cannot be modified to clip triangles using the hardware texture matrix, the surface triangles must therefore be clipped against the reference image plane before being sent to the graphics pipe-line.

5.4.1 Pre-processing triangles

Unless the surface triangles are clipped against the reference camera's imaging plane, the surface *behind* the camera will be erroneously textured with the reference image. This problem is illustrated in Figure 5.7(a). The point $\boldsymbol{p}_2$, although outside the reference frustum, maps to a point within the image, and will therefore be assigned the same colour as the point $\boldsymbol{p}_1$ because of OpenGL's modulo texture co-ordinate system.

Fortunately, OpenGL is able to clamp co-ordinates against the texture's border. Although this is not useful when mapping texture from the reference image (because a RGB colour is assigned, regardless), clamping is effective with shadow mapping. Clipping against the reference image boundaries is implemented by simply clamping texels to the depth buffer's border and setting the border depth to $+\infty$. While clamping is used to clip points against image boundaries, this does not work for points behind the reference camera that project within the image boundary, such as $\boldsymbol{p}_3$ in Figure 5.7(a).

The fixed-pipe texture algorithm must therefore clip triangles against the reference camera's image plane to prevent texture being applied to surface elements behind the reference view. The primitives cannot be clipped by OpenGL against the near plane in clip-space because the OpenGL camera is not used to model the projection by the reference image. OpenGL's clipping planes cannot be trivially used either, since clipping planes partition primitives in clip-space. Consequently, a clipping plane would have to be defined per triangle to transform the reference camera's imaging plane into the surface's texture space.

Our solution must clip the texture triangles on the processor before they are sent to the graphics pipe (see Figure 5.7(c). There are two disadvantages with this approach. Firstly, it is relatively slow compared to clipping triangles using graphics hardware because triangles can potentially be clipped into quadrilaterals, and the software must interpolate vertex attributes (such as its world and texture co-ordinates) for the new vertices. The second disadvantage is that clipped geometry must be regenerated for every instance of the hypothesised scene configuration, which precludes compiling the primitive geometry into a display list.

Triangles must be clipped when its edges cross the reference camera's imaging plane. Because the camera's optical axis is aligned with the $-z$-axis in camera co-ordinates by the extrinsic matrix, a point in scene-space is behind the camera if

$$\frac{[0,0,1,0]\boldsymbol{E}\boldsymbol{v}}{[0,0,0,1]\boldsymbol{E}\boldsymbol{v}} > 0. \tag{5.9}$$

Vertices behind the camera must be clipped so that all points on the triangle's surface satisfy (5.9).

Suppose that a line segment is defined by two vertices $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in camera co-ordinates, and that $\boldsymbol{v}_1$ is in the negative half-space of the image plane $z = -f$. If the line segment $\overset{\longmapsto}{\boldsymbol{v}_1\boldsymbol{v}_2}$ is parameterised by λ,

$$r(\lambda) = \boldsymbol{v}_1 + \lambda(\boldsymbol{v}_2 - \boldsymbol{v}_1) \tag{5.10}$$

then the line will intersect the image plane $\boldsymbol{p} = [0, 0, 1, -f]^\top$ when $p^\top r(\lambda) = 0$. The new line segment is therefore $\overset{\longmapsto}{r(\lambda')\boldsymbol{v}_2}$, where

$$\lambda' = \frac{-\boldsymbol{p}^\top \boldsymbol{v}_1}{\boldsymbol{p}^\top \boldsymbol{v}_2 - \boldsymbol{p}^\top \boldsymbol{v}_1} \tag{5.11}$$

is the parameter where $r(\cdot)$ intersects the imaging plane.

Clipping each triangle against the imaging plane creates a new primitive entirely in front of the reference camera. This process could be further used to clip against the frustum planes coincident with the image boundary. Rather than clipping the polygon against the frustum edges, fragments falling outside the frustum can be marked as occluded 'for free' by virtue of the texture clamping mechanism. Additional fill-rate is the only small disadvantage of this approach, but is more efficient than clipping against five planes.

5.4.2 *Combining colour and occlusion*

The set of view-specific textures is generated by iterating over each surface triangle associated with a given texture image. The reference image is bound on one texture unit and the depth-map of the entire surface when imaged from the reference camera's view-point is bound as a shadow-map on another. The OpenGL texture matrix is configured with the reference camera's projection from (5.7) and the OpenGL projection matrix is configured to map texture triangles into the frame-buffer by (5.6).

Each surface triangle is sent to the graphics pipe by specifying the surface's world co-ordinates in OpenGL's texture space and the surface texture co-ordinates in OpenGL's vertex space. The triangle is textured as it is written to the frame-buffer to map RGB texels from the reference image by projecting each texture co-ordinate into texture space. Similarly, the frame-buffer's α-channel is defined by the shadow-map so texels corresponding to visible surface elements have $\alpha \leftarrow 1$ whereas $\alpha \leftarrow 0$ indicates the corresponding point is occluded. The complete algorithm is listed in Algorithm 2.

Algorithm 2: Fixed-pipe texture algorithm.

```
Input  : A reference camera's projection $\hat{\boldsymbol{K}}$ and extrinsic matricies $\boldsymbol{E}$; the set
         of surface triangles $\{\mathrm{t}_i^j\}$ where $\mathrm{t}_i^j$ is the $i^{\mathrm{th}}$ triangle associated with
         texture $j$.
Output: The view-specific texture set $\mathcal{I}_j$ that recreates the reference view.
/* First pass: generate the depth-map for the reference image                    */
1  $\boldsymbol{P}_{\mathcal{G}} \leftarrow \hat{\boldsymbol{K}}$; $\boldsymbol{M}_{\mathcal{G}} \leftarrow \boldsymbol{E}$
2  z $\leftarrow \infty$; enable depth testing
3  for $\forall i, \forall j$ do
4      render($\mathrm{t}_i^j$)
5  endfor
6  z $\leftarrow$ depthbuffer
/* Second pass: generate the depth-map for the reference image                   */
7  $\boldsymbol{P}_{\mathcal{G}} \leftarrow (5.6)$; $\boldsymbol{M}_{\mathcal{G}} \leftarrow \boldsymbol{I}$; $\boldsymbol{T}_{\mathcal{G}} \leftarrow (5.7)$
8  bind z as a $\alpha$ shadow-map on texture unit 1; bind r on texture unit 2
9  disable depth testing
10 for $\forall j$ do
11     attach $\mathcal{I}_j$ as a RGB$\alpha$ render-target
12     for $\forall i$ do
           /* Render clipped triangle $\mathrm{t}_i^j$                                */
13         $\{\boldsymbol{v}'_k\} \leftarrow$clip $\mathrm{t}_i^j$ against the plane $[0, 0, 1, 0]$
14         render($\{\boldsymbol{v}'\}$)
15     endfor
16 endfor
```

5.5 Hardware texture algorithm

The algorithm from Section 5.4 must clip primitives on the processor because the graphics hardware cannot clip against texture co-ordinates. The introduction of vertex and fragment shaders in OpenGL 2.x's programmable pipe-line allows for more control over vertex attributes. Although the programmable pipe-line cannot clip triangles against texture co-ordinates, the view-specific texture algorithm presented in this Section is able to combine the projection of a surface triangle and the corresponding texture triangle into a single triangle which is mapped into the frame-buffer. This triangle can be clipped with respect to the reference camera's imaging plane while the texture space triangles are written into the frame-buffer.

OpenGL clips vertices against the imaging plane based on the vertex's $z-$ co-

Algorithm 3: Programmable-pipe vertex shader.

```
1 $\underline{\boldsymbol{t}}' \leftarrow \boldsymbol{M}\boldsymbol{p}$                                  // scene point in camera co-ordinates
2 $\boldsymbol{p}'_{xy} \leftarrow 2\underline{\boldsymbol{t}}_{xy} - 1$                  // map scene texture co-ordinate to the frame-buffer
3 $\boldsymbol{p}'_z \leftarrow \frac{-2\underline{\boldsymbol{t}}'_z}{(f-n)\underline{\boldsymbol{t}}'_w} - 1 - n$       // scale scene point distance to reference camera
4 $\boldsymbol{p}'_w \leftarrow 1$                                      // orthogonal projection onto the frame-buffer
5 $\underline{\boldsymbol{t}}' \leftarrow \boldsymbol{P}\underline{\boldsymbol{t}}'$                                 // project the scene point into the reference image
6 return $\{\boldsymbol{p}', \underline{\boldsymbol{t}}'\}$
```

ordinate in clip space; but the vertices must also map into the frame-buffer in a manner consistent with the texture triangle's map into texture space. Fortunately, the vertex shader hardware allows a non-linear combination of vertex attributes which can be used to combine the surface point's distance from the camera with the point's map into texture space. This process writes triangles into texture space while the graphics hardware culls fragments whose corresponding point in scene-space is behind the reference camera.

This has two advantages over the fixed-pipe algorithm. Firstly, the triangles are clipped by graphics hardware, which can be potentially faster than software clipping. Secondly, the geometry is static, regardless of the surface's orientation to the reference camera. Consequently, the scene-graph geometry can be compiled into display lists. Display lists are potentially a faster method of describing geometry than OpenGL's immediate mode, which is required for the algorithm in Section 5.4.

5.5.1 *The vertex shader*

Key to our algorithm is a vertex shader which combines the surface's corresponding world and texture vertices. The input to this vertex shader is a surface point in the primitive's local co-ordinate space and the corresponding point in the surface's texture space. The output from the shader is a vertex that maps to the frame-buffer in a manner consistent with the point's map into texture space, but whose depth is a function of the surface point's distance to the reference camera. This depth is defined such that points on the reference image plane have depth $z = -1$ and surface points that are furthest from the reference camera have depth $z = 1$.

The output of all vertex shaders is a point in normalised device coordinates.

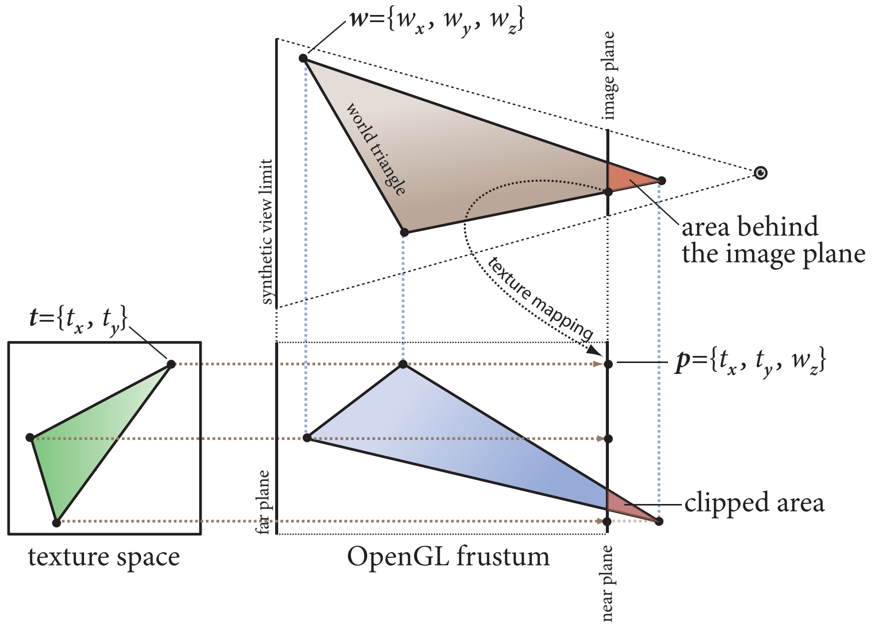


Figure 5.8: Triangles are clipped by scaling the depth of each vertex according to its corresponding depth in scene-space.

Since the frame-buffer represents the primitive's texture space, the point's x and y co-ordinates are given by

$$\underline{\boldsymbol{t}}'_i = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \underline{\boldsymbol{t}}_i - \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{5.12}$$

which maps the point's texture space to the frame-buffer. The distance from the reference camera's optical centre to the point in scene-space is

$$z = \frac{\hat{\boldsymbol{K}}_3 \boldsymbol{E} \boldsymbol{w}_i}{\hat{\boldsymbol{K}}_4 \boldsymbol{E} \boldsymbol{w}_i}, \tag{5.13}$$

where $\hat{\boldsymbol{K}}_i$ is the i^{th} row of the OpenGL-class projection matrix $\hat{\boldsymbol{K}}$. This distance is mapped into normalised device co-ordinates by

$$d(z) = \frac{2(z-n)}{h-n} - 1, \tag{5.14}$$

where n is the focal length of the reference camera and $z = h$ is equivalent to OpenGL's far plane. The far plane $z = h$ must be sufficiently far away from the reference camera so that all scene points are on its negative half-space. The map to the frame-buffer defined by equation 5.12 and the depth defined by equation 5.14 is combined to form the vector

$$\boldsymbol{v} = \begin{bmatrix} \underline{\boldsymbol{t}}'_i \\ d(z) \\ 1 \end{bmatrix} \tag{5.15}$$

which defines the vertex in normalised device co-ordinates returned by the vertex shader.

The shader computes a vertex position that maps to the frame-buffer in a manner consistent with the surface point's map into texture space, yet its depth is a function of the surface point's distance to the reference camera. Surface points that are behind the camera are designed to leave the vertex shader outside the canonical cube in clip-space so that new vertices are created along edges that cross the plane $z = 1$ corresponding to the reference camera's imaging plane.

Fragments are textured by projecting the surface's texture co-ordinates into the reference image, but only texels in front of the camera are written into frame-buffer. Again, shadow-mapping is used to test visibility with respect to the hypothesised surface and the reference camera. The vertex shader pseudo-code to map a scene-graph point from texture to the frame-buffer is given in Algorithm 3 and is illustrated in Figure 5.8.

5.6 Results

Figure 5.9 is an example of a view-specific texture generated by back-projecting the image onto a scene with correct parameters. The left image is the texture's RGB component and the right image is its α-channel, where black denotes texels occluded with respect to the reference view and the surface. The texture corresponds to a ground plane from a scene with a plane and cube. The two sources of occlusion are from the cube's projection onto the plane (the inner region) and points outside the reference view's frustum (the outer region). Note that the reference view is repeated numerous times on the plane: but all but one copy is masked by the α-channel. This is an example of the repeated texture co-ordinate space from OpenGL's texture co-ordinate generation described in Section 5.4.1 and illustrated in Figure 5.7.

Figure 5.10 illustrates synthetic images of reference surface from three different perspectives while using the view-specific texture of one camera. The reference images are illustrated in the top row. The first image is used to generate the view-specific

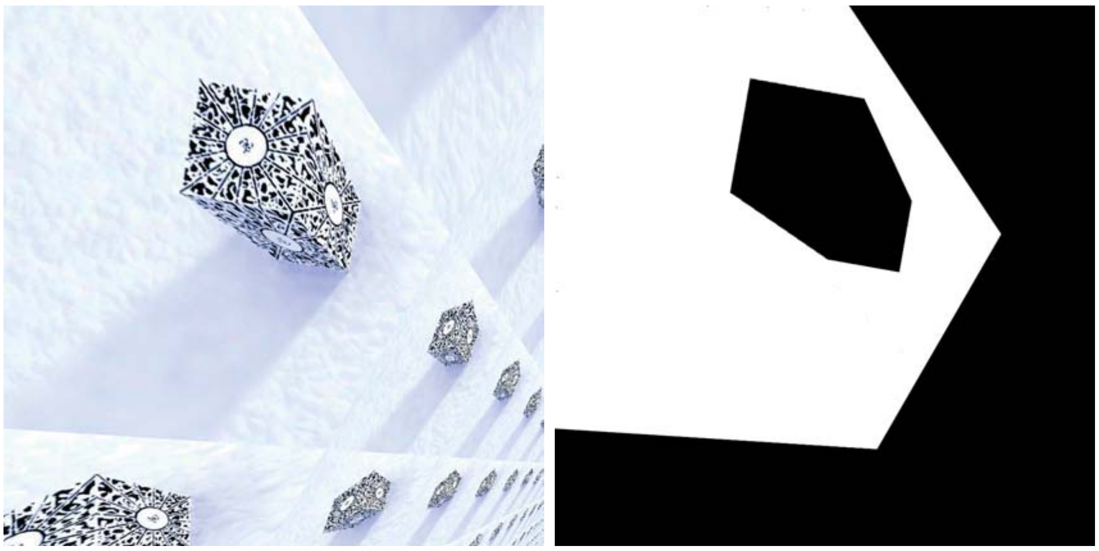


Figure 5.9: Plane texture generated by back-projecting one image onto the ground plane.

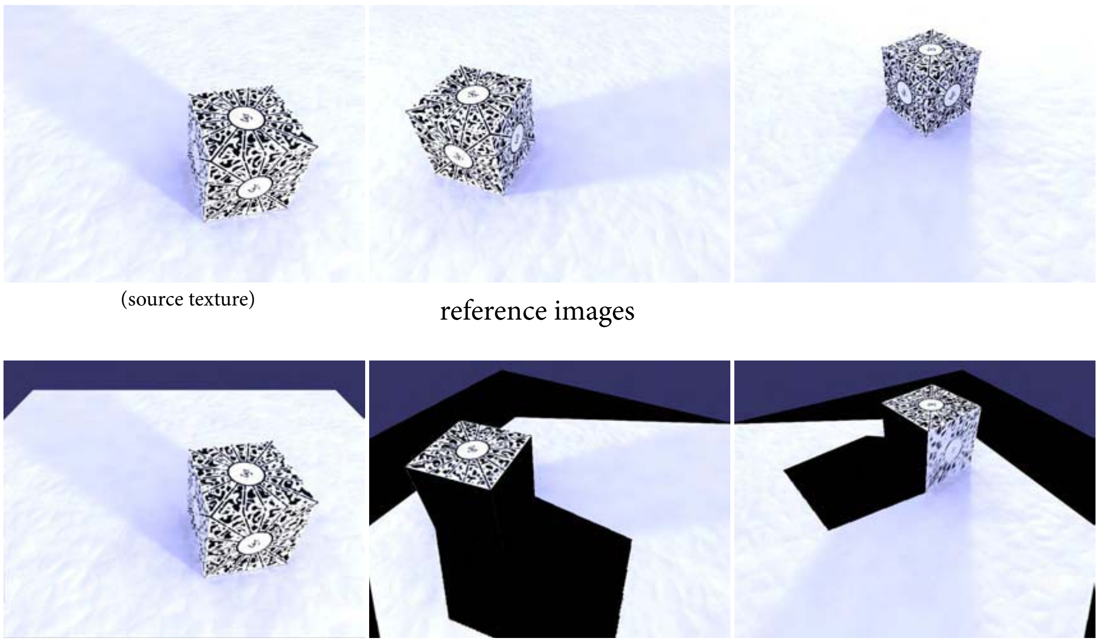
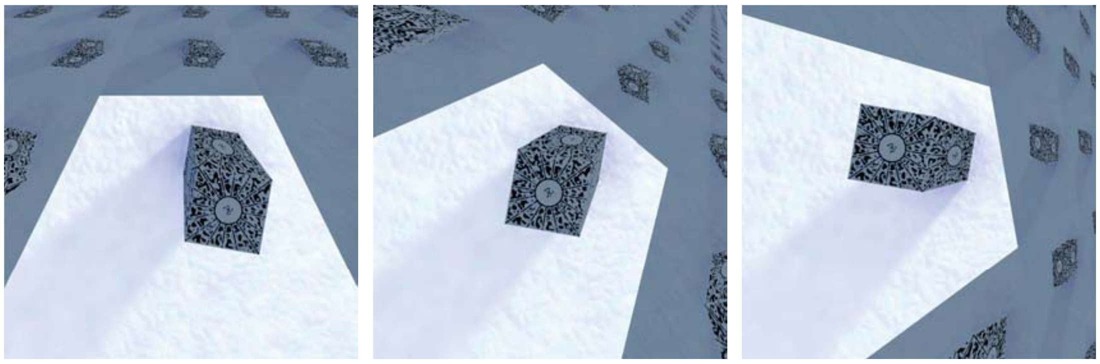


Figure 5.10: The juxtaposition of three reference views and the synthetic image generated using the corresponding reference camera positions while using the view-specific texture of one image.


(a)

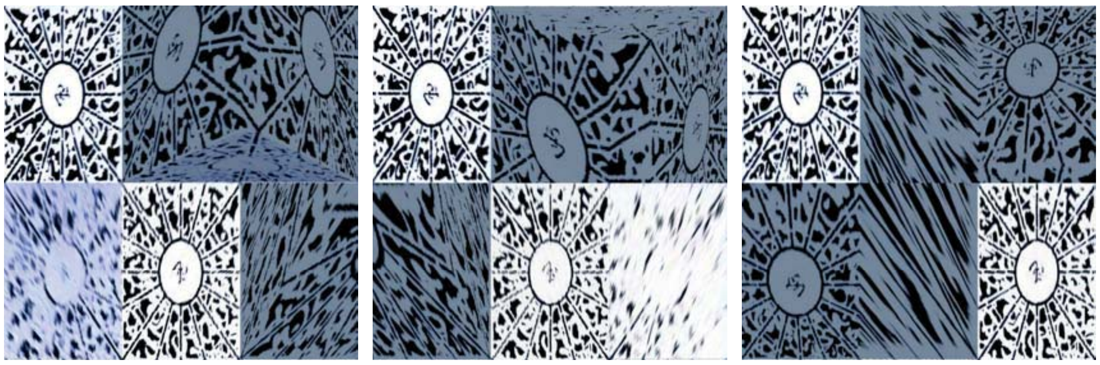

(b)

Figure 5.11: Three view-specific plane textures stylised by shading occluded pixels.

texture. This texture is applied to the surface and imaged from each of the reference camera positions. The synthetic images recreate the reference view but only on the areas also visible in the source view.

Further examples of the back-projected texture are illustrated in Figures 5.11(a) and 5.11(b), representing the ground plane and cube texture respectively. The cube textures illustrates the assignment of the same colour to all surface points with co-incident projection. The texture applied to surface points facing away from the reference camera appear distorted, in this case, but are correctly identified as being occluded. Note that the top face is visible in all three views and is assigned the same texture across all three view-specific textures, indicated by the top, left-most quadrilateral in each view-specific texture. The cube's texture map is the configuration illustrated in Figure 5.4. The face denoted by the bottom far right quadrilateral is significantly different in the second and third texture, and is significantly different in the first, but is marked as occluded. The degraded image in the second view-specific texture is caused by the oblique projection of the corresponding triangles in scene space. This is a problem with sampling the surface and is described in the next Chapter.