Assembling The Composite Texture

Chapter VI

Back-projecting each reference image onto the scene generates a set of view-specific textures, thereby simplifying the problem of correlating the hypothesised observations for each surface point. Each view-specific texture represents a potentially unique subset of the points sampled across the hypothesised surface. The set of view-specific textures are evaluated for photo-consistency by measuring the variance of each corresponding texel across the set of view-specific textures. Evaluating the variance requires first generating the surface's average observation. Computing the average colour from a set of observations is not difficult; nor is it difficult to blend images in graphics hardware. The difficulty with computing the average, '*composite*' texture is that the view-specific textures cannot be simply blended or accumulated together in graphics hardware because not every surface element is visible in all cameras.

6.1 Composition on graphics hardware

There are three problems involved in computing the hypothesised surface's composite texture on graphics hardware. Firstly, the number of view-specific textures that contribute to the point's average must be counted so that the per-texel sum can be normalised. Although simply counting the number of unoccluded pixels is straightforward (supported by the CPU and stencil buffers, for example) it is not clear how this normalisation map could be best applied. Secondly, the accumulation must be careful to not saturate the frame-buffer because OpenGL 1.0 clamps colour attributes to the range $[0, 1]$ at multiple stages through the graphics pipe-line. Finally, the colour frame-buffer is typically only 8-bits per channel, which is likely to have an effect on the precision of the composite texture.

OpenGL 2.0 has added support for floating-point buffers, thereby eliminating a number of these technical concerns related to the frame-buffer. Indeed, OpenGL 2.0's support for floating-point buffers motivates the development of two algorithms to com-

pose the view-specific textures into a single surface representation. The first algorithm does not require floating-point buffers and can work in OpenGL's clamped colour space. The second algorithm uses OpenGL 2.0's floating-point buffers. The increased precision and un-clamped space of the floating-point algorithm allows for a more sophisticated weighted average of the view-specific textures. Texels in this algorithm are weighted as a function of the angle between the corresponding surface normal and view-vector. This approach reduces filtering artifacts introduced by an equal consideration of all observations. Addressing this problem is particularly important for samples taken from oblique surfaces, because such samples can introduce artifacts which lead to errors in computing the surface's photo-consistency.

6.2 Composition in clamped colour buffers

Back-projecting each reference view onto a hypothesised surface generates a potentially unique view-specific texture v_i. Distorting each reference view into a common texture space via the surface's texture map simplifies the problem of computing the variance of hypothesised point correspondences. Accordingly, the photo-consistency of a surface point $\boldsymbol{p}$ under a perfectly diffuse reflectance-model is inversely proportional to

$$Var\{\, \ddot{v}_i(\boldsymbol{p}) \mid i : \dot{v}_i(\boldsymbol{p}) \neq 0 \,\}, \tag{6.1}$$

where $\dot{v}_i$ is the occlusion mask associated with the view-specific texture v_i (ie. the α–texture illustrated in Figure 5.9) and $\ddot{v}_i$ is the observed colour from the i^{th} reference image.

Determining the surface's colour variance requires first determining the average reflectance. This texture is computed by a per-texel average of all view-specific textures and represents the hypothesised surface reflectance independent of any reference view. If all surface points were visible in all cameras, then the *composite texture* t could be computed simply by blending all view-specific textures in the frame-buffer. This technique cannot be applied when some texels are be marked as occluded, however, because not accounting for occlusion would tacitly contribute a black observation to the average. This effectively under-exposes partially occluded surface-points and therefore leads to an erroneously high variance.

Our algorithm handles occlusion by twice iterating through the set of view-specific textures. The first pass counts the number of unoccluded samples that con-

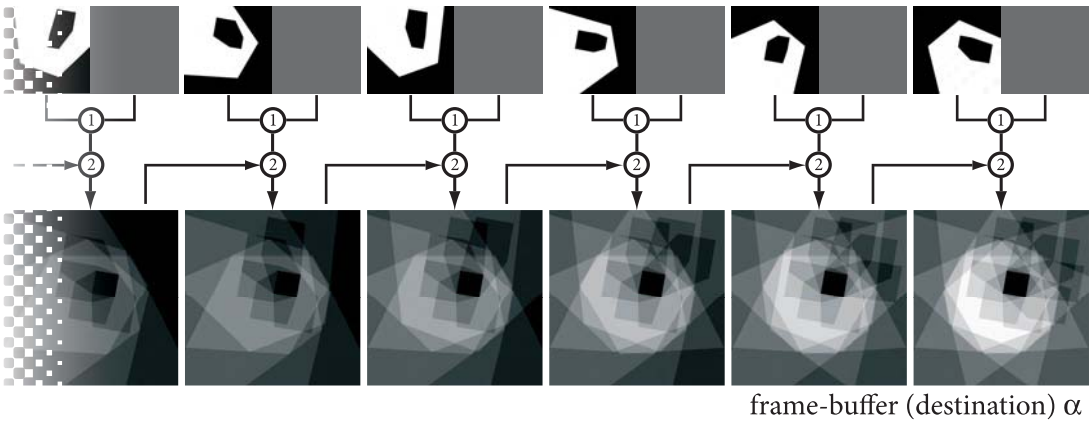


Figure 6.1: The surface's visibility map is computed by modulating each view-specific visibility mask with the frame-buffer.

tribute to each point in t, storing the result in the α-channel. This information is then used to weight corresponding points in each view-specific texture in the second pass. Because the number of observations is known, each texel in the view-specific texture is appropriately scaled so the frame-buffer is not saturated while also normalising the per-texel average.

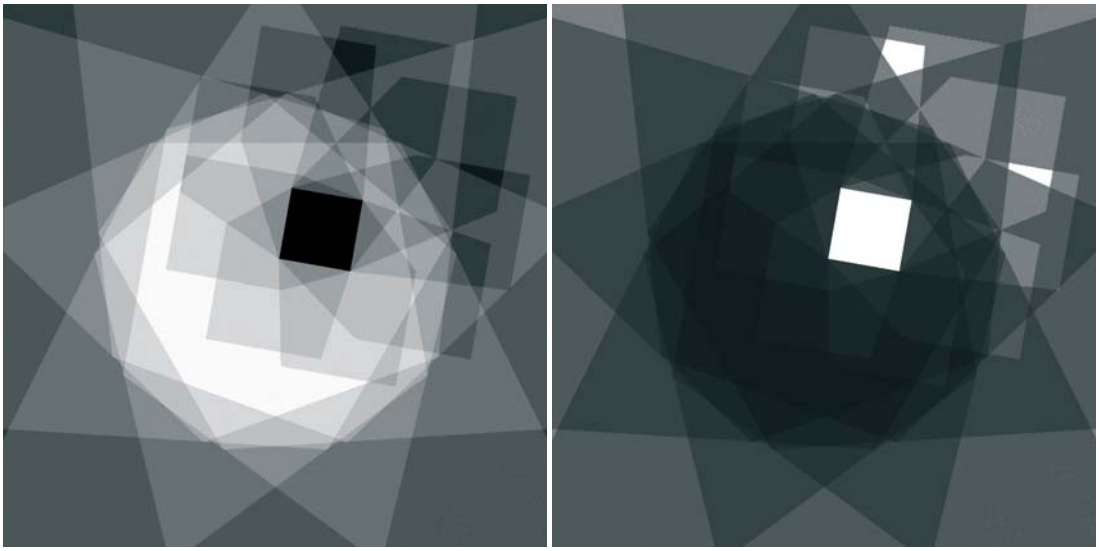
6.2.1 *Computing surface visibility*

Accumulating the α-channels of the view-specific textures determines the number of observations contributing to t. The occlusion mask $\dot{\mathrm{v}}$ is mapped to the frame-buffer by first modulating the texture onto a quadrilateral with colour $[1, 1, 1, n^{-1}]^\top$. This effectively scales $\dot{\mathrm{v}}$ by $\frac{1}{n}$ to ensure that the frame-buffer is never saturated, even if a point is visible by all n reference images. The quadrilateral is orthogonally projected onto the frame-buffer and accumulated by using 1 for both the source and destination co-efficients. Iterating over all view-specific textures generates the visibility map ω,

$$\omega(\boldsymbol{p}) = \sum_{i=1}^{n} \frac{1}{n} \dot{\mathrm{v}}(\boldsymbol{p}). \tag{6.2}$$

stored as per-texel visibility metric in the frame-buffer. Figure 6.1 illustrates the process of combining $\dot{\mathrm{v}}$ to compute ω. The process that modulates $\dot{\mathrm{v}}$ is denoted by (1), and the process that accumulates each modulated $\dot{\mathrm{v}}$ with the frame-buffer is denoted by (2).

The image ω indicates the visibility of each surface point in $\mathcal{S}(\Psi)$. Inverting ω



sample count (f) texel weight (f^{-1})

Figure 6.2: Per-pixel contribution weights are computed by inverting the visibility image.

yields the per-texel weight that each unoccluded sample contributes to t. The visibility map is inverted by binding ω as a texture and using the fragment processor to compute

$$\varsigma(\boldsymbol{p}) = \frac{1}{n\omega(\boldsymbol{p})}. \tag{6.3}$$

Figure 6.2 illustrates the per-texel weight generated by equation 6.3. Note that zero visibility is represented as 1; but this does not introduce artifacts into the composite image since zero-visibility necessarily means that no texels rely on these weights.

6.2.2 *Compositing view-specific textures*

The composite surface texture t is generated by accumulating the results of applying the per-texel weights ς to each view-specific texture v_i. Since ς is the inverse sum of all visible texels, the composite texture is computed by pre-multiplying view-specific layers by their α-channel and modulating the result by the corresponding weight in ς. The composite texture t is therefore given by

$$\mathrm{t}(\boldsymbol{p}) = \sum_{i=1}^{n} \varsigma(\boldsymbol{p})\dot{\mathrm{v}}(\boldsymbol{p})\ddot{\mathrm{v}}(\boldsymbol{p}). \tag{6.4}$$

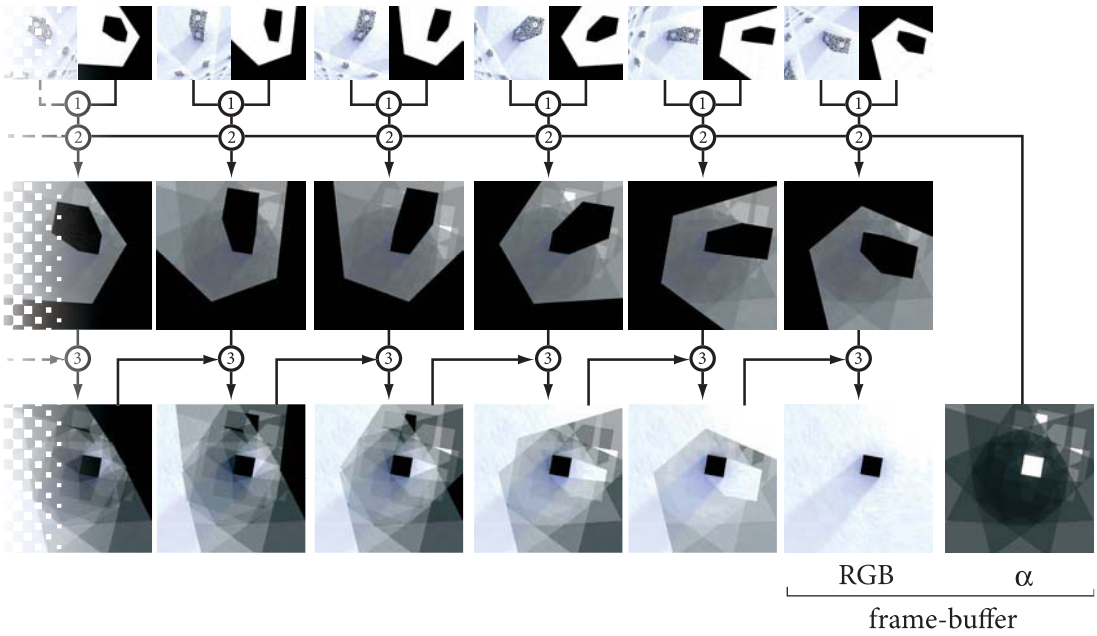


Figure 6.3: Assembling view-specific textures together into a single composite texture representing the hypothesised surface reflectance.

Figure 6.3 illustrates the process of assembling multiple observations of a single view. The process of combining the view-specific texture $\ddot{\mathrm{v}}$ with the visibility mask $\dot{\mathrm{v}}$ is denoted by (1). Re-weighting this result with ς is denoted by (2), and the process of accumulating the results is denoted (3).

An unoccluded texel $\boldsymbol{p}$ in a view-specific texture can be modulated by its corresponding weight $\varsigma(\boldsymbol{p})$ in one of two ways. One approach is to leave ς in the frame-buffer after computing Equation 6.3 and use the blender to combine it with the incoming view-specific texture. The second approach modulates the texels from $\ddot{\mathrm{v}}$, $\dot{\mathrm{v}}$ and ς together in a fragment shader so the result can simply be accumulated with the frame-buffer. The task of both algorithms is to compute

$$\mathrm{t}_i(\boldsymbol{p}) \leftarrow \mathrm{t}_{i-1}(p) + \varsigma(\boldsymbol{p})\dot{\mathrm{v}}(\boldsymbol{p})\ddot{\mathrm{v}}(\boldsymbol{p}) \tag{6.5}$$

where t_i represents the composite texture from the i^{th} iteration after composing $[1, i]$ view-specific textures from the set $\{\mathrm{v}_i | i \in [1, n]\}$. The initial texture, t_o is black (ie. the initial texture does not reflect any light) and the final composite texture, $\mathrm{t} = \mathrm{t}_n$, represents the hypothesised surface's estimated reflectance.

Storing visibility weights in the frame-buffer

The first algorithm to compute t stores ς in the frame-buffer, therefore moving equation 6.5 entirely onto the blender. The blend function's RGB source co-efficient must therefore be `GL_DST_ALPHA` to modulate the incoming fragments by the weight stored in the frame-buffer. This precludes, however, modulating the fragment's colour by the α-channel (since this would require the source co-efficient to be `GL_SRC_ALPHA`). The 'correct' solution is to pre-multiply the fragment's α-channel in the fragment processor; but an easier approach is to simple enable α – testing to reject fragments with $\alpha = 0$.

The frame-buffer must maintain the integrity of ς in the α-channel because the per-texel weight is applied to all view-specific textures. The blend equation must therefore be separated into RGB and α co-efficients so the view-specific colour $\ddot{\mathrm{v}}$ is modulated and accumulated but the associated visibility mask $\dot{\mathrm{v}}$ is not. The blender must therefore compute

$$\ddot{\mathrm{t}}_i(\boldsymbol{p}) = \dot{\mathrm{t}}_{i-1}(\boldsymbol{p})\ddot{\mathrm{v}}_{i-1} + 1\ddot{\mathrm{t}}_i \tag{6.6}$$

$$\dot{\mathrm{t}}_i(\boldsymbol{p}) = 0\dot{\mathrm{v}}_{i-1} + 1\dot{\mathrm{t}}_{i-1} \tag{6.7}$$

given the incoming RGBα fragment $[\ddot{\mathrm{v}}_i|\dot{\mathrm{v}}_i]$ and the frame-buffer pixel from t_{i-1}. The 'initial' composite texture is given by $\mathrm{t}_0(\boldsymbol{p}) = [0, 0, 0, \varsigma(\boldsymbol{p})]^\top$.

Modulating visibility weights in the fragment processor

An alternative to storing ς in the frame-buffer is to bind it as texture and modulate images b and ς in the fragment processor. In this case, the blender computes

$$\mathrm{t}_i(\boldsymbol{p}) = 1f(\boldsymbol{p}) + 1\mathrm{t}_{i-1}(\boldsymbol{p}), \tag{6.8}$$

for the i^{th} view-specific texture, where

$$f(\boldsymbol{p}) = \varsigma(\boldsymbol{p})\ddot{\mathrm{v}}_{i-1}(\boldsymbol{p})\dot{\mathrm{v}}_{i-1}(\boldsymbol{p}) \tag{6.9}$$

is the result of the fragment shader. While α-testing is required in previous algorithm, it can be also be used here to cull fragments from $f(\cdot)$ and therefore save the overhead of Equation 6.8.

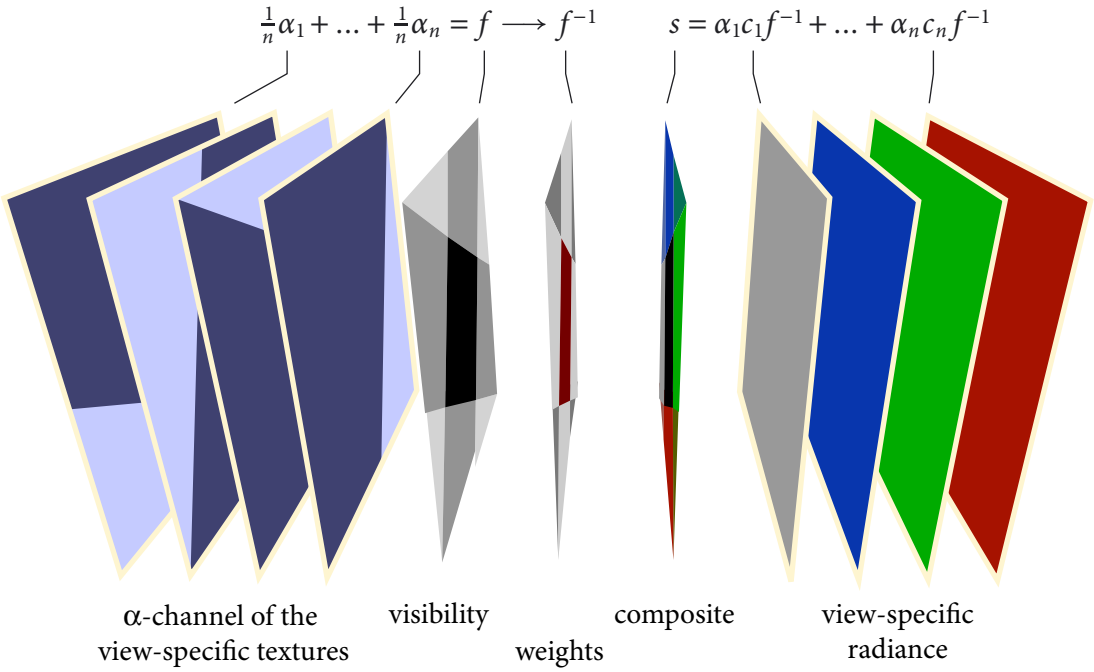
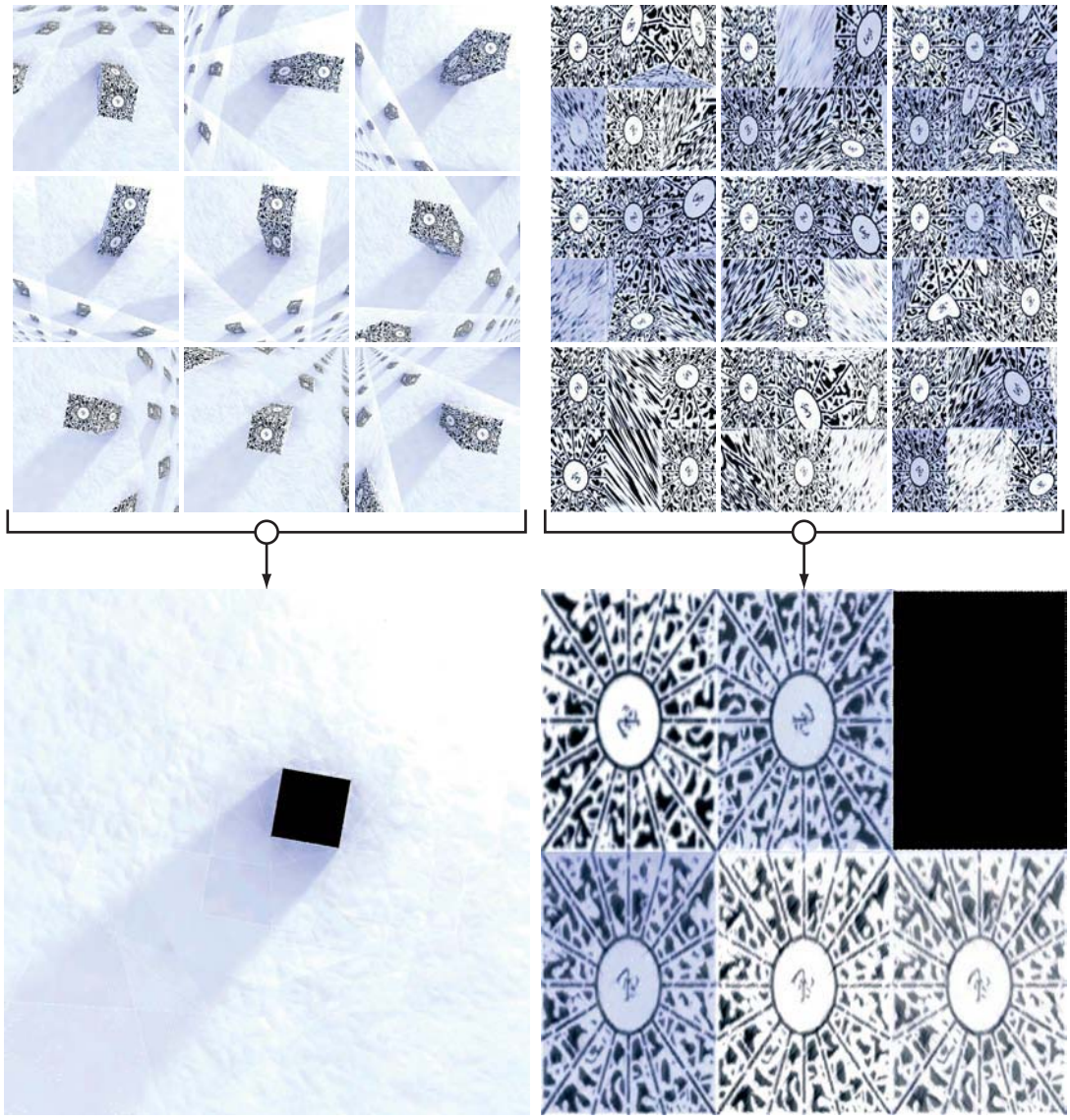


Figure 6.4: Overview of assembling the composite texture.

6.2.3 *Summary*

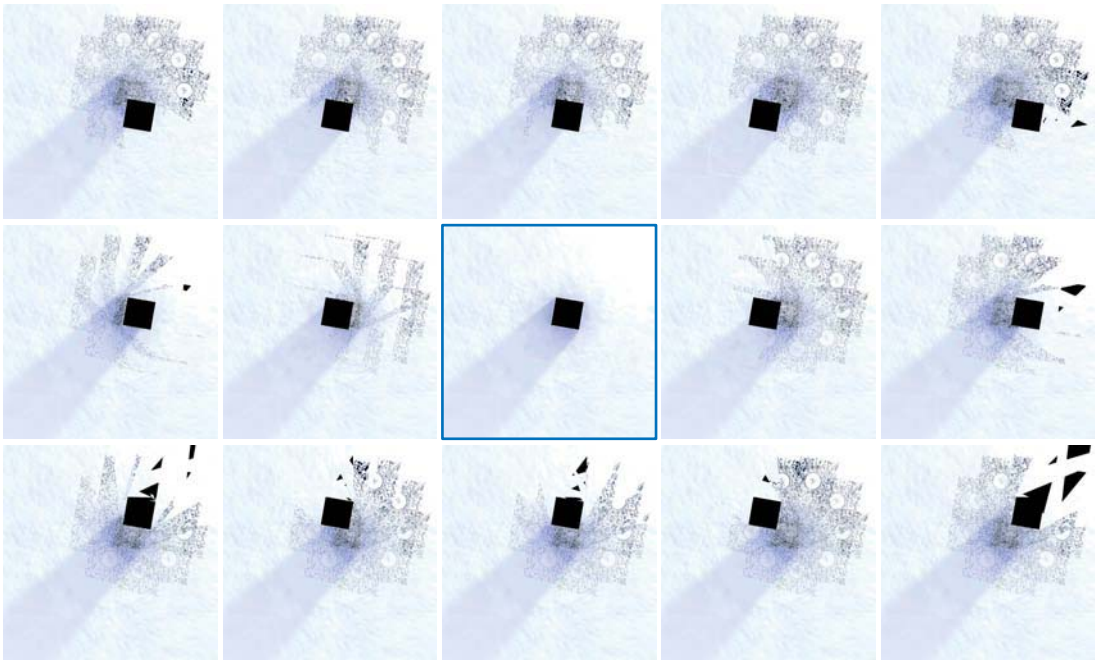
The complete algorithm for compositing a set of view-specific textures is given in Algorithm 4 and is illustrated in Figure 6.4. The algorithm requires three stages. The first collects the visibility mask across all images to compute the summed visibility texture; this texture is inverted and scaled to provide a weight image. Unoccluded texels can be appropriately scaled without the aggregate saturating the frame-buffer since the inverse sum is known when revisiting each view-specific texture.

Figure 6.5 is an example of two textures assembled from nine view-specific images at truth. Although not illustrated, the visibility mask of each view-specific texture is used to ignore erroneous colour attribute to each texture—the projection of the cube on the ground plane, for example. The composite texture is consistent with each view-specific texture insofar as the unoccluded texel of the view-specific texture has an identical colour to the corresponding texel of the composite texture. This property is held by the composite texture because the underlying hypothesised surface is at truth. This is not true for the textures illustrated in Figure 6.6 which illustrate the composite texture from hypothesised surfaces around truth.

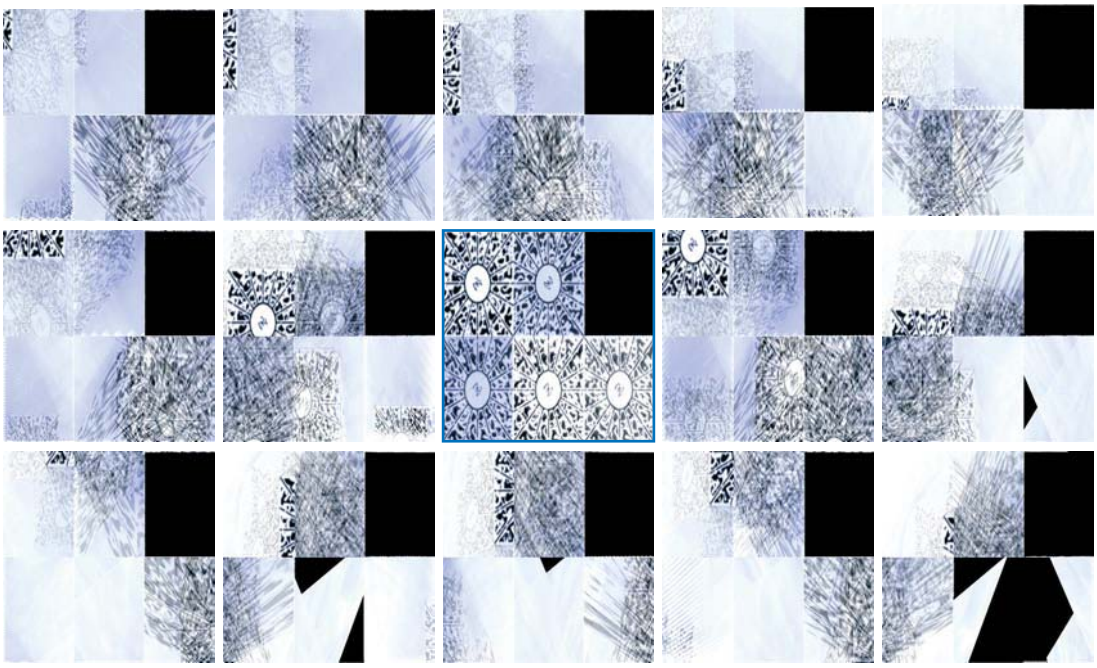


plane texture cube texture

Figure 6.5: An example of composite texture formed from nine view-specific textures.



(a) Plane texture.



(b) Cube texture.

Figure 6.6: The plane and cube composite textures generated by varying the cube's position. The centre texture corresponds to the surface at truth. Note that the cube's footprint appears as a black square in the plane texture, which also illustrates how far the cube has been translated when generating the corresponding textures.

Algorithm 4: Compositing view-specific textures.

Input : $\mathcal{V} = \{\mathsf{v}_i\}$, the set of view-specific textures.
Output: t, the RGBα composite texture denoting the average reflectance from $\mathcal{V}$ and inverse-visibility (ie. weighting) of every point $\in \mathsf{t}$

1 $\mathcal{Q} = \left(\begin{bmatrix} -1 \\ -1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right)$

2 $\boldsymbol{P}_{\mathcal{G}} \leftarrow \boldsymbol{M}_{\mathcal{G}} \leftarrow \boldsymbol{I}$

/* count the number of samples per-pixel */
3 bind render-target ω
4 $\omega \leftarrow 0$
5 blend source-factor $\leftarrow$ source$_\alpha$, blend destination-factor $\leftarrow 1$
6 **for** $\mathsf{v} \in \mathcal{V}$ **do**
7 bind and modulate texture v to colour $[0, 0, 0, \|\mathcal{V}\|^{-1}]^\top$
8 render $\mathcal{Q}$
9 **endfor**

/* convert to per-pixel weights */
10 bind render-target t
11 $\ddot{\mathsf{t}} \leftarrow 0$
12 bind fragment shader to evaluate Equation 6.3
13 bind texture ω to replace fragment colour
14 render $\mathcal{Q}$

/* composit view-specific textures */
15 set α–test to cull fragments $\alpha = 0$
16 blend RGB source-factor $\leftarrow$ destination$_\alpha$, blend RGB destination-factor $\leftarrow 1$
17 blend α source-factor $\leftarrow 0$, blend α destination-factor $\leftarrow 1$
18 **for** $\mathsf{v} \in \mathcal{V}$ **do**
19 bind texture v to replace fragment colour
20 render $\mathcal{Q}$
21 **endfor**

6.3 Variable-weight composition

The composite texture is generated by giving equal consideration to all points in the view-specific texture. This is valid if images are defined over a continuous light-field and captured from a perfect pin-hole camera. Every scene-point would then project to a single, unique point in the reference image and, conversely, every observation would be equally valid because each image point would be an observation of a *single* infinitely

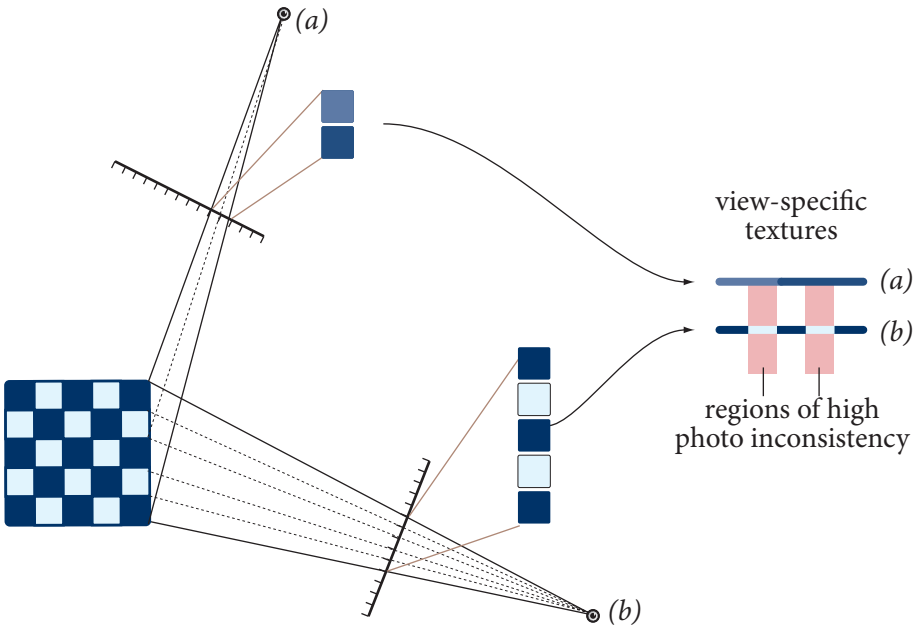


Figure 6.7: Filtering issues when sampling oblique surfaces.

small point on the reference surface. Real pin-hole cameras are not practical, however, because they require a very long exposure time and digital camera sensors can only *approximate* a continuous tricolour light-field. They typically use a Bayer mosaic sensor that samples different wavelengths across the image and interpolates the samples to yield a tricolour image.

Due to the Bayer approximation of the light-field, the colour of a single pixel is the result of many different scene-points. The back-projection of a pixel onto the surface therefore effectively a projective convolution over the surface. In cases where the surface is observed by more than one reference image with different sampling rates, giving equal consideration to these observations leads to an incorrect expected surface reflectance.

The problem arising from area-sampling is a particular concern for surfaces orthogonal to the imaging plane. Figure 6.7 illustrates the inconsistency introduced by sampling the surface at different rates where a box's striped surface is observed by two cameras. Camera *(b)'s* optical axis is almost orthogonal to the surface, but camera *(a)* is not. Given the orientation of the respective cameras, the same edge is sampled by five pixels in *(b)'s* sensor but only two pixels in *(a)*. Comparing the sampled surface from the two cameras indicates a high degree of inconsistency because the high frequency

texture cannot be accurately reconstructed from camera *(a)*'s sensor. This can lead to photo-consistency errors even for correctly registered scenes.

The artifacts caused by back-projecting a pixel over an orthogonal area can be partially addressed by favouring observations whose surface normals are parallel to the optical axis of the respective reference views. To this end, each observation is assigned a weight described by a function of the angle between the element's hypothesised normal and the view-vector. The composite texture is then given by a weighted average of each observation so that oblique samples have less effect on the estimated reflectance.

6.3.1 *Compositing variable weight view-specific textures*

Suppose there exists a weight function $w(\cdot) \mapsto \mathbb{R} \in [0,1]$ which represents the confidence that a pixel's reflectance can be attributed to a view-specific texel. In this respect, w is effectively equivalent to the image space's α-channel. Given the weight $w(\cdot)$ assigned to each texel in the view-specific texture, the composite texture is then given by

$$\mathsf{t}'(\boldsymbol{p}) = \frac{\sum_i w_i(\boldsymbol{p})\dot{\mathsf{v}}_i(\boldsymbol{p})\ddot{\mathsf{v}}_i(\boldsymbol{p})}{\sum_i w_i(\boldsymbol{p})\dot{\mathsf{v}}_i(\boldsymbol{p})}, \tag{6.10}$$

the weighted average of the corresponding unoccluded texels. This equation cannot be implemented in the blender using the same process as the fixed-weight composite given by Equation 6.4. The difficulty lies with the specific weight *per-view* rather than the fixed weight for *all* views. This necessitates using a fragment shader to modulate view-specific textures and accumulate them in floating-point buffer to compute

$$\omega'(\boldsymbol{p}) = \sum_i w_i(\boldsymbol{p})\dot{\mathsf{v}}_i(\boldsymbol{p}). \tag{6.11}$$

The results are stored in the α-channel in a process identical to computing the fixed-weight contribution from Section 6.2.1, but without scaling each contribution by n^{-1}. The view-specific textures are then modulated by the corresponding weight-map by using a fragment shader to compute

$$\mathsf{r}_i(\boldsymbol{p}) = \omega'(\boldsymbol{p})^{-1}\dot{\mathsf{v}}_i(\boldsymbol{p})\ddot{\mathsf{v}}_i(\boldsymbol{p}) \tag{6.12}$$

and accumulating the result in the frame-buffer.

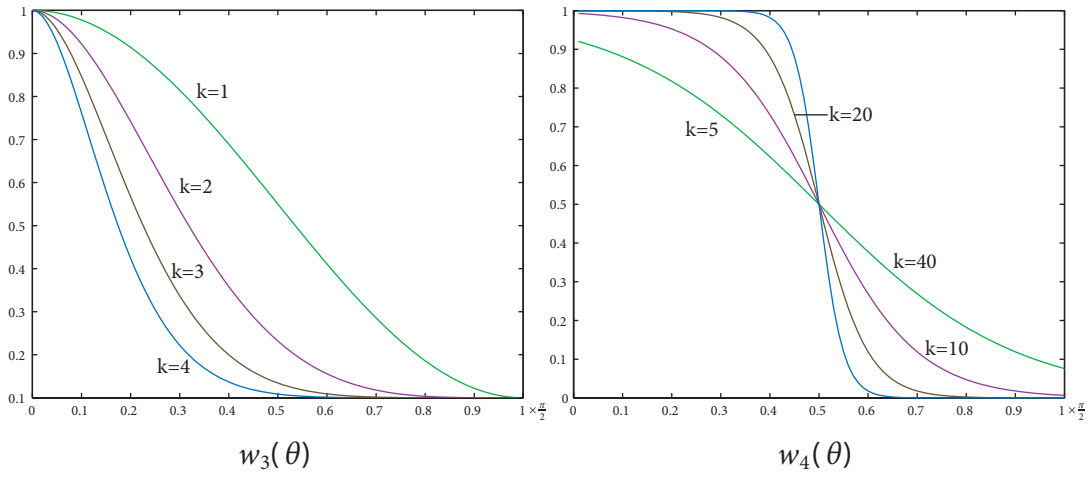


Figure 6.8: Weight functions.

6.3.2 *Weighted composite texture experiments*

The weight function considers surface points with normals parallel to the view-vector as more important than points whose normals are parallel with the imaging plane. Functions of this class are defined such that $w(\theta) = 1$ for $\theta = 0$ and approaches τ as θ approaches $\frac{\pi}{2}$, where

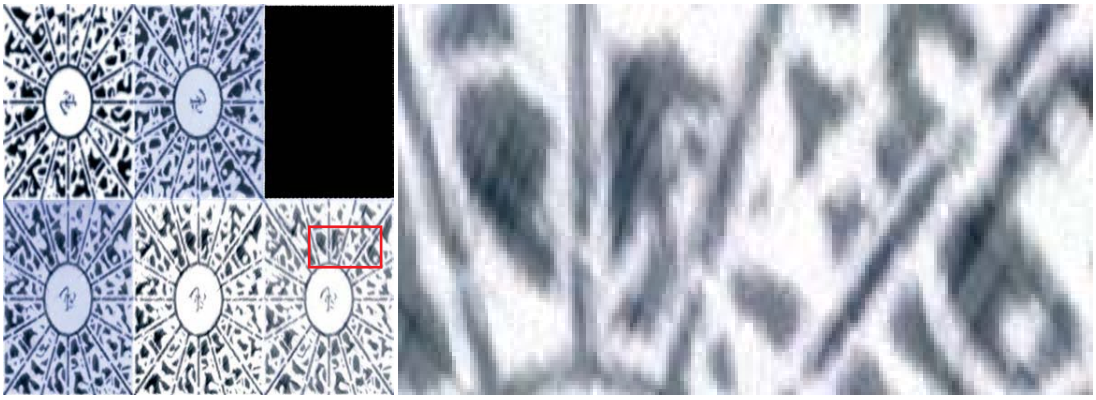
$$\cos(\theta) = \frac{\boldsymbol{v} \cdot \boldsymbol{n}}{\|\boldsymbol{v}\| \|\boldsymbol{n}\|} \tag{6.13}$$

is the angle between the normal $\boldsymbol{n}$ at the surface point $\boldsymbol{v}$ in the camera's co-ordinate frame. Four weighting functions were tested for a qualitative measurement of the composite texture:

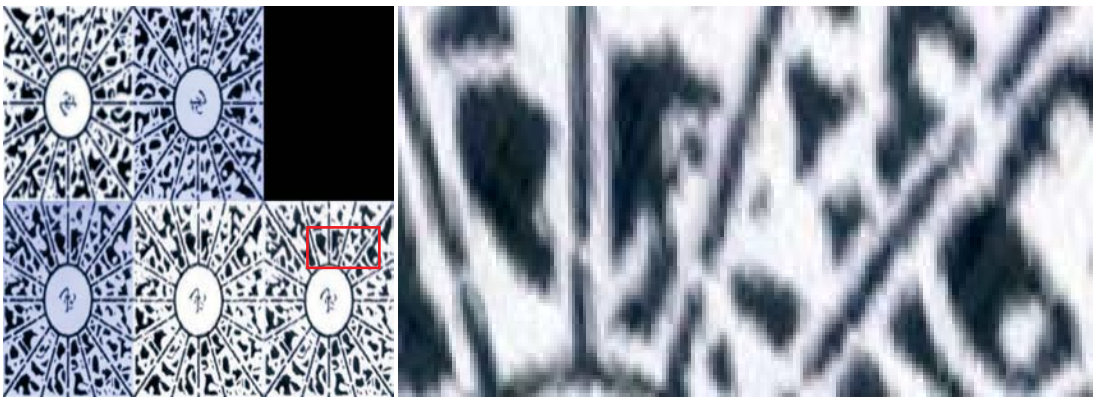
- $w_1(\theta) = 1$
- $w_2(\theta) = \frac{2\theta}{\pi}$
- $w_3(\theta) = \tau + (1 - \tau)\frac{1}{2}(1 + \cos(\pi(1 - \theta^k)))$
- $w_4(\theta) = (1 + e^{k(\frac{2\theta}{\pi} - \frac{1}{2})})^{-1}$

The functions w_3 and w_4 are parameterised by a constant k to control attenuation. Figure 6.8 illustrates different values of k for w_3 and w_4.

Figure 6.9 illustrates the difference between the fixed-weight and variable-weight textures in one particular instance using the view-specific textures illustrated in Figure 6.5. Figure 6.9(a) is derived by weighting each texel from the view-specific textures



(a) fixed



(b) variable

Figure 6.9: The juxtaposition of fixed and varying weight textures. The entire texture is on the left and an enlargement illustrating the improved image sharpness is on the right.

illustrated in by the weighting function $w_1(\cdot)$, whereas Figure 6.9(b) weights each texel by $w_3(\cdot)$ with $\tau = 0.5$ and $k = 1$. The variable weight texture is sharper because oblique views are not 'smeared' across the hypothesised surface.

Surface-Space Photo-Consistency

Chapter VII

A view-specific texture represents the estimated surface reflectance from a particular view. Each view-specific texture is *photo-consistent* with the set of observed textures if the hypothesised surface is correctly aligned with the reference surface. In the case of a perfectly diffuse surface, photo-consistency is measured by the variance of a point's observed reflectance across the set of view-specific textures. Points with a high variance are considered to be *inconsistent* with the reference images, and indicate that the hypothesised surface is incorrect. Comparing the disparity of corresponding surface points from two view-specific textures is illustrated in Figure 7.1. The presence of inconsistent points indicates that the hypothesised surface associated with these textures is not a good representation of the surface in the reference images.

Photo-consistency was used to drive the image-space likelihood function described in Chapter 4. This approach, however, was biased toward trivial solutions which occluded the hypothesised surfaces in all reference cameras. This problem motivated a surface-space likelihood that could monitor the surface's visibility in the set of reference images. This allows an occlusion likelihood that finds occluded surfaces less likely than noisy, but visible, surfaces. Our surface model, however, allows for 'intentional occlusion' by embedding primitives to construct complicated scenes. This intentional occlusion must be identified so that the likelihood is not biased toward maximising visibility at the expense of consistency.

7.1 Surface consistency likelihood

Each view-specific texture $\mathtt{v}$ is a noisy observation of the hypothesised surface reflectance $\mathtt{t}$. Given the estimated surface texture $\mathtt{t}$ and visibility image ω of the hypothesised surface $\mathcal{S}(\Psi)$, the probability of observing a pixel $\boldsymbol{p}$ in the view-specific

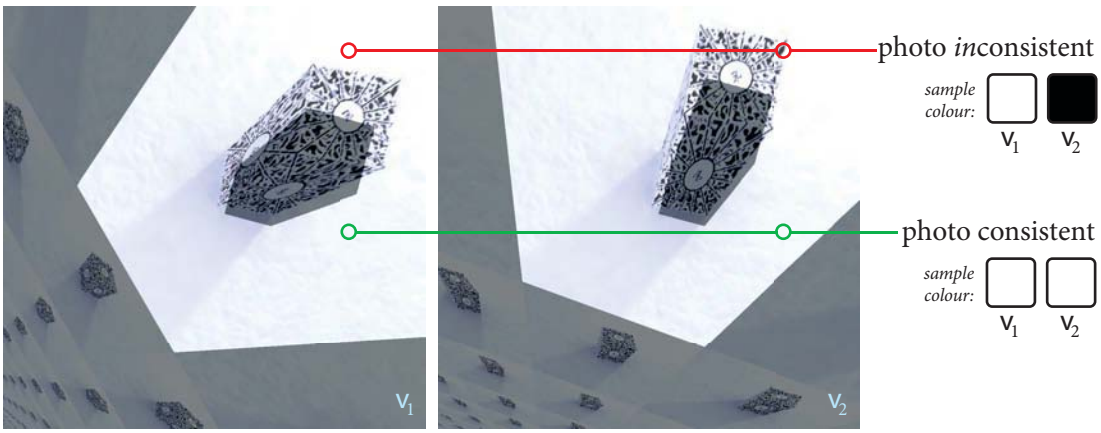


Figure 7.1: Differences in colour between corresponding texels indicates that the hypothesised surface is incorrect.

texture v_i is given by

$$\Pr(\mathsf{v}_i(\boldsymbol{p}) \mid \mathsf{t}, \omega) = \begin{cases} \Pr(\omega_i(\boldsymbol{p}) = \mathsf{o}) & \text{if } \omega(\boldsymbol{p}) = \mathsf{o} \\ \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\mathsf{v}_i(\boldsymbol{p}) - \mathsf{t}(\boldsymbol{p})\|}{4\sigma^2}\right) & \text{otherwise} \end{cases} \tag{7.1}$$

where σ is the assumed standard deviation of the sensor's Gaussian noise model and $\Pr(\omega(\boldsymbol{p}) = \mathsf{o})$ is the occlusion prior whose rôle is to discourage trivially photo-consistent solutions. The most straight-forward visibility constraint is to assume that each surface point is visible in at least k images. In this case, the occlusion prior is given by

$$\Pr(\omega(\boldsymbol{p}) = \mathsf{o}) = \begin{cases} \frac{1}{n-k} & \text{if } \omega(\boldsymbol{p}) \geq \frac{k}{n} \\ \mathsf{o} & \text{otherwise} \end{cases} \tag{7.2}$$

where n is the number of reference images. If all points in the surface texture are assumed to be independent, then the surface probability is given by

$$\Pr(\mathsf{v}_1, \mathsf{v}_2, \cdots, \mathsf{v}_n \mid \mathsf{t}, \omega) = \prod_i \prod_{\boldsymbol{p}} \Pr(\mathsf{v}_i(\boldsymbol{p}) \mid \mathsf{t}, \omega) \tag{7.3}$$

Given the projective relationship between the reference images and points on the surface described in Section 3.3.1, the likelihood of generating the images is given by

$$\mathscr{L}(\mathcal{I} \mid \Psi, \mathcal{S}) \propto \Pr(\mathsf{v}_1, \mathsf{v}_2, \cdots, \mathsf{v}_n \mid \mathsf{t}, \omega). \tag{7.4}$$

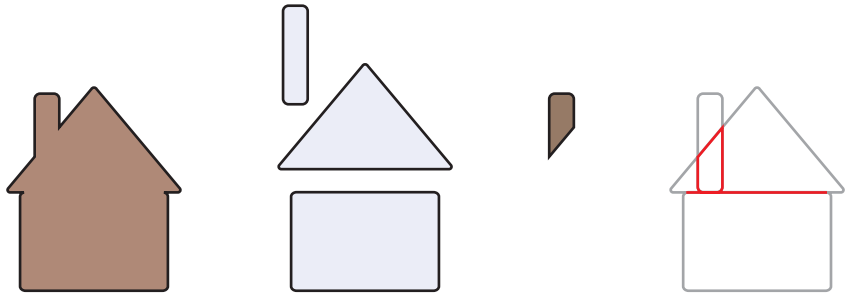


Figure 7.2: The surface model allows objects to be embedded to give the external appearance of more complicated surfaces. Embedding primitives leads to interior points, marked by red in Figure (7.2(d)).

7.1.1 The occlusion prior

The occlusion prior (7.2) assumes that all texels are visible in at least k views. While this may be true for points actually on the reference surface, the construction of the scene-graph allows primitives to be embedded to create the outward appearance of a complicated surface. Embedding one primitive into another leads to primitive occlusion *by design*. Unless these points are identified, the occlusion prior will push the reconstruction to undesirable solutions by forcing a visibility constraint on these 'latent' surfaces. An example of occlusion by design is illustrated by the house scene-graph in Figure 7.1.1. The house in this example is constructed by two boxes representing the walls and chimney and a triangle prism representing the house's roof (Figure 7.2(b)). The chimney box is embedded into the roof prism, giving the impression that the chimney primitive is modelled by the shape illustrated in Figure 7.2(c), but without having to define a new primitive.

Although adding new primitives is not difficult, the key advantage of embedding primitives in this manner is that it simplifies the description of geometric constraints. The angle of the roof, for example, is simply expressed by defining a prism to represent the entire roof. If the chimney block was not allowed to intersect or occlude any part of the roof, then the roof primitive would have to be broken into separate components within the scene-graph. Consequently, expressions along transformation links between different components would be required to ensure each section is transformed and

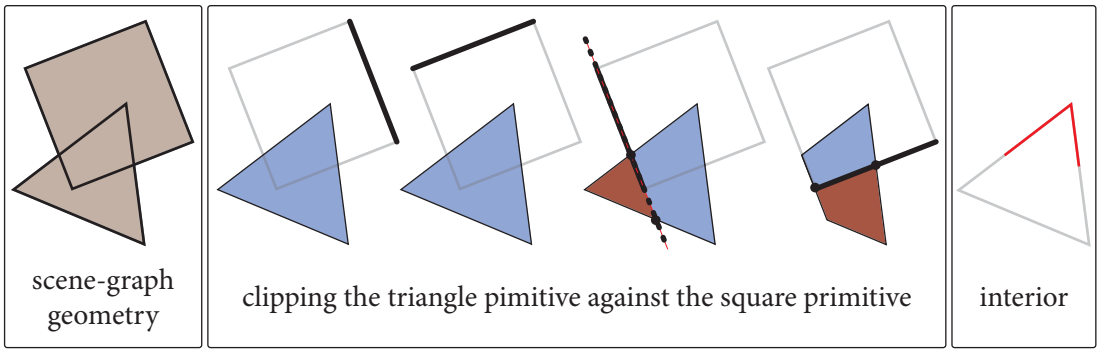


Figure 7.3: Using CSG to find interior surfaces by clipping edges of one primitive against all edges of another. The interior points are marked in red in the final frame.

orientated correctly. Not allowing embedded primitives substantially increases the complexity of the scene-graph's construction.

Embedding primitives is a convenient way of constructing relatively complicated surfaces. The occlusion prior must therefore be adapted to recognise that not all points in the scene-graph are intended to correspond with points on the reference surface. If left unchecked, the occlusion prior would adversely affect the scene by applying a visibility constraint to the scene-graph's interior points (marked by red in Figure 7.2(b)) to ensure that they, too, are visible in the reference image.

The solution is to identify these interior points so that the occlusion prior is only applied to the external surface of the hypothesised model. Assuming there exists a function $interior(\cdot)$ that computes the interior surface of the hypothesised scene-graph, the occlusion prior would then be

$$\mathscr{L}(\alpha(\boldsymbol{p}) = 0) = \begin{cases} \frac{1}{n-k} & \text{if } \omega(\boldsymbol{p}) \geq \frac{k}{n} \text{ or } \boldsymbol{p} \in interior(\mathcal{S}(\Psi)) \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

so that interior scene-points are considered as likely as exterior points that satisfy the minimum visibility constraint.

7.1.2 *Identifying interior points*

Constructive solid geometry (CSG) is one approach to identify the interior surface points of $\mathcal{S}(\Psi)$. CSG applies Boolean operations to solid primitives to create new polyhedral

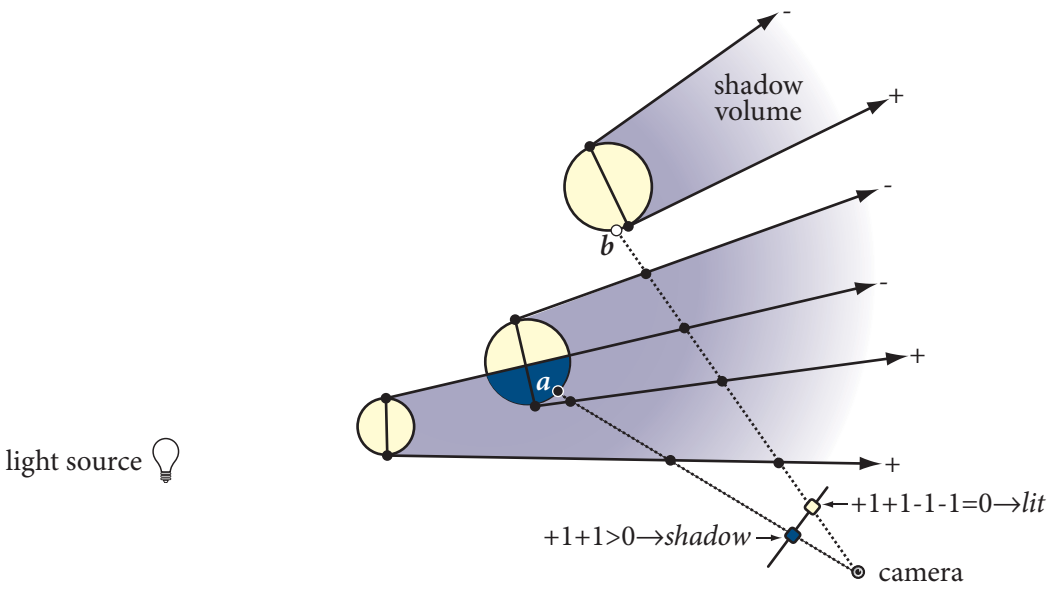


Figure 7.4: The edges of shadow volumes partition the space into lit and unlit regions.

meshes [39], and can be used to find interior surfaces with respect to two primitives by clipping each edge of one primitive against all edges of the second. This process, illustrated in Figure 7.3 and repeated for all pairs of primitives, would yield the interior points of the hypothesised scene-graph. The disadvantage of CSG, however, is that it requires a surface representation that can be decimated by clipping, and the result must be mapped back into surface space of the original primitives. These requirements break the graphics hardware model because generating a new mesh must be performed on the processor. Rather than generating an explicit mesh, our approach computes the set of interior points directly in the surface's texture-space, avoiding both limitations of the CSG approach.

The set of interior points is computed in a process not dissimilar to Crow's algorithm for rendering shadow volumes using stencil hardware [12]. Crow defines a shadow volume for each object in the scene to partition the scene into lit and unlit sub-spaces. Shadow volumes are defined by back-projecting each object's silhouette with respect to a point light-source. Determining whether a scene point is in a shadow volume involves stepping along the ray to the camera's optical centre and counting the number times the ray crosses the volume's opening and closing faces. This process is illustrated in Figure 7.4, where ***a***, for example, is in shadow because the ray from ***a*** to

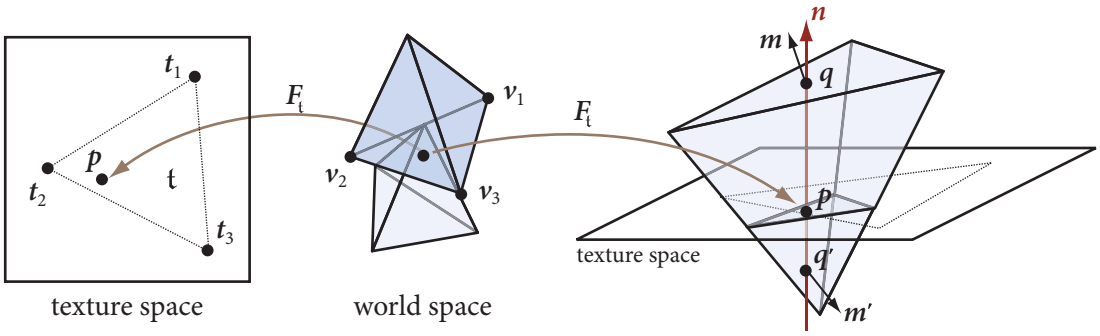


Figure 7.5: The world space is mapped to texture space for intersection testing.

the virtual camera's optical centre enters two shadow volumes without leaving either, whereas $\boldsymbol{b}$ is lit because it enters and leaves both volumes.

Heidmann renders shadows using graphics hardware's support for stencil buffers [24]. The scene is first rendered from the camera's point-of-view to generate the distance to each visible scene-point. Scene points are tested for inclusion in the shadow volume by rendering the edges of the shadow volume while using the stencil buffer to find which edges fail the depth-test. As the shadow is rendered, the stencil buffer is incremented by front-facing fragments and decremented by back-facing fragments. The result is a stencil mask where a stencil value of zero indicates the pixel is lit and non-zero indicates it is in shadow.

We use the stencil buffer to similarly determine the intersection of a primitive's surface against the entire hypothesised scene. Instead of rendering the scene from a new view, the surface is orthogonally projected onto every texture triangle while the orientation of the surface is used to track which parts of the triangle are enclosed by the scene. The complexity of this algorithm is, unfortunately, squared in the number of triangles because the scene must be rendered once *for every* triangle in the scene-graph.

7.1.3 Intersection in graphics hardware

The hypothesised surface $\mathcal{S}(\Psi)$ can be partitioned into interior and exterior (ie. visible) surfaces. If the classification of a surface point is represented by an 'interior' texture-

image i defined by

$$\mathsf{i}(\boldsymbol{p}) = \begin{cases} 1 & \text{if } \exists \mathcal{P} \text{ s.t. } \mathit{interior}_{\mathcal{P}}(\boldsymbol{G}_{\mathfrak{t}}^{-1}\boldsymbol{p}) \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

where $\mathfrak{t}$ is the triangle that encloses the given point $\boldsymbol{p}$ and $\mathcal{P} \in \mathcal{S}(\Psi)$ is a primitive transformed into world co-ordinates. The homography $\boldsymbol{G}_{\mathfrak{t}}^{-1}$ relates a triangle's world and texture space (from (5.3)).

The function $\mathit{interior}_{\mathcal{P}}(\boldsymbol{p})$ determines if $\boldsymbol{p}$ is enclosed within $\mathcal{P}$ by finding the set of points on $\mathcal{P}$ co-incident with a vector through $\boldsymbol{p}$. If $\mathcal{P}$ is assumed not to be self-intersecting, then the orientation of the boundary points with respect to the vector through $\boldsymbol{p}$ can be used to determine the negative semi-space of the boundary and count $\boldsymbol{p}$'s entry and exit into $\mathcal{P}$'s interior. This process is simplified by mapping $\mathcal{P}$ into $\boldsymbol{p}$'s texture space with the homography

$$\boldsymbol{F}_{\mathfrak{t}} = \begin{bmatrix} \underline{\boldsymbol{t}}_1 & \underline{\boldsymbol{t}}_2 & \underline{\boldsymbol{t}}_3 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^\top \\ \boldsymbol{v}_2^\top \\ \boldsymbol{v}_3^\top \\ \{\boldsymbol{v}_1 + (\boldsymbol{v}_3 - \boldsymbol{v}_1) \times (\boldsymbol{v}_2 - \boldsymbol{v}_1)\}^\top \end{bmatrix}^{-\top} \tag{7.7}$$

where $\boldsymbol{p} \in \mathfrak{t} = \{\underline{\boldsymbol{t}}_1, \underline{\boldsymbol{t}}_2, \underline{\boldsymbol{t}}_3\}$ is the texture triangle enclosing $\boldsymbol{p}$ and $\mathfrak{w} = \{\underline{\boldsymbol{v}}_1, \underline{\boldsymbol{v}}_2, \underline{\boldsymbol{v}}_3\}$ is the corresponding world triangle.

The mapping described by (7.7) is illustrated in Figure 7.5. The homography $\boldsymbol{F}_{\mathfrak{t}}$ transforms the surface so that the world triangle $\{\boldsymbol{v}_i\}$ is mapped to the corresponding texture triangle $\{\boldsymbol{t}_i\}$. The points $\boldsymbol{q}$ and $\boldsymbol{q}'$ along the vector $\boldsymbol{n}$ are readily identified because all points on $\boldsymbol{n}$ project to $\boldsymbol{p}$. The orientation of the boundary points defines the negative semi-space of the surface: a point is classified as interior if it enters more negative semi-spaces than it leaves. This is determined by choosing a vector from $\boldsymbol{p}$ and counting the number of negative and positive subspaces crossing the vector. Given the boundary classification $s_{\underline{\boldsymbol{p}}}(\underline{\boldsymbol{q}})$ that returns the sign indicating the half-space with respect to $\boldsymbol{p}$, then $\boldsymbol{p}$ is interior if

$$\mathit{interior}_{\mathcal{P}}(\underline{\boldsymbol{p}}) = \left(\sum_{\underline{\boldsymbol{q}} \in \mathcal{P}} s_{\underline{\boldsymbol{p}}}(\boldsymbol{F}_{\mathfrak{t}}\underline{\boldsymbol{q}}) \neq 0 \right). \tag{7.8}$$

The interior classification assumes negative semi-spaces are bound by $\mathcal{P}$. Planes

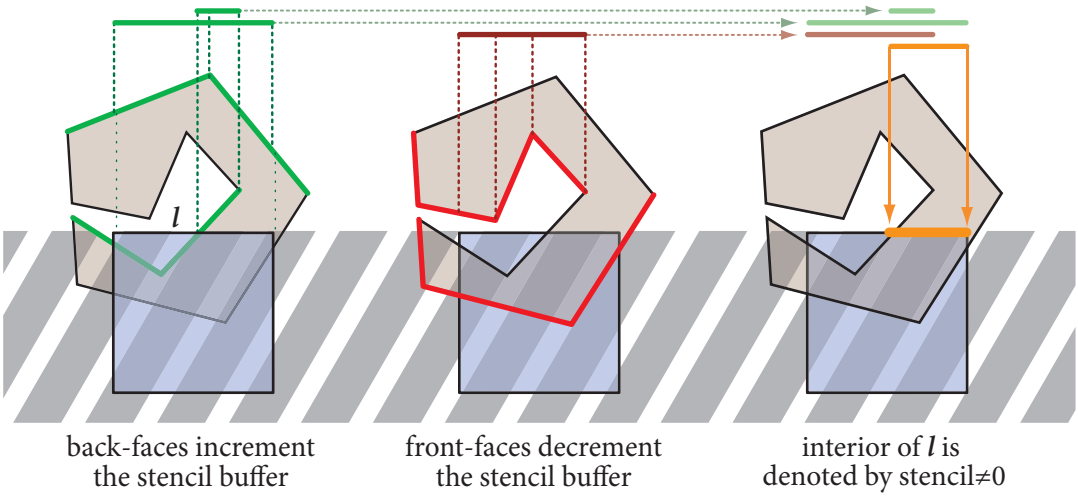


Figure 7.6: Using stencil buffers to compute surface intersection.

do not have a defined interior, however, and therefore effectively partition space into an infinite negative semi-space. Because our system uses back-face culling, a plane entity is actually two coplanar quadrilaterals with opposite normals. Points in the scene-graph can also be occluded if they are coplanar, however. To identify these points, we intentionally 'uncap' the surface volume by ignoring primitive closures within ϵ distance to $\boldsymbol{p}$. If the hemi-space of $\boldsymbol{p}$ with normal $\boldsymbol{n}$ with respect to a boundary point $\boldsymbol{q}$ with normal $\boldsymbol{m}$ is indicated by

$$h = \operatorname{sign}\left(\frac{\pi}{2} - \cos^{-1}(\boldsymbol{n} \cdot \boldsymbol{m})\right) \tag{7.9}$$

then the boundary classification

$$s_{\underline{p}}(\underline{q}) = \begin{cases} 0 & \text{if } \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \boldsymbol{F}_t \underline{q} > \frac{\epsilon}{2}\operatorname{sign}(h+1) \text{ or } \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \underline{q} \neq \underline{p} \\ h & \text{otherwise} \end{cases} . \tag{7.10}$$

will count boundary entry and exit points on one side of $\boldsymbol{p}$, but open interiors if the closure is within ϵ of $\boldsymbol{p}$.

The function $s(\cdot)$ from (7.10) can be implemented with stencil testing. Transforming the surface by $\boldsymbol{F}_t$ will align candidate boundary points in texture space if $\boldsymbol{P}_{\mathcal{G}}$ is configured to orthogonally project texture triangles into the surface texture space. The scene is then rendered twice with different stencil modifiers for front and back faces,

Algorithm 5: Identifying interior texels.

Input : the set of surface triangles $\{\{\mathfrak{t}, \mathfrak{w}\}_i^j\}$ where $\mathfrak{t}_i^j$ is the i^{th} texture triangle associated with texture j and $\mathfrak{w}_i^j$ its corresponding triangle in world space.

Output: The image $\mathfrak{i}_j$ is marked in the frame-buffer such that $\mathfrak{i}_j(\boldsymbol{p}) \neq 0$ indicates $\boldsymbol{p}$ is interior.

$\boldsymbol{P}_{\mathcal{G}} \leftarrow$ orthogonal projection to texture-space
disable depth testing and writing to the colour and depth buffer
stencil $\leftarrow 0$
for i **do**
    define and enable clipping planes for $\mathfrak{t}_i^j$
    enable stencil testing and set it to always pass
    set stencil function to *decrement*
    enable back-face culling
    render $\mathfrak{w}_k^j \forall k \neq i$
    set stencil function to *increment*
    enable front-face culling
    define and enable clipping plane $z = -\epsilon$
    render $\mathfrak{w}_k^j \forall k \neq i$
    disable all clipping planes
endfor

leaving the stencil buffer in a state where a stencil value of 0 indicates the corresponding point is not interior. The algorithm is illustrated Figure 7.6 and listed in Algorithm 5.

7.1.4 Computing the surface-space likelihood

A texel in a view-specific texture can be in one of three states: it is either visible with an hypothesised colour; it is inside the hypothesised shape; or it is on the surface but is occluded in the particular reference image. A likelihood map is defined for each view-specific texture, where the likelihood of each texel is given by (7.1). The probabilities from each texel in all views are combined by (7.3) to yield the surface's likelihood while taking occlusion into account.

Figures 7.7 and 7.8 illustrates the consistency of a hypothesised model from eight synthetic images of a house. Six different configurations are illustrated; each configuration is rendered using two of the reference camera's parameters. The three

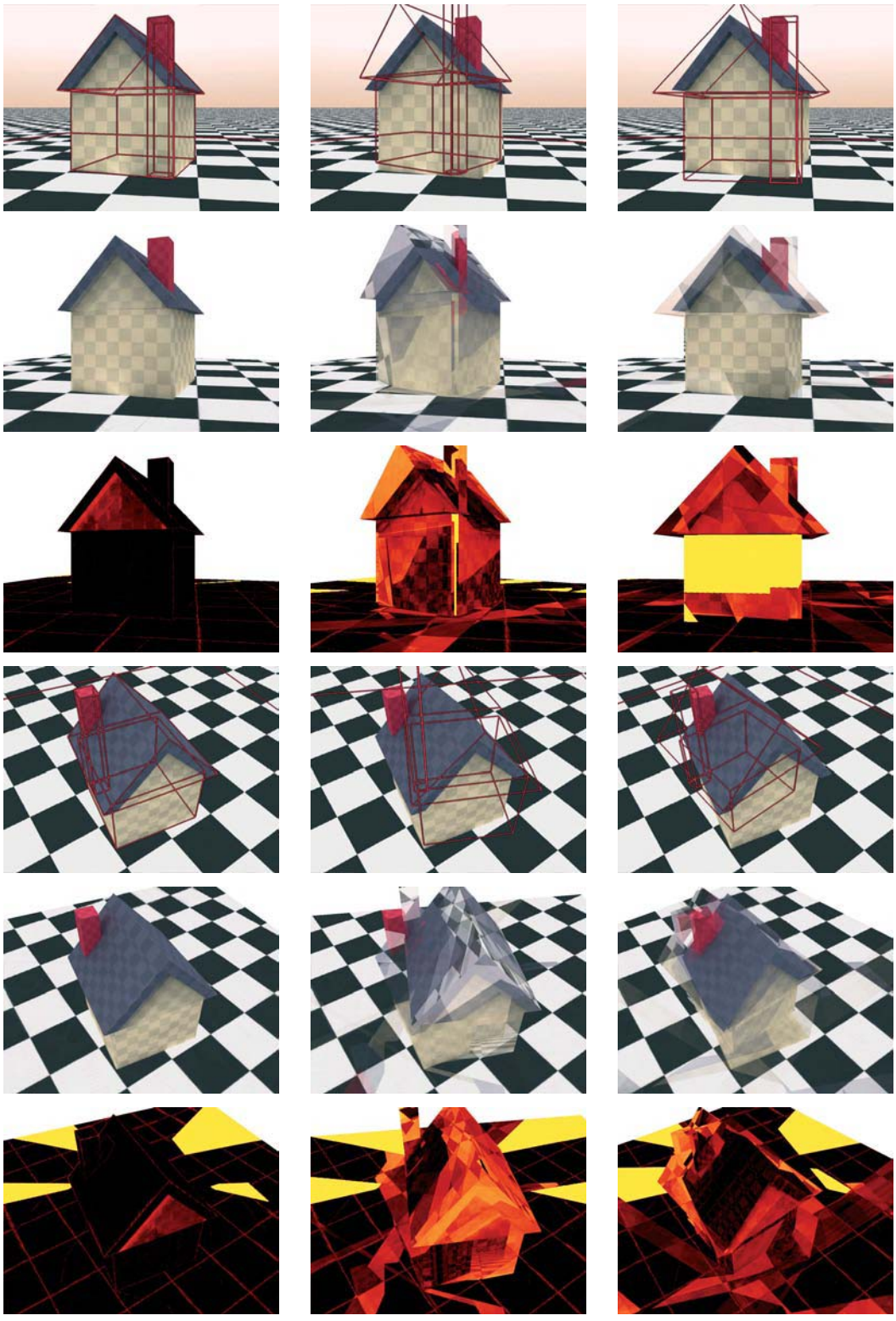


Figure 7.7: Photo-consistency from hypothesised surfaces reprojected into two views *(1)*.

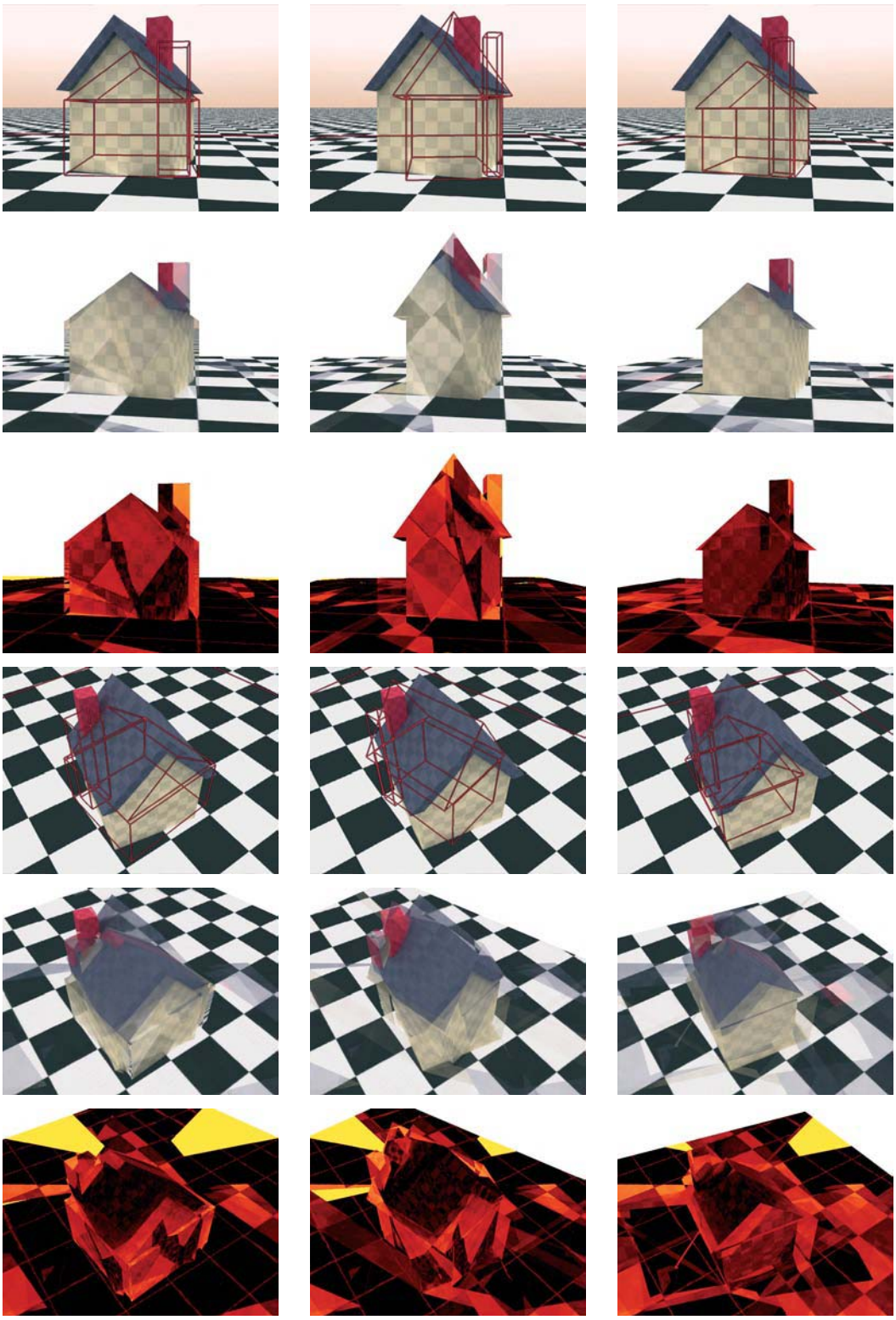


Figure 7.8: Photo-consistency from hypothesised surfaces reprojected into two views *(2)*.

images associated with each view-point illustrate different properties of the hypothesised surface:

- the wire-frame model is super-imposed with the reference images in rows 1 and 4;
- the model with composite texture is illustrated in rows 2 and 5; and
- the surface's consistency is visualised in rows 3 and 6.

The configuration at truth is illustrated in Figure 7.7(a). It is evident from the consistency frames (the third and sixth images) that the model parameters corresponding to ground-truth causes a degree of surface inconsistency on the house model, and large regions of inconsistency on the ground-plane. Despite this apparent inconsistency, however, this solution is clearly more consistent than the other illustrated examples. The inconsistency on the house is caused by using a triangular prism to represent the entire roof, despite tiles extending beyond the walls. This causes the short sides of the prism to also extend beyond the walls, leading to slight inconsistency from mis-matched parallax.

The visibility prior's effect is evident on both the house model and ground-plane. The ground-plane inconsistency in all reconstructions from Figures 7.7 and 7.8 means that the hypothesised surface requires a smaller ground-plane to ensure that every point on the surface is visible in two views. The visibility prior's effect is also evident on the house model in some of the reconstructions. In particular, the side of the house in the third frame of the hypothesis illustrated in Figure 7.7(c) is considered unlikely because the walls are occluded by the overhanging roof in too many reference images.

7.2 Experiments

The surface-space likelihood was tested on a similar sequence used for the experiments in Section 4.5. The sequence from Section 4.5 could not be used with the surface-space likelihood because significant parts of the scene is visible in only one camera. The new image sequence of a cube upon a ground-plane is illustrated in Figure 7.9. The majority of the scene-points are visible in at least two reference images, with the exception of some points near the base of the cube which are visible by only one camera. Although this is not occlusion as an artifact of the scene-graph, the tests demonstrate that this occlusion does not adversely affect the likelihood.

Five tests that varied a subset of the scene parameters while keeping the others at truth were conducted to compare the behaviour of the surface-space and image-space likelihoods. Like the graphs from Section 4.5, the true values are in the centre of the graphs. These tests include:

- *Camera optical-centre—Figure 7.10*. Both likelihoods are maximised at truth.
- *Camera focal-length—Figure 7.11*. Both graphs have a global maximum at truth. Longer focal lengths increasingly occlude the hypothesised scene. This, in turn, leads to less photo-consistency error in the image-space likelihood, but less likely surfaces from the surface-space likelihood as the visibility constraint is applied.
- *Cube scale—Figure 7.12*. The image-space likelihood demonstrates the same occlusion bias as the test in Figure 4.11(c): although the likelihood has a clear local maxima, the global maximum is *not* at truth. In contrast, the surface-space likelihood has a global maximum at truth because the visibility constraint penalises configurations that occlude the surface.
- *Cube translation—Figure 7.13*. While both likelihoods are maximised at truth, the surface-space likelihood has more a pronounced 'cross' because half of the likelihood is derived from each of the two primitives. In contrast, the inconsistency attributed to the cube in the image-space likelihood is a function of the cube's footprint in each image which, in turn, is a function of the cube's translation.
- *Cube rotation—Figure 7.14*. Both likelihoods are maximised at truth.

Figure 7.9: Nine synthetic cube/plane reference images.

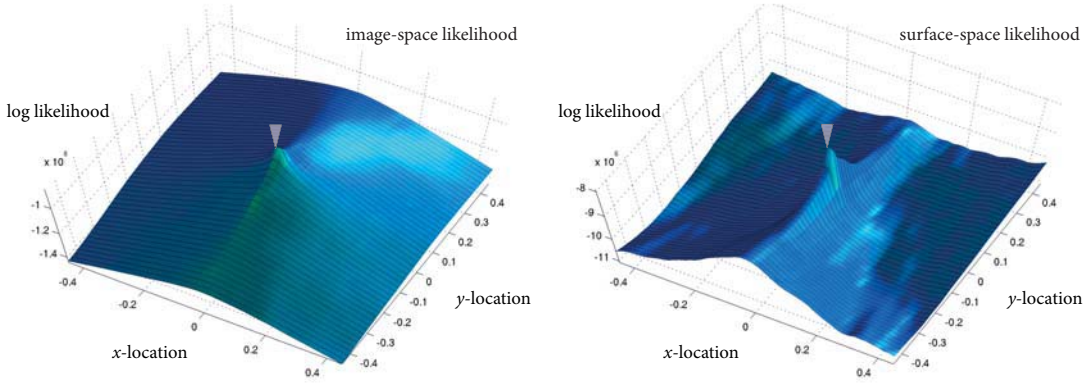


Figure 7.10: Synthetic optical centre likelihood.

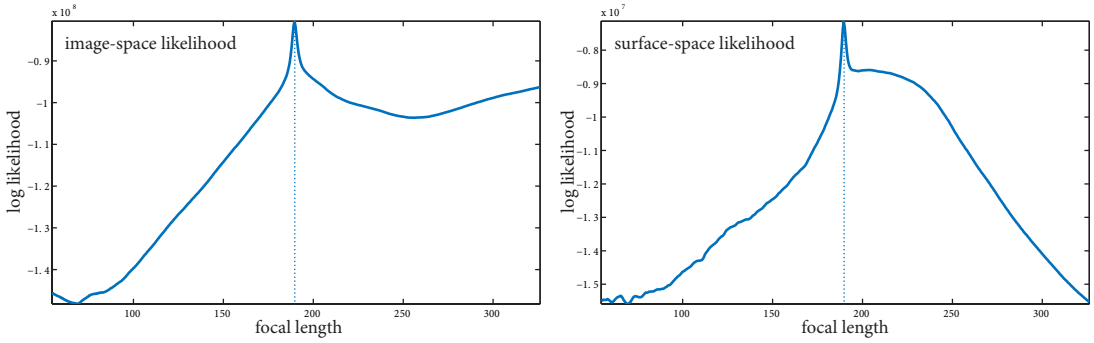


Figure 7.11: Synthetic focal length likelihood.

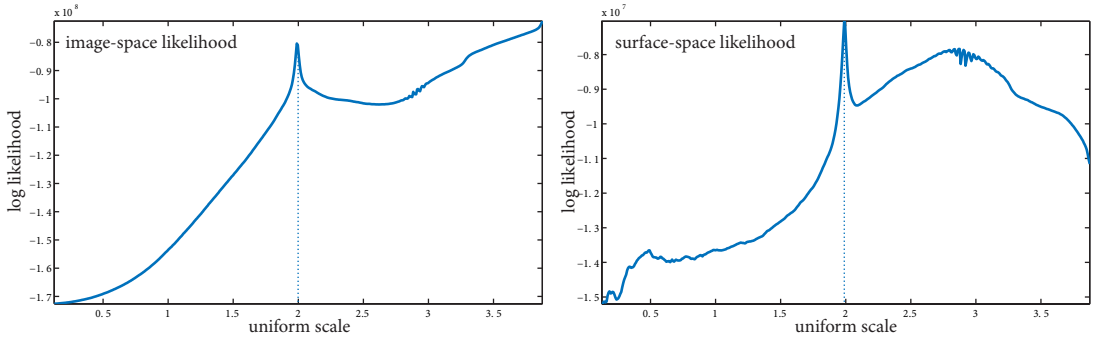


Figure 7.12: Synthetic scale likelihood.

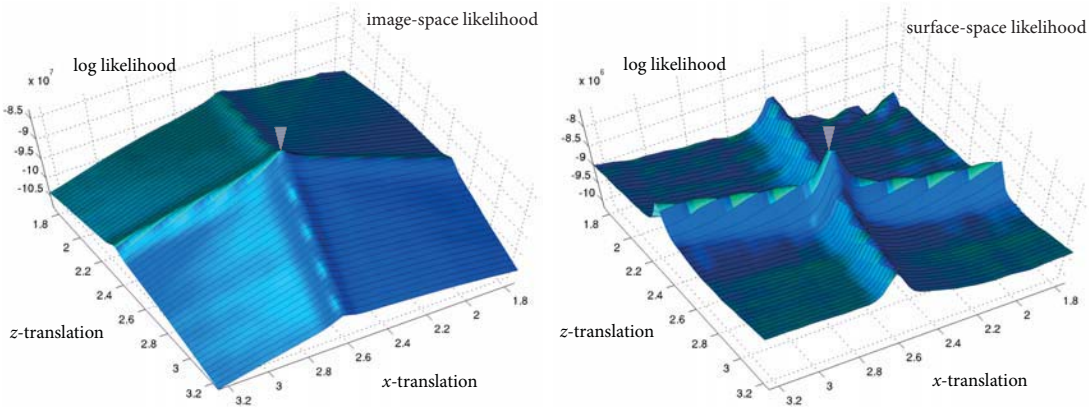


Figure 7.13: Synthetic translation likelihood.

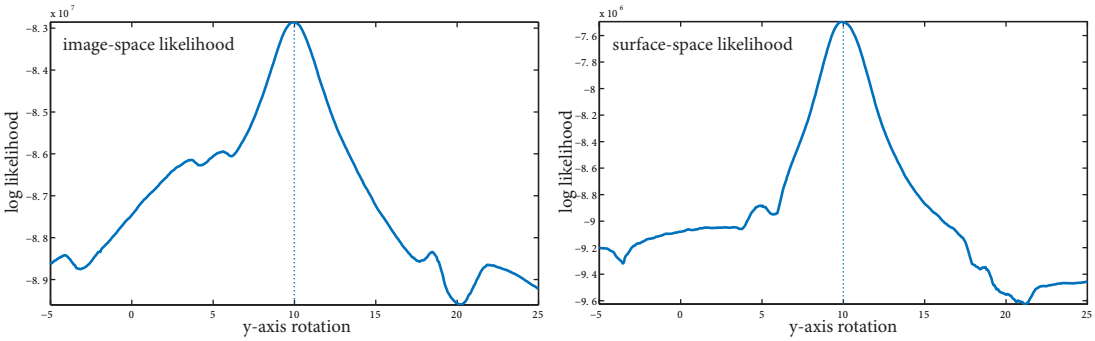


Figure 7.14: Synthetic rotation likelihood.

7.2.1 Synthetic stair test

We tested the surface-space likelihood's behaviour against a synthetically generated scene illustrated in Figure 7.15. The surface consists of a cube on a set of stairs, illustrating an example of a pattern constraint. The scene-graph is usefully constrained by recognising that each step is the same height and the top step's depth is uniform around its perimeter. The stairs were defined using ten quadrilaterals in a scene-graph defined by seven parameters to control the stair's position, orientation, scale and depth (ie. the top step's offset). The cube's position, scale and orientation relative to the stair's corner was unknown, but the scene-graph arrangement ensured the cube was always on top of the step.

A simulated annealing chain of length 50,000 states converged to the MAP estimate illustrated in Figures 7.16 and 7.19. Figure 7.17 is a visualisation of the chain's random walk, where each frame was generated by rendering the previous 5000 hypotheses, demonstrating that the distribution cools over time. Figure 7.19 illustrates a number of synthetic images of the solution. These images were generated by mapping the composite texture onto the reconstructed surface and imaging the result from novel virtual cameras.

Not all MCMC chains converged to the solution illustrated in Figure 7.19; Figure 7.18 illustrates the re-projections of the top three unique solutions. Interestingly, the two solutions not at truth (*b* and *c*) successfully reconstructed the cube and therefore correctly estimated the stairs height, but were less successful at reconstructing the base of the stairs. The robustness of the cube reconstruction can be attributed to its texture being significantly different from the stairs. Since the stair's height also controls the cube's projection in the images, the vertical scale of the stairs is well defined despite difficulty in estimating the stair's base due to the largely homogeneous stair texture.

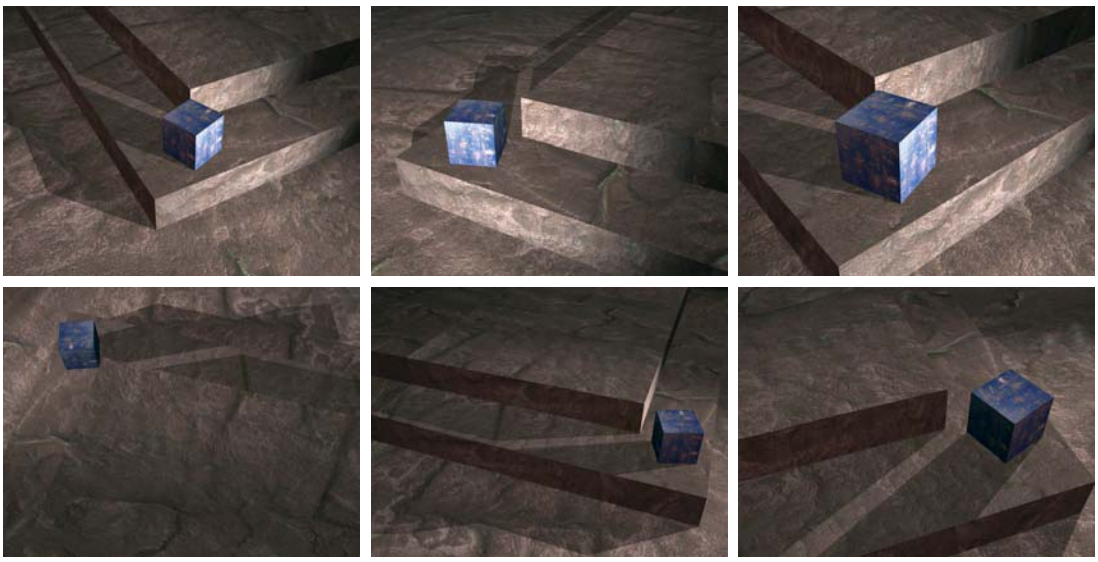
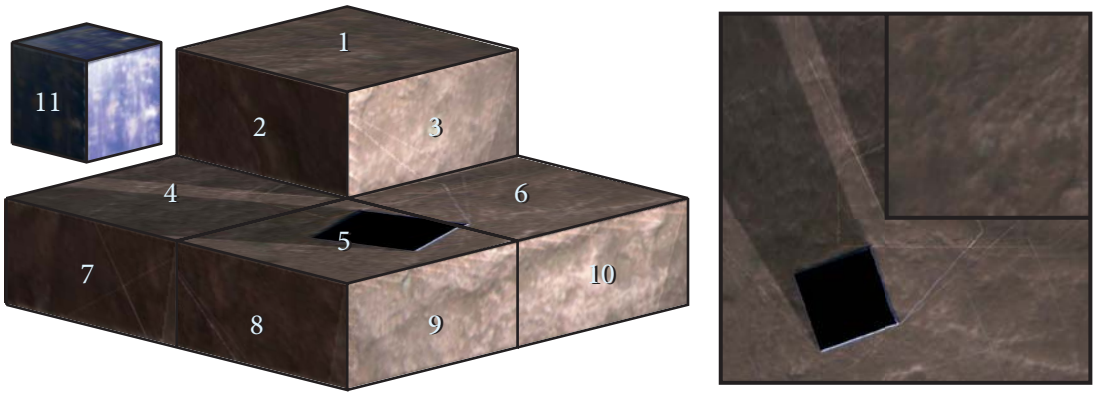
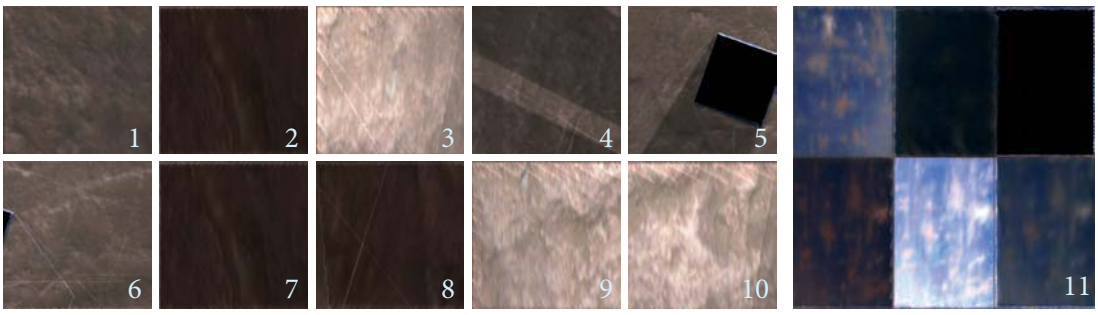


Figure 7.15: The six synthetic reference images for the stairs test.



(a) Orthogonal and overhead re-projection of the textured stair model.



(b) Stair textures.

Figure 7.16: The stair test's composite textures. The orthogonal projections are only to illustrate the texture arrangement–its proportions are approximate.

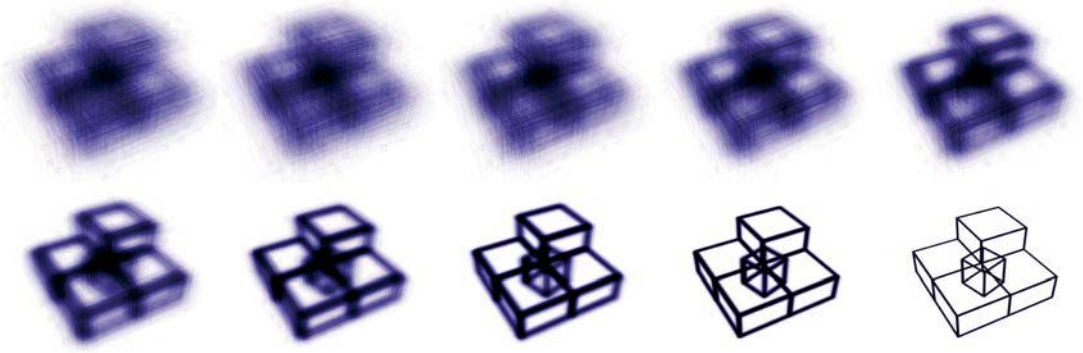


Figure 7.17: An annealed MCMC random walk visualisation for the stairs test.

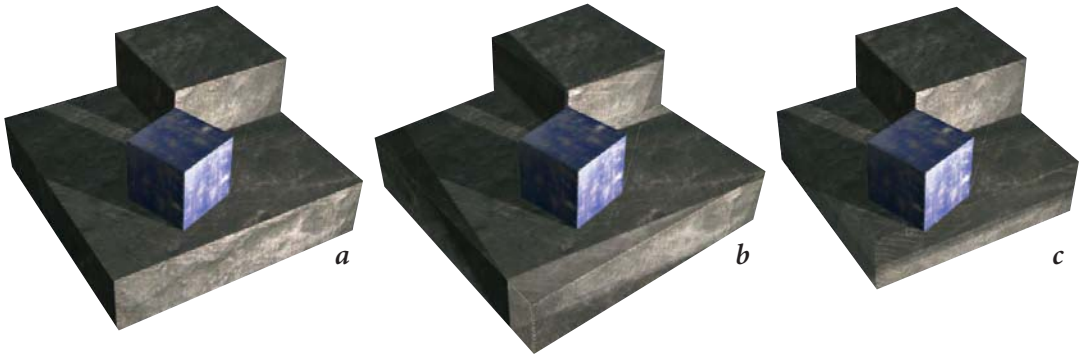


Figure 7.18: Three stair reconstructions.

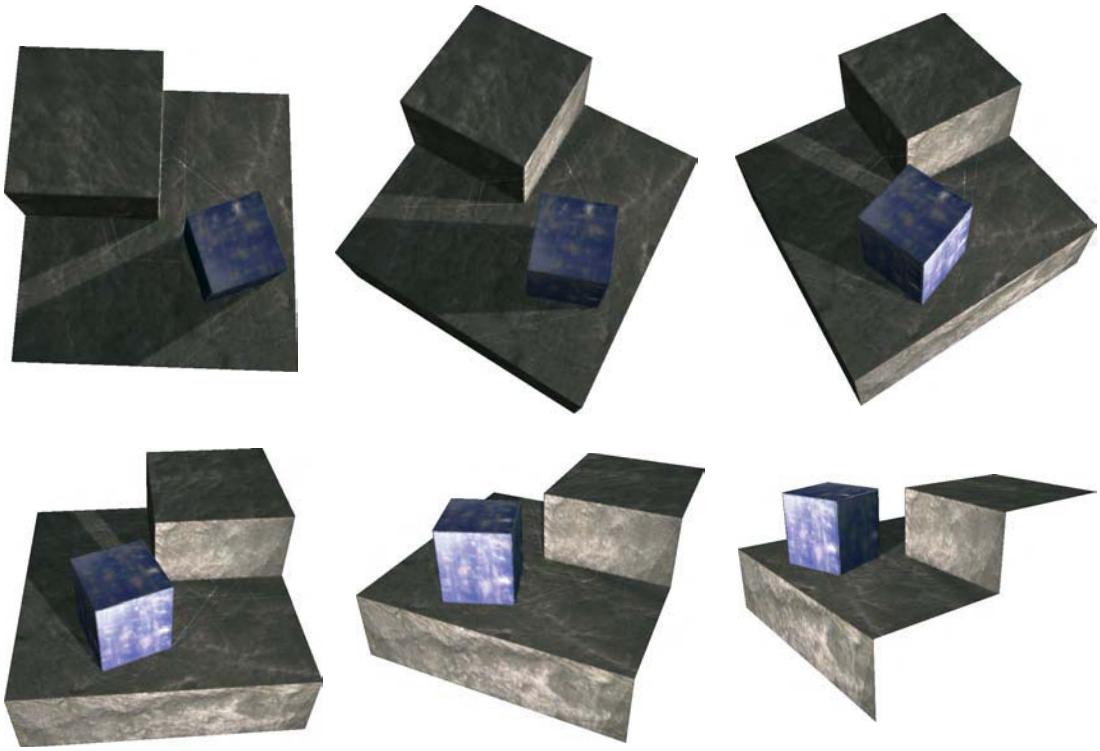


Figure 7.19: Synthetic views of the stair test reconstruction.

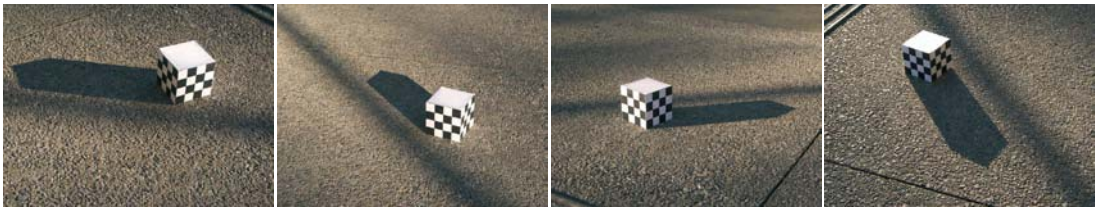


Figure 7.20: Reference images for the second cardboard box test.

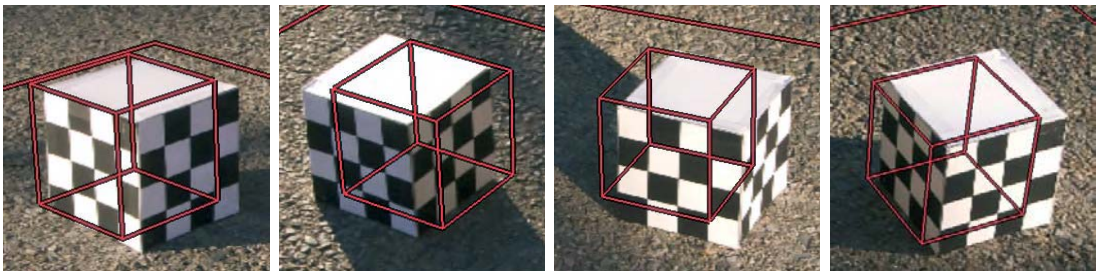
7.2.2 Cardboard box test revisited

The experiment from Section 4.5.3 was repeated with the four image sequence illustrated in Figure 7.20. These reference images are similar to the sequence in Figure 4.23, but with an important difference in lighting conditions. Although both sequences were taken from the same area, the sequence used in Chapter 4 was captured when the sun was diffused through clouds. The sequence in Figure 7.20, in contrast, was captured in bright, directional sun-light when the sun was low in the sky. These lighting conditions were deliberately chosen because the change in light has a significant effect on the photo-consistency likelihood.

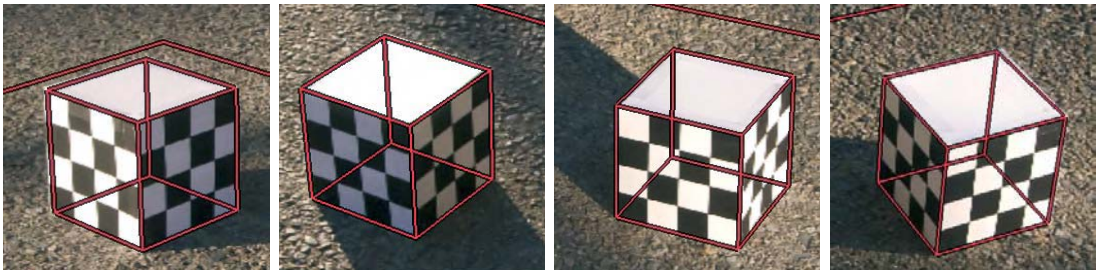
The aim of this experiment is to recover camera parameters relative to the cardboard box. The cameras were initially calibrated using Tsai's camera calibration [67] relative to manually identified correspondences between a 3D point cloud and the reference images. The world co-ordinate system was defined by this 3D point cloud; each camera was calibrated with respect to the common co-ordinate frame. Accordingly, the scene-graph was defined as a static cube on a ground plane but with variable camera parameters. The priors on the camera parameters were relative to the values estimated by the Tsai calibration:

- $\Pr(\boldsymbol{t}) = \mathcal{U}(-3, 3)$—the translation prior is relative to the size of the cube, which is defined by the points $[\pm 1, 1 \pm 1, \pm 1]^\top$;
- $\Pr(\theta) = \mathcal{U}(-5, 5)$—the rotation prior (in degrees) about the camera's three local axes;
- $\Pr(f) = \mathcal{U}(0.85, 1.15)$—focal length as a multiple of the length estimated by Tsai;
- $\Pr(\boldsymbol{o}) = \mathcal{U}(-0.1, 0.1)$—the optical centre[1] offset from the Tsai estimation.

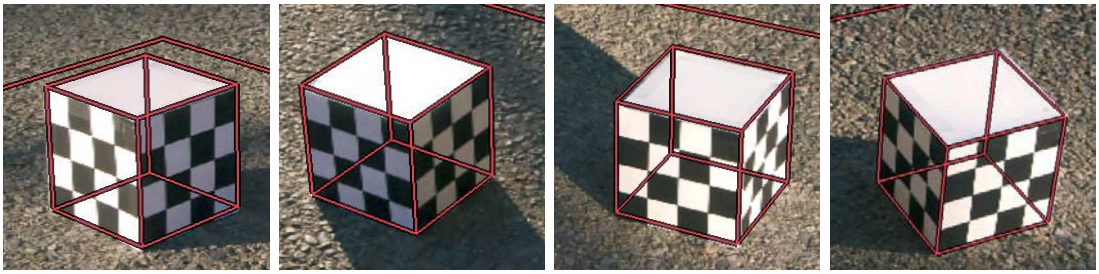
[1] The camera co-ordinate system is described in Appendix B



(a) Variable weight composite texture with the ground plane.



(b) Variable weight composite texture *without* the ground plane.



(c) Fixed weight composite texture with the ground plane.

Figure 7.21: The re-projection of the MAP estimates from the three different cardboard box tests.

Beginning from the initial calibration, the cameras were optimised by a simulated annealed MCMC chain over 10,000 iterations using the surface-likelihood. Each view-specific texture in this test was weighted by w_3 from Section 6.3.2 with $k = 1$ and $\tau = 0.5$. The MAP estimate for the variable-weight surface-likelihood is illustrated in Figure 7.21(a). The surface-space likelihood in this instance has clearly been unable to find a solution that aligns the static model with the corresponding edges in the reference images.

This result suggests that the surface-space likelihood is no better than the image-space likelihood, which also was unable to align the cameras with the reference images. Inspection of the reference images, however, suggests that the surface-likelihood is affected by the ground-plane's reflectance model. Figure 7.22 illustrates the luminance

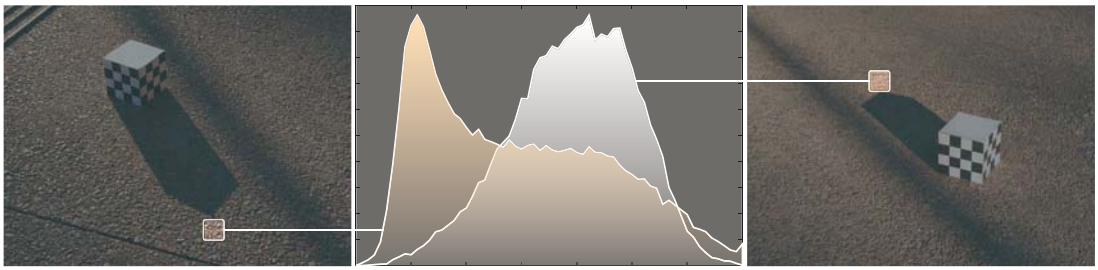


Figure 7.22: The luminance histogram of a corresponding 128 × 128 pixel region from two views.

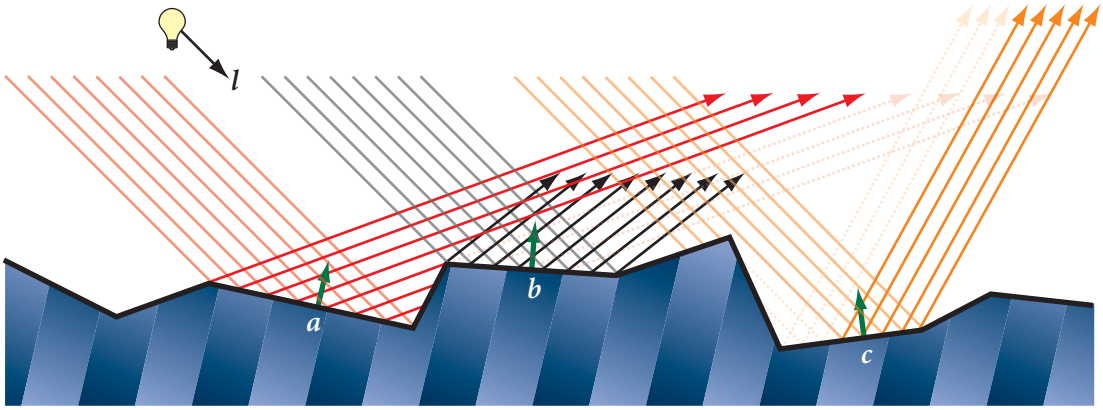


Figure 7.23: Although micro-facet surfaces may be perfectly diffuse, they may *appear* to be specular due to occlusion in both the eye- and light-directions.

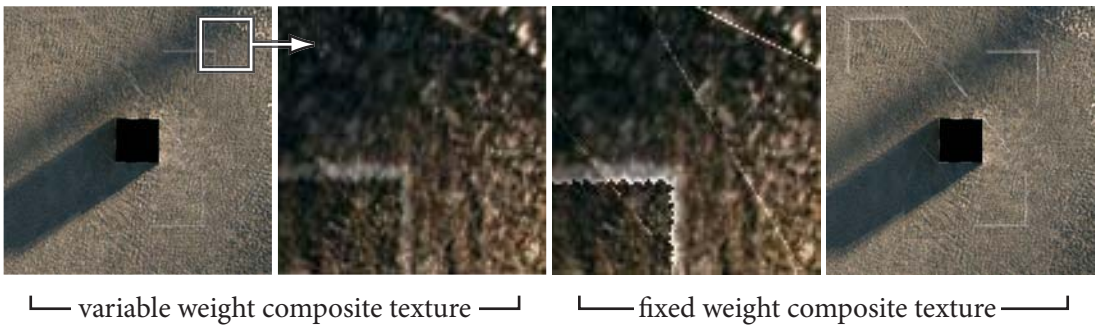


Figure 7.24: A juxtaposition of fixed- and variable-weight composite textures. The contrast of the centre two enlargements has been increased.

histograms of two approximately corresponding ground-plane regions in opposite views. The histograms in Figure 7.22 are clearly dissimilar, suggesting that the ground plane is not diffuse.

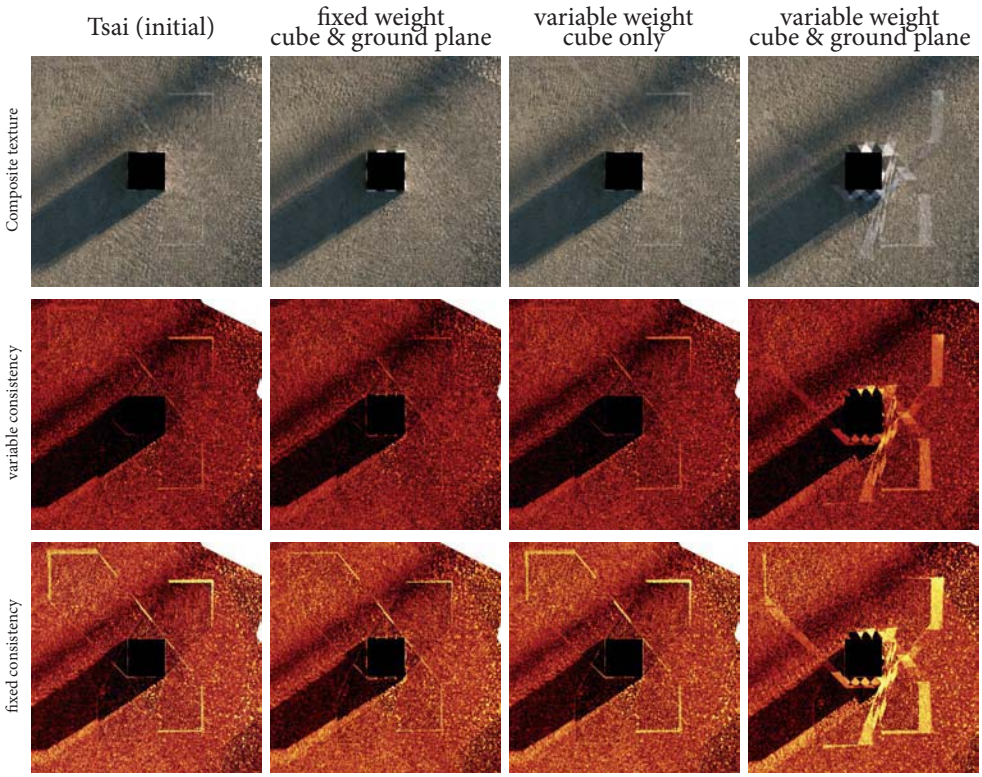
Although the ground-plane material *is* diffuse, micro-facets [5] in its texture change cause a change in reflection depending on the view direction. This reflectance model is illustrated in 7.23, where small features of the surface cause occlusion in both the eye direction (surface '*a*') and the light direction (surface '*c*'). The implication of imposing a plane model over a diffuse, but micro-faceted surface, means that the occlusion in both light and eye directions is not modelled, and, therefore, points sampled over the 'plane' appear to be specular. Furthermore, the facets are incorrectly transformed into texture space because a plane model is imposed on the ground. This property would cause inconsistency even if lighting conditions were different and the facets were sufficiently large in the image. As the ground texture cannot be represented by the assumed reflectance model while imposing a plane-constraint, photo-inconsistency was minimised by adjusting the cameras to reduce the area which is visible in each camera.

As the ground cannot be modelled as plane in these lighting conditions, a second experiment was conducted by removing the ground-plane from the surface model so that only the box's photo-consistency was used to define the surface-space likelihood. The MAP estimate in this case is shown in Figure 7.21(b)[2]. The cameras are correctly aligned with the cube in this case, demonstrating that the surface-space likelihood does not require the ground-plane to generate hypothesised point-correspondences, unlike the image-space likelihood.

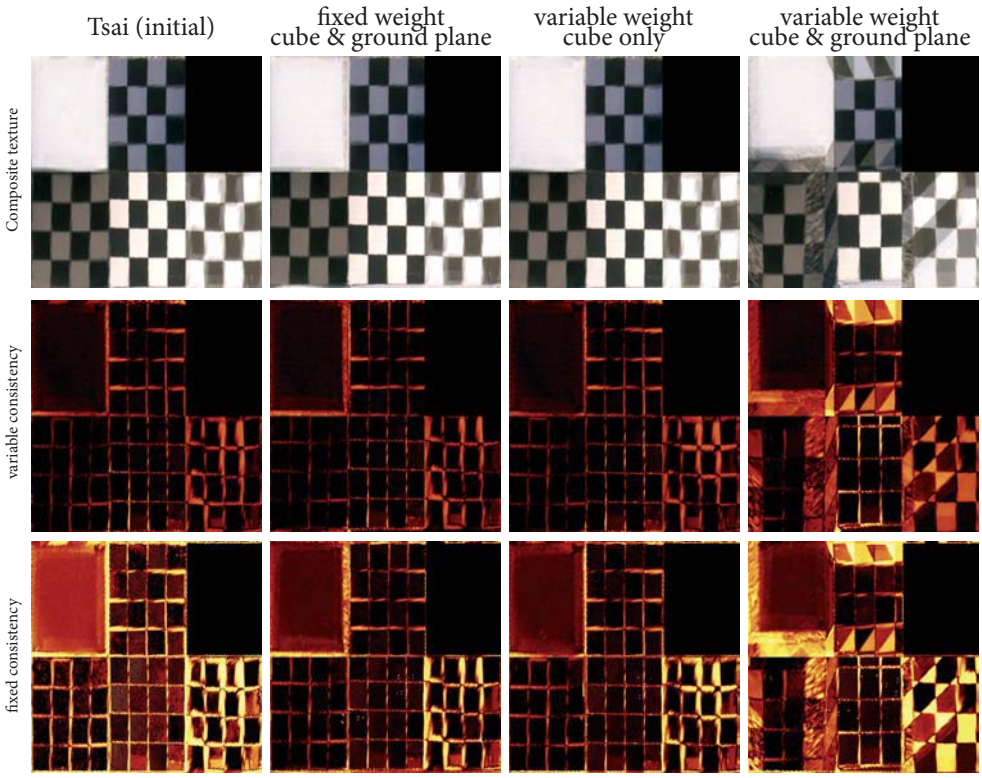
The results illustrated in Figure 7.21(a) and 7.21(b) were generated using the surface-space likelihood based on a variable-weight composite texture. The surface-space likelihood test was repeated using the fixed-weights for each view-specific texture while including the ground-plane in the scene-graph. Interestingly, the MAP estimate (illustrated in Figure 7.21(c)) was able to align the model with the reference images.

It is not clear what advantage the fixed-weight composite texture affords over the variable texture in this test. A comparison of the textures at the Tsai solution shows that the back-projection of the cube's edges onto the ground-plane is more evident on the fixed texture (see Figure 7.24). Figure 7.25 illustrates the consistency image for the three

[2]The figure also includes the ground-plane, although it was not used to compute the surface-space likelihood.



(a) Ground plane texture.



(b) Cube texture.

Figure 7.25: The composite, variable-consistency and fixed-consistency textures.

MAP estimates when compared with the initial Tsai estimate. The texture consistency images suggests that the visibility constraint as a possible reason for the failure of the variable weight and ground-plane test. The ground-plane in all other cases has a region of high inconsistency in the top right-corner, indicating that this area does not satisfy the visibility prior. Consequently, the test involving the variable weight composite texture with the ground-plane has found a set of camera parameters that meets the minimum visibility requirements at the expense of inconsistency across the surface.

The ground-plane, then, was too large for this particular test. Ensuring that the scene is sufficiently visible is a fundamental problem with the surface-space likelihood approach; the plane's visibility constraint is was not a problem when it was removed for the second, successful test. The concerns with the visibility prior does not explain why the fixed-weight test was successful with the ground-plane, because it, too, used a fixed-size ground-plane. It is likely that the fixed-weight photo-consistency mitigated the visibility prior in this instance.

7.2.3 Template test

The advantages of using template structure to reduce scene-ambiguity was tested against the reference images illustrated in Figure 7.26. This test comprises two identical structures, differing only in position and orientation along a common ground-plane. The regular structure exhibited by two seemingly identical but disjoint objects can be exploited by a carefully constructed scene-graph.

The scene sub-graph for a single house is identical to the graph used in Section 7.2.4, although the chimney height in this test was an additional scene parameter. The house sub-graph was connected to two transformation edges that translated and rotated two copies of the same hypothesised house structure. The house was therefore used a *template* that was matched against two distinct objects in the scene. Each house template was parameterised by five parameters, and the scene-graph required three parameters per house to position each house independently of the other, giving a total of eleven scene-parameters.

The cameras were calibrated with the Voodoo Camera Tracker [64, 63] and the eleven scene parameters were optimised by a simulated annealed MCMC chain over 50,000 iterations. The MAP estimate is illustrated in Figure 7.27, and synthetic views of the scene from novel view-points are illustrated in Figure 7.28. The wire-frame overlay in Figure 7.27 illustrates that the solution is correlated with the reference surface.

Figure 7.26: Three reference images of the house-template test. There are eleven images in the complete sequence.

Figure 7.27: The projection of the template-house test using the reference camera parameters. The model is rendered as a 1) wire-frame model and 2) as a solid model with the composite texture.

Figure 7.28: Synthetic images of the template-house test generated from novel view-points using the maximum *a posteri* composite texture.

7.2.4 House test

The surface-space likelihood was tested on seven frames of the real sequence illustrated in Figure 7.29. The cameras were calibrated using the Voodoo camera tracker [64, 63] and a scene-graph was constructed by stacking a prism on top a cube. Seven scene-parameters were used to describe:

- house translation about the ground-plane;
- house orientation (rotation about the $y-$ axis);
- house scale along the $x-$ and $z-$ axes;
- wall height; and
- vertical scale of the roof prism.

The scene-graph was arranged so that the 'house' parameters transform both the wall cube and roof prism, implying that the roof prism shared the same orientation as the wall block. The height of the prism and the height of the wall block were considered independently because the prism height effectively controls the roof angle. Uniform priors were used for each parameter; the shape distribution is illustrated by the last frame in Figure 7.29.

The maximum a posteri estimate of the seven house parameters was found using a simulated annealed MCMC chain over 20,000 iterations. A synthetic image of the scene-graph, parameterised by the MAP estimate and textured with the corresponding composite texture, is illustrated in Figure 7.30, and with the reference cameras in Figure 7.32. These Figures illustrate that the surface-space likelihood has been able to create a 'clean' texture that is not corrupted the back-projection of the house onto the ground-plane, even though the ground-plane is partially occluded in every view.

A second test was conducted to improve the reconstruction by allowing for small changes in camera parameters. In this test, each camera was allowed to translate and rotate about its local co-ordinate system and make small adjustments to its focal length and position of the optical centre. This added nine parameters per camera to give a total of 80 free parameters over the seven cameras. A simulated annealing chain of 50,000 iterations was used to find the solution shown in Figure 7.31.

The refinements made by the camera optimisation can be seen in Figure 7.33, which illustrates the improvements to the composite texture by re-projecting the MAP estimate with one of the reference camera parameters. Four exemplar areas of the im-

Figure 7.29: The seven reference images and prior distribution for the house reconstruction test.

Figure 7.30: Synthesised images of the house reconstruction from novel view-points.

(a) House parameters.

(b) House & camera parameters.

Figure 7.31: The juxtaposition of the two house solutions. The top frame illustrates the re-projection of the wire-model onto the reference view; the second frame is the re-projection of the model from the reference camera parameters with the composite texture; and the third frame illustrates the surface consistency metric.

Figure 7.32: A novel view of the house reconstruction.

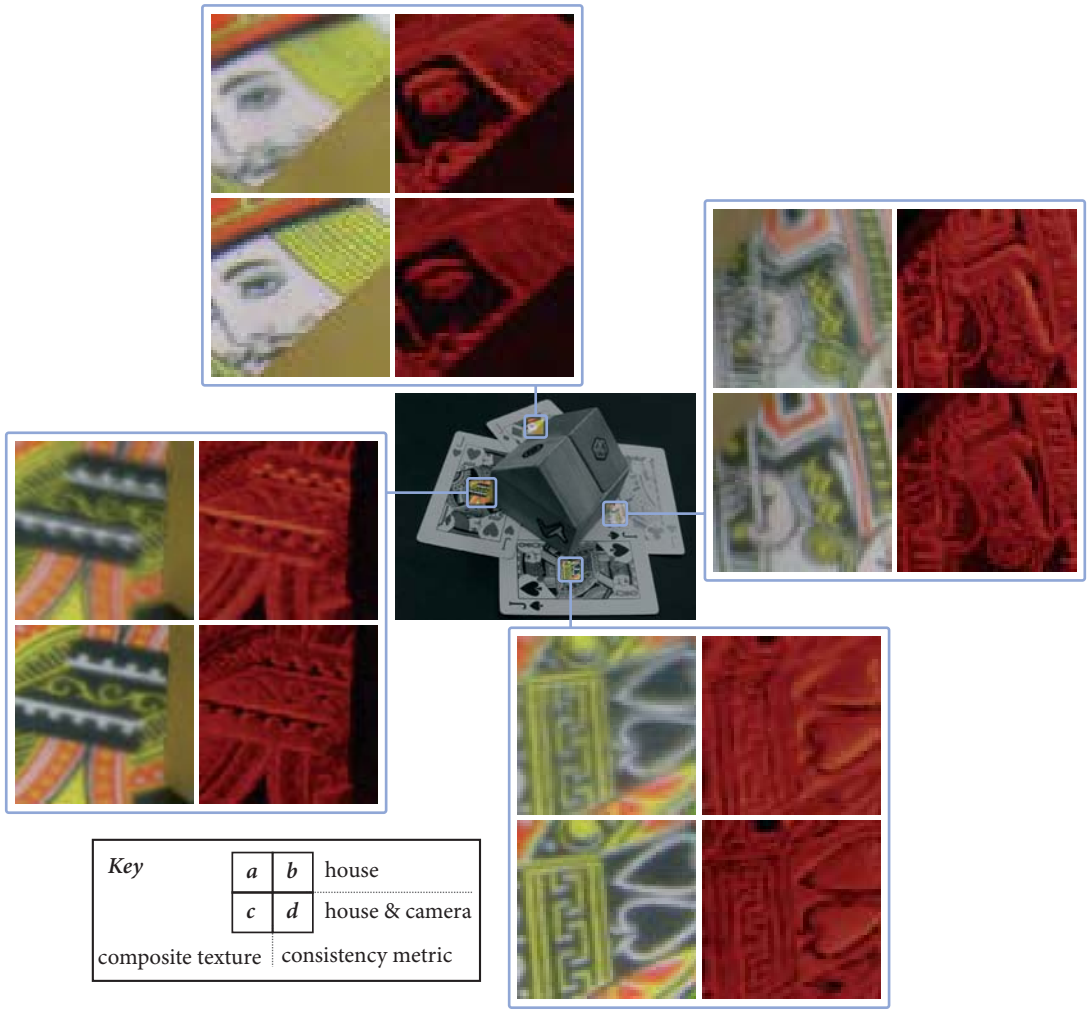


Figure 7.33: The juxtaposition of synthetic images from two optimisations of the house test where only the house was optimised along (frame *a*) and when the cameras could also be adjusted (frame *c*).

ages are highlighted. The blur evident in the *a* frames is caused by slight misalignment of the reference cameras, effectively leading to a surface-space convolution when the samples from the seven reference images are averaged together. Correcting the camera alignment reduces the size of the convolution, leading to the sharper images in the *b* frames.

7.3 Gradient-based optimisation

Like the image-based likelihood described in Chapter 4, we used Simulated Annealing to find the maximum a posteri surface model. Simulated Annealing involves randomly walking through hypothesised surface models, and could potentially take a large number of iterations to converge to a global maximum. Given that the slices through the surface-based and image-based likelihoods illustrated in Figures 7.10—7.14 indicate that the likelihood functions are 'well behaved', could a 'classical' optimisation approach, based on the derivatives of the likelihood, converge to the MAP estimate faster than a random walk?

In general, classical optimisation strategies such as Powell's method [69] rely on computing the derivatives of the objective function to estimate the direction of the local maximum. Accordingly, the surface likelihood must be continuous over the space of parameter values. The characteristics of both the image-based and surface-based likelihoods are determined by three properties:

- the likelihood's treatment of occlusion;
- the surface texture in the reference images;
- the method used to sample synthetic images of the hypothesised geometry.

These properties affect the continuity of the likelihood, and therefore are a factor in considering whether the likelihood is amenable to a gradient-based optimisation.

Occlusion is a defining characteristic of both likelihoods. The surface-based likelihood relies on an occlusion prior to prohibit trivial solutions. Because the occlusion prior defines the event that a point is occluded in all views with probability zero, the log likelihood is undefined for significant sub-spaces of parameter values. In contrast to the surface-likelihood, the image-based likelihood is defined for all hypothesised parameter values. The image-based likelihood is based on the pair-wise comparisons of hypothesised point correspondences, which yields a finite, scalar photo-consistency metric for all hypothesised correspondences. Occluded correspondences are indistinguishable from perfectly photo-consistent observations, however, and therefore this property holds for both visible and occluded correspondences.

For a given hypothesised surface that is sufficiently visible, both the image-based and surface-based likelihoods evaluate the surface using the variance of hypothesised point correspondences. The continuity of the reference images can therefore affect the continuity of the likelihood. Consider, for example, the scene configuration illustrated

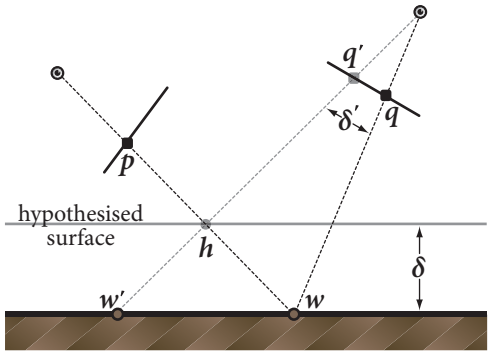


Figure 7.34: Change in surface space involves a corresponding change in image space.

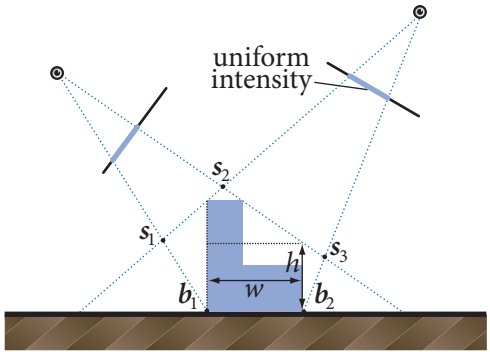


Figure 7.35: Image plateaus lead to ambiguous likelihoods.

in Figure 7.34, where the scene point $\boldsymbol{w}$ projects to points $\boldsymbol{p}$ and $\boldsymbol{q}$ in the first and second reference cameras, respectively. If the hypothesised surface is correct, then computing the variance of $\boldsymbol{w}$ will correctly involve comparing the similarity between points $\boldsymbol{p}$ and $\boldsymbol{q}$ in RGB space. In this example, translating the hypothesised surface by δ will mean that the hypothesised point correspondences will also translate by δ' in image-space.

The likelihood of $\boldsymbol{p}$ with respect to the range of hypothesised surfaces described by translating the hypothesised surface by δ is therefore directly related to the continuity of the second image's intensity function. If the reference image intensity function is discontinuous (ie. small changes across the image correspond to large changes of intensity), then the likelihood will also be discontinuous. In practice, however, the likelihood tends to be smooth because the *overall* effect of combining the variance across the entire surface mitigates local discontinuity along edges. This result is illustrated by Figures 7.10—7.14, which appear continuous despite the sharp edge detail over the cube texture.

The effect of combining the joint probability over multiple scene-points is also useful in regards to plateaus in the reference image intensity. Consider the scene configuration illustrated in Figure 7.35 where the ground-plane is fixed with respect to the reference cameras. Suppose that the image intensity under the projection of the two boxes is uniform, but different from the ground-plane texture. In this example, the uniform appearance will create plateaus in both the image-based and surface-based likelihood. The uniform intensity constrains the boxes' geometry to lie within the polygon defined by the points $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{b}_2, \boldsymbol{b}_1\}$, with the constraint that at least one scene point must lie on the each edge of the silhouette. While the edge constraint will

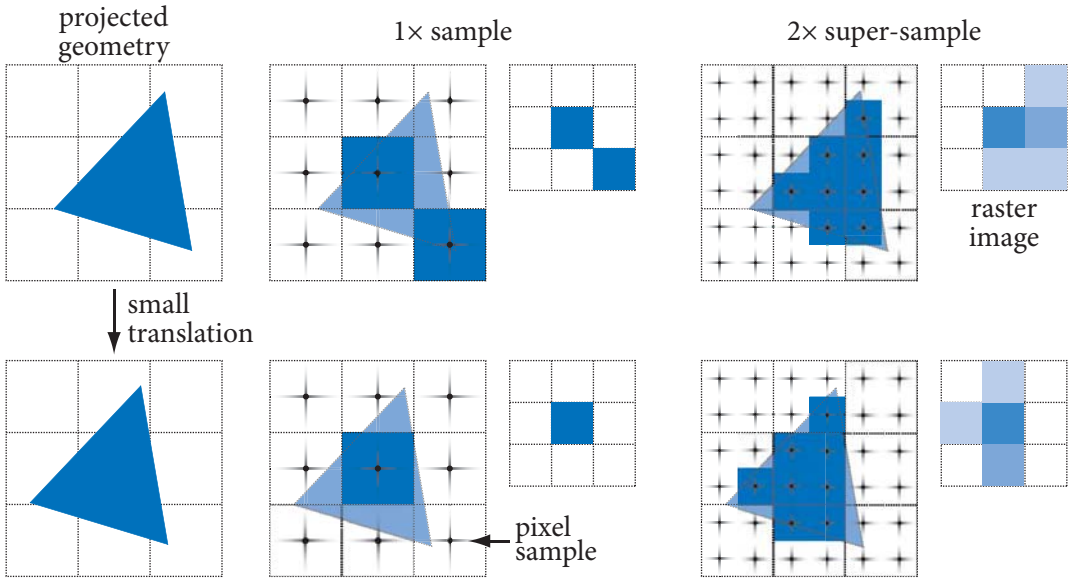


Figure 7.36: Approximating images by discrete pixel sampling.

correctly align the base of the boxes at $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, there is insufficient texture information to constrain the width and height of the two boxes. The joint probability over all scene-points is therefore able to establish correct values for subsets of the scene parameters, given suitable geometry constraints imposed by the scene-graph, but is unable to infer structure from the plateaus induced *in some cases* by uniform intensity in the reference images.

Even given a sufficiently textured, continuous image, the likelihood's continuity is further affected by the method used to sample synthetic images of the hypothesised scene. Both likelihoods generate synthetic images of the hypothesised surface by sampling the reference images. As noted by Smelyansky [56], render engines typically sample only a single point per pixel, yielding images that are not differentiable. Smelyansky uses a renderer that accounted for the partial flux of all triangles that projected to a given pixel, allowing the images to be differentiated with respect to scene-geometry and therefore accommodate a gradient-based optimisation. Our approach, however, relies on OpenGL, and is therefore subject to its discrete sampling process.

Super-sampling the images is one approach supported by OpenGL that is able to mimic Smelyansky's differentiable renderer. Super-sampling is the process of generating more than one scene sample per pixel; the result is the average of all samples contributing to that pixel. The problem of discrete sampling of two triangles with identical

image area is illustrated in Figure 7.36. A renderer that is capable of accounting for all flux attributing to a given pixel would, in this case, yield two images with constant accumulated intensity, ie. $\sum_{\boldsymbol{p}} \mathrm{i}_1(\boldsymbol{p}) = \sum_{\boldsymbol{p}} \mathrm{i}_2(\boldsymbol{p})$. In this example, the single sample per-pixel yields two images with different intensity— namely $\frac{2}{9} \neq \frac{1}{9}$—whereas the intensity remains constant under 2× super-sampling. Super-sampling, however, is able to only push back the problem of discrete sampling; it is not a complete solution. While the sampling effect is a theoretical limitation affecting the likelihood's continuity, it is not clear how this would translate into a practical problem regarding differentiable images.

These sources of discontinuity would suggest that the image-based and surface-based likelihoods are not suited to a gradient-based approach. Occlusion is the key problem, because occluding the scene in all reference cameras will create a plateau in both likelihoods, thereby stalling a gradient-ascent based approach. A gradient-based approach, however, could be used in conjunction with a random sampling approach like MCMC. In such a scheme, a MCMC process might generate random starting locations for a gradient-ascent approach. This would overcome the problem of starting in a plateau, but allow the advantages of a gradient-ascent approach to quickly find the local maximum, with the hope that generating sufficient number of starting points would eventually find the global MAP surface.

7.4 Visibility bias

Despite using a visibility constraint to reject trivial solutions, the surface-space likelihood is also subject to a visibility bias. The bias is evident in experiments where camera parameters were permitted significant freedom to change the surface's visibility. The MAP estimate in these cases would find a solution that minimised the surface visibility while satisfying the occlusion prior. Reducing the number of cameras that can contribute to the surface's photo-consistency metric unfortunately also increases shape ambiguity.

The visibility bias is caused by decreasing variance as observations are removed. Each observation $\boldsymbol{x}_i$ of a surface point $\boldsymbol{p}$ is a measurement of the surface reflectance $\boldsymbol{y}$ at $\boldsymbol{p}$. We assume that each observation is affected by Gaussian sensor noise $\epsilon \leftarrow \mathcal{N}(\mu = 0, \sigma)$ to yield a set of j observations $\mathcal{M} = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i = \boldsymbol{y} + \epsilon_i\}$. The χ^2 statistic,

$$\chi^2 = \sum_{i=1}^{j} \frac{(x_i - \boldsymbol{y})^2}{\sigma^2}, \tag{7.11}$$

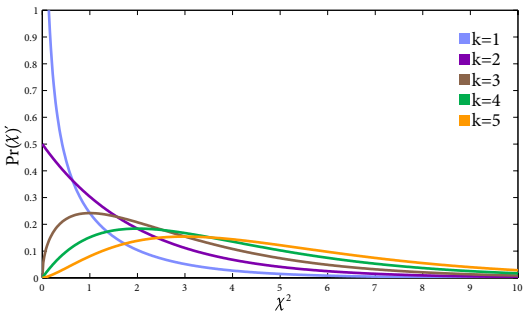


Figure 7.37: The χ^2 PDF

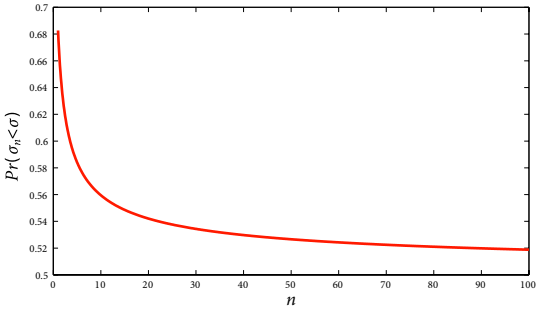


Figure 7.38: Sample size versus probability.

can be used to test if the distribution of the measurement set $\mathcal{M}$ deviates from the expected distribution [10]. The χ^2 statistic is distributed according to the PDF illustrated in Figure 7.37, which is given by

$$f(x,k) = \frac{(1/2)^{k/2} x^{k/2-1} e^{-x/2}}{\Gamma(k/2)} \tag{7.12}$$

where $k = j - 1$ is the number of degrees of freedom and

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt. \tag{7.13}$$

The χ-statistic (7.11) is expected to equal j, the number of samples in $\mathcal{M}$; it will be less than j if the standard deviation of $\mathcal{M}$ is smaller than the expected distribution. Given the PDF from (7.12), the probability of this occurring from a random sample of size j is

$$\Pr(\chi^2 < j) = \int_{-\infty}^{j} f(x, j-1)\, dx. \tag{7.14}$$

This probability is graphed with respect to varying sample size in Figure 7.38, illustrating that as the sample size decreases, the probability of the set having a smaller variance will increase. The consequence of this relationship between sample size and variance is that removing observations from the sample set will often improve the surface's perceived photo-consistency. This leads to a bias toward minimising the set of observations over the surface.

Conclusion

Chapter VIII

We approach the problem of reconstructing geometry from images by using Bayesian inference over the parameters of a user-defined scene-graph. Given the scene-graph and the reference images, the reconstruction problem becomes one of finding the transformation parameters that best recreates the reference images given the constraints imposed by the scene-graph's hierarchy. To this end, we seek the maximum a posteri solution from a Bayesian learning process characterised by one of two likelihoods. The first likelihood compares the hypothesised surface's projection against the reference images, and the second evaluates the consistency of the surface texture that is estimated by back-projecting the reference images onto the hypothesised surface. The advantage of the surface-space likelihood is that the hypothesised surface's visibility can be monitored, and is therefore less likely than the image-space likelihood to find trivial photo-consistent solutions.

The posterior distribution is defined over the parameters of a scene-graph. The scene-graph describes a set of user-identified shapes that are related by parameterised transformations. The graph's structure defines geometric constraints by stacking and aligning primitives in the hierarchy. The advantage of using a scene-graph is that it constrains the set of recoverable shapes and simplifies the reconstruction problem by greatly reducing the number of free-parameters. Unlike Debevec's Façade, which also requires the user to build a scene-graph [15], we do not require the user to identify correspondences between the 3D model and its projection in the reference images.

Given the user-defined scene-graph, the reconstruction problem becomes one of finding the parameter values that transform the graph into structure that best models the surface in the reference images. Our approach to this problem seeks the maximum of a Bayesian posterior distribution given by the joint probability of the scene-graph priors and the likelihood of the surface generated by a particular parameter vector. The surface-space likelihood is given by the the hypothesised surface's ability to recreate the reference images when imaged using the reference camera's parameters. While

this is a photo-consistency measurement used by increasing number of reconstruction methods, our approach reduces the hypothesised surface's consistency to a single likelihood metric for the *entire* surface.

We investigated two likelihoods to measure the hypothesised surface's photo-consistency. The first likelihood, described in Chapter 4, measures consistency between every pair of reference images. The hypothesised surface's consistency is measured between a given pair of cameras by back-projecting one image onto the surface and imaging the result in a second camera. The hypothesised surface's image-space likelihood is given by accumulating the pixel-wise difference in colour-space between the synthetic image and the destination camera's reference image.

The image-space likelihood is biased against occlusion because the likelihood relies on the projection of hypothesised surface points in two images. Because occluded surface points cannot be compared and therefore do not contribute to the photo-consistency metric, the image-space likelihood is maximised when the surface is completely occluded. The image-space likelihood is therefore biased toward trivial solutions unless suitable bounds are placed on the scene graph's parameters. Given suitable priors, however, tests demonstrate that the image-space likelihood can perform robustly even in the presence of heavy image noise (see Figure 4.14 on page 67, for example).

We addressed the occlusion bias by rejecting hypotheses that do not satisfy a visibility constraint. Monitoring the surface's visibility required a surface-space likelihood to measure the photo-consistency of points on the surface, rather than the projection of the surface points into the reference images. This shift in measurement space allows for an occlusion prior to classify noisy, visible points as more likely than points not visible in at least two views. This constraint ensures that all points on the surface are measured for consistency, thereby prohibiting trivial solutions.

The surface-space likelihood measures photo-consistency by 'folding' each reference image into surface space to simplify the problem of measuring photo-consistency. A given reference image is back-projected onto the hypothesised surface and then mapped into the surface space via the surface's texture map. Whereas Isidoro must dynamically compute the map between geometry and texture image from the estimated hull to unfold reference images into surface space [26], our approach is able to use static texture maps defined specifically for each scene primitive. This process, described

in Chapter 5, creates a *view-specific texture* to represent the surface's hypothesised reflectance in a space independent of the surface's projection.

As described in Chapter 6, the set of view-specific textures are assembled into a composite texture to yield the surface's estimated reflectance. Each texel in the composite texture is a weighted average of the corresponding unoccluded texels in the set of view-specific textures. Similar to consistency metrics proposed by Hornung and [25] and Li [33], we use the angle between the surface's normal and the reference camera's optical axis as a confidence measure when considering a hypothesised point's projection. This view-angle is used to weight each texel to minimise artifacts caused by projecting an oblique surface into a single pixel.

The composite texture is given the average observation of the hypothesised surface from each reference image. Under the assumption that the surface is purely diffuse, a low variance across the view-specific images suggests that the hypothesised surface is an accurate representation of the surface visible in the reference images. Accordingly, the surface-space likelihood described in Chapter 7 is given by the similarity between each view-specific texture and the composite texture. This approach effectively measures the variance of the view-specific textures, and therefore represents the photo-consistency constraint employed by a large number of prior approaches to appearance-based reconstruction [51, 52, 29, 13, 54, 55, 46, 37, 48, 47, 25, 19, 61, 6, 68, 8, 1, 9, 26, 27, 72, 70, 65, 56].

The surface-space likelihood evaluates the variance of the view-specific textures as a measure of the surface's photo-consistency. Unfortunately, a surface point's variance is likely to decrease as the set of hypothesised observations decreases, thereby making occluded surfaces appear more likely. An occlusion prior is used with the surface-space likelihood to prohibit trivially photo-consistent solutions. Agrawal's probabilistic carving [1] also require a lower bound on the surface's visibility, but uses this statistic to choose a point's likelihood from a set of consistency measures. Instead, our approach defines a point's likelihood as a function of the point's photo-consistency if it is sufficiently visible, but zero if it is occluded. Although the occlusion likelihood enforces a lower-bound on the surface visibility, the likelihood may converge to a solution that meets the *minimum* number of views from an occlusion prior. Reducing the number of views increases scene ambiguity, leading to solutions that do not accord with the reference images.

The difficulty our approach has with occlusion is not shared by approaches that

maintain a monotonically increasing visibility model. Space carving [29], for example, is not concerned with estimating the likelihood that a voxel is occluded. Instead, it uses occlusion to determine which views are allowed to vote for a voxel's consistency. Because the hypothesis begins with a super-set of the true scene, the surface visibility can *only increase* as inconsistent voxels are removed. Space carving therefore maintains a conservative estimate of occlusion and modifies the surface only when there is sufficient evidence that a voxel is inconsistent.

Our approach, in contrast, must allow the surface's occlusion to radically change as the scene-parameters are adjusted. Because consistency can only be determined by visible surface points, this leads to trivially photo-consistent solutions and has required the use of visibility priors to ensure a 'sufficiently visible' surface. This does not address all problems with occlusion, however: the likelihood will often move cameras away from the scene to minimise the amount of accumulated sensor noise over the surface. The tendency to take short-cuts to minimise small but accumulated inconsistency is a problem also experienced by a number of prior approaches. Whereas Vogiatzis [68] and Isidoro [27] use heuristics to artificially inflate the reconstruction, our approach must rely on priors to ensure that part of the surface is visible.

Neither the image-space nor the surface-space likelihoods assume that all points are visible in all views. While this allows for very wide-base lines between cameras, this assumptions means the likelihood function is highly non-linear. We randomly sample the posterior using a Monte Carlo Markov Chain to avoid being trapped in a local maxima. This process involves evaluating the likelihood of a succession of shape hypothesises. Evaluating the likelihood of a shape hypothesis requires measuring the photo-consistency of a very large number of surface points—any number of which could be occluded. Because a large number of hypothesises must be tested, this requires testing the photo-consistency of millions of points to find the MAP estimate.

We leverage graphics hardware to compute the hypothesised surface's photo-consistency metric. The graphics hardware's affinity for projection and texture mapping is used to remap the reference images into target spaces, such as a reference images in the image-space likelihood and the texture space in the surface-space likelihood. Graphics hardware is designed for synthesising images from geometry and textures, and therefore the hardware had to be adapted to estimate texture given the reference images and a hypothesised surface. Some operations—such as occlusion with respect to reference images, for example—could not be carried out by the hardware's dedicated mechanism

because these operations only worked with respect to the frame-buffer. Because the frame-buffer was used to represent the surface texture, shaders were used to manipulate and encode the geometry in a manner 'compatible' with OpenGL's rendering pipe-line.

Our reconstruction paradigm confers a number of advantages over the state-of-the-art. Although our approach requires a user-supplied scene-graph, we do not require the scene-graph to be precisely correlated with all reference images. Although Façade [15], for example, is able to simultaneously recover geometry *and* cameras, the user is required to mark the model's silhouette in all reference images. Other approaches require an *a priori* upper-bound on the shape [29] and sometimes a segmented image [30, 62]. Our approach does not require either.

Our system recovers a polyhedral mesh rather than a sparse set of feature-points that cannot be directly used for new-view synthesis, nor a dense voxel model which is unable to express geometric constraints. Instead, our reconstructions are described by textured geometry parameterised by relatively few unknowns. This model can be directly used to generate convincing synthetic views of the scene. Although the accuracy of the reconstructions critically depend on the scene-graph's description, we have demonstrated that the Bayesian framework is able to exploit geometric constraints to recover models even when the scene-graph was an 'idealised' form of the scene in the reference images. The reconstruction illustrated on page 70 is one example of this case.

Even given relatively uninformative priors, the appearance-based likelihood is able to create models that are strongly correlated with the images. This is evident in the house test on page 142, where the edges of the house model are aligned with the edges in the reference images by illustrating the projection of the wire-frame model in the reference images. Edge detection is not used by either the image-space or surface-space likelihoods. Back-projecting a single reference image onto the reconstructed surface often 'hides' small errors in the reconstruction. Our approach, however, simultaneously back-projects *all* reference images onto the surface to illustrate the accuracy of the reconstruction.

8.1 Future Work

The advantage of using a scene-graph as the underlying surface model is that useful geometric constraints such as co-planarity and orthogonality. The disadvantage, however, is that the scene-graph limits the user to a set of pre-defined primitives and, more importantly, the user is required to describe a scene-graph to initialise the process. While we have described the scene-graph's *structure* in Chapter 3, we have not explored methods for its construction. In practice, the scene-graph was defined in a text-based environment. This approach is not intuitive for an end-user.

8.1.1 Extended primitives

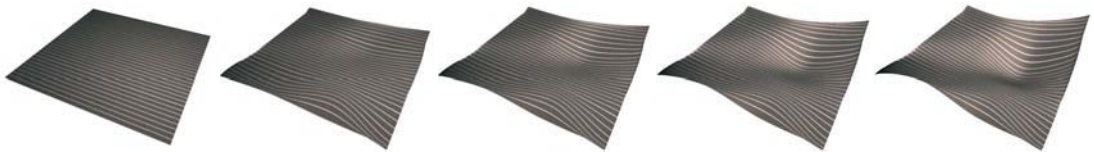
Chapter 3 describes primitives that can only be parameterised by transformations along the scene-graphs edges. Although these primitives can be embedded to create apparently complicated shapes, the rigidity imposed by the primitives is useful only for certain types of architectural scenes. The paradigm behind the surface-based photo-consistency approach is not limited to fixed geometry, however, and could be extended to new types of parameterised geometry.

Parameterised surfaces

Adding parameterised curves, such as Non-uniform Rational B-Splines (NURBS) [42], could usefully recover detail which is too fine to adequately model with geometric blocks. Adding parameterised surfaces is consistent with the modelling paradigm presented in Chapter 3: a static texture map can still be defined for the surface patch, and the control points become the optimisation parameters. A parameterised surface would be particular suitable in the cardboard box sequence to correct the concave surface in Figure 4.25, for example. Using photo-consistency over a NURBS surface is illustrated in Figure 8.1, where Figure 8.1(a) illustrates an example deformed surface and Figure 8.1(b) illustrates the back-projection of the reference images onto the surface.

Generalised cylinders

The generalised cylinder is another type of primitive that could be used in our modelling paradigm. Generalised cylinders are defined by a 2D cross-section swept along a 3D curve [43]. Although the 'classical' cylinder is modelled by a circle cross section that is



(a) Psuedo-texture to illustrate curvature.

(b) Hypothesised composite texture.

Figure 8.1: An illustration of changing a parameterised surface's curvature and the effect of averaging its back-projected texture.

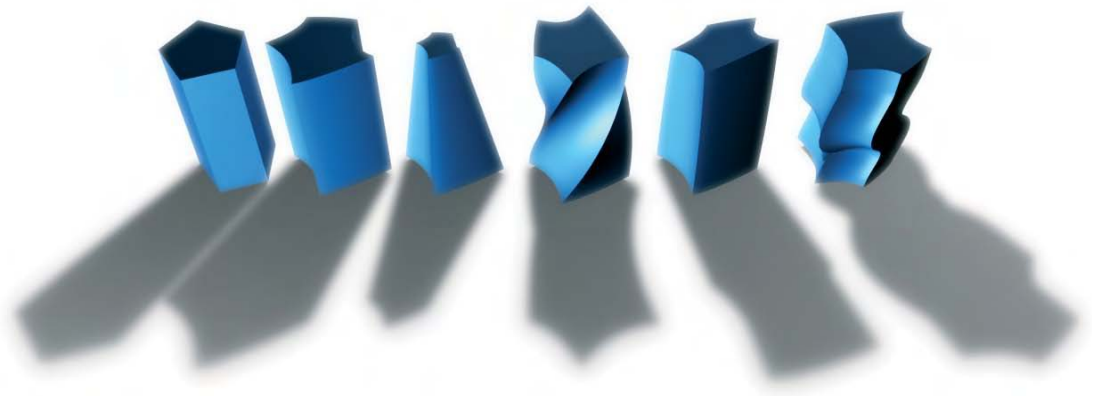


Figure 8.2: Examples of generalised cylinders illustrating changing the cross-section shape (*1* and *2*), its scale (*3*), rotation (*4*) and alignment (*5*) across the axis and the cylinder axis itself (*6*)

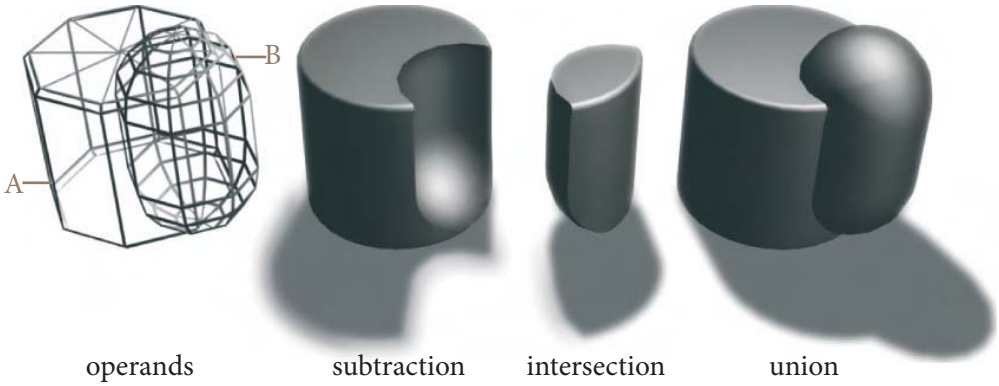


Figure 8.3: An example of constructive solid geometry illustrating the operands and the results from subtracting B from A, the intersection of A and B and the union of A and B.

orthogonal to a line-segment axis, a generalised cylinder has an arbitrary cross section, illustrated by cylinders 1 and 2 from Figure 8.2 and a non-linear axis (the sixth cylinder). Furthermore, the cross section can be transformed as a function of its position along the axis, including scale (the third cylinder), rotation about the axis (cylinder 4) and off-axis rotation (cylinder 6). Generalised cylinders could be incorporated into our modelling paradigm by using parameterised cross-section and axis. Parameterised functions could also be used to control the cross-section's position along the curve.

Extending constructive solid geometry

Constructive solid geometry is a method for creating new shapes by applying Boolean operations to polyhedral meshes [39, 14]. Because closed geometry defines a subset of interior space, the set of interior space between different primitives can be combined to create new sets of interior space, and therefore new polyhedral meshes. Although our scene-graph effectively allows the union of two primitives, CSG could be used to create more complicated shapes defined by the subtraction of one shape from another or the intersection of two shapes. An example of CSG used to construct three different surfaces from two operands is illustrated in Figure 8.3. Because CSG generates a new polyhedral mesh that is fundamentally different from its source operands, the static texture unfolding would have to be replaced by a method for dynamically computing a new set of texture co-ordinates for an arbitrary polyhedral mesh.

8.1.2 Reconstruction without prior calibration

Although we have conducted experiments where the camera parameters were part of the reconstruction, all of our experiments have used calibrated cameras. What, then, are the prospects for using the photo-consistency likelihood when reconstructing without calibration? Without prior information to constrain the search-space, a MCMC chain would take considerably longer to explore the set of surface parameters, particularly as we rely on the priors to bootstrap the process. There is at very least, however, a subspace of solutions up to an unknown scale; but it is quite likely that there is a further photo-consistent solutions with varying camera parameters beyond the unknown scale factor. Further investigation might explore this problem and whether a data-driven MCMC chain could be used by looking for image features in the reference images to define a relaxed set of priors on camera parameters.

8.1.3 *Scene construction*

A similar approach to Façade [15], which allows the user to interactively construct a scene-graph, could be used in our system. The Façade interface presents a view of the model that allows the user to add and manipulate the building blocks. Blocks can be selected via the display, which can then be used as parents for new primitives. Control forms are used to define (in)equalities between parameters by allowing the user to define scene-parameters which can then be used in expressions concerning scene transformations.

We have explored a method of interactively stacking and aligning primitives as a basis for future work. Like Façade, our proposed approach maintains a view of the reconstruction and allows the user to iterative describe the scene hierarchy. We propose several extensions to describe relationships between primitives and priors on scene parameters. In general, our proposed approach:

- creates, orientates and scales primitives by clicking and dragging on a 'construction plane',
- defines constraints between primitives by 'snapping' vertices together, and
- defines priors on the scene parameters using example transformations to indicate uncertainty.

A primitive requires an initialisation of nine parameters to define its translation, rotation and scale in scene-space. Most user/computer interfaces, however, are limited to clicking and dragging points on a 2D view-port. This presents the problem of defining a nine-parameter model as efficiently and intuitively as possible using a two-parameter interface. We approach this problem by using a 'construction plane' to add constraints in scene-space. This construction plane ensures that each pixel in the view-port corresponds to a unique point in scene-space. The user is able to select any scene-plane to use as a new construction plane, allowing the user to define a scene-hierarchy by iteratively defining primitives as a basis for further construction planes.

Beginning with a default ground plane, the user creates primitives by clicking and dragging on the plane to define the origin and scale in two dimensions of new primitives. This process is illustrated in frame 1 of Figure 8.4. In this example, the base of the house is defined by the line segment $\overline{\boldsymbol{pq}}$, where $\boldsymbol{p}$ is used as the rectangle's origin and the length $\|\boldsymbol{q} - \boldsymbol{p}\|$ describes the rectangle's scale. The primitive's height is defined by clicking a point along the ground plane's normal. This is illustrated in frame 2, where

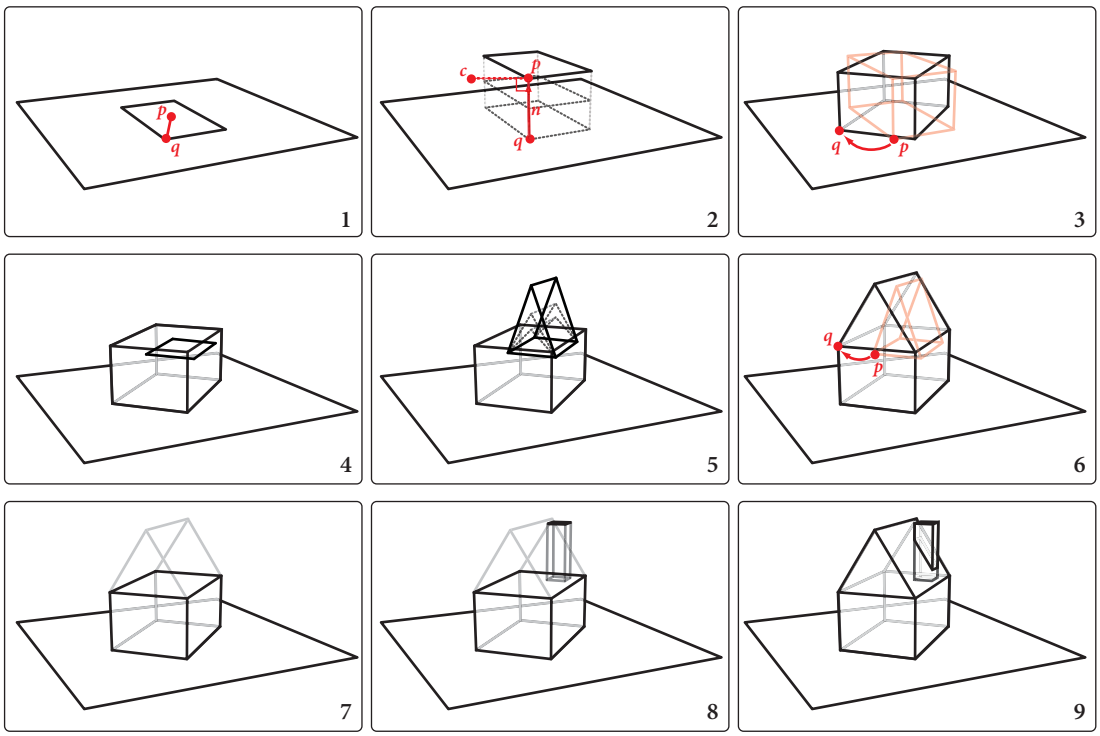
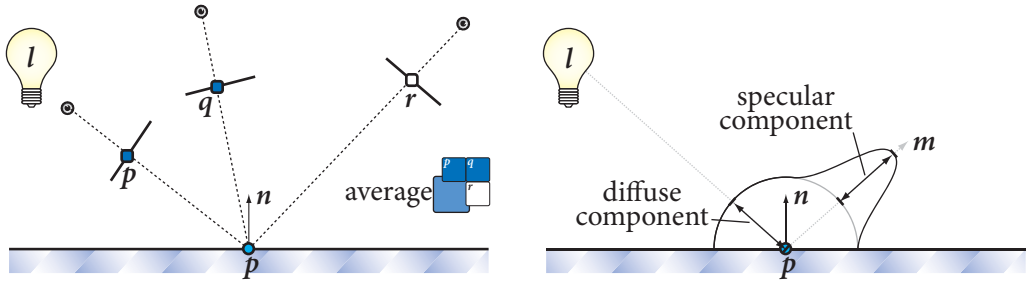


Figure 8.4: Scene construction example.

the cube's height is defined by the length $\|\boldsymbol{p}-\boldsymbol{q}\|$, where $\boldsymbol{p}$ is the projection of the mouse cursor $\boldsymbol{c}$ onto the ground plane's normal $\boldsymbol{n}$.

A primitive's rotation about $\boldsymbol{n}$ could be changed by clicking and dragging one vertex coincident with the ground plane. The cube in frame 3, for example, is rotated about the ground plane's normal by dragging the vertex $\boldsymbol{p}$ to a new point $\boldsymbol{q}$ on the ground plane. Rotations about other axes could be specified by using control modifiers to select new rotation vectors, or by selecting the sides of the cube as new construction planes to define rotation in spaces orthogonal to the ground plane.

Primitives can be related by defining construction planes using existing scene-primitives. This process is illustrated in frames 4 and 5 of Figure 8.4, where the user has selected the top of the cube to use as a new 'ground plane' for constructing the roof prism. This process describes a scene-hierarchy by defining the transformation of one primitive (the roof prism, in this case) relative to the cube's transformation. Equality constraints between primitives could be defined by 'snapping' together vertices of disjoint primitives. The scale, orientation and position of the house roof could be



(a) Specular surfaces appear inconsistent with a diffuse light model.

(b) The light position, diffuse and specular components must be estimated.

Figure 8.5: Specular surfaces cannot be estimated by a likelihood function that assumes diffuse reflectance.

aligned with the walls by dragging the roof prism vertices coincident with its ground-plane onto the top of the cube (as illustrated by aligning point $\boldsymbol{p}$ with $\boldsymbol{q}$ in frame 6).

Because our model places much emphasis on 'attaching' primitives together, primitives can be hidden to access the underlying geometry. This is illustrated in frames 7 and 8, where the roof prism is 'hidden' in frame 7 so that the chimney can be constructed on top of the cube in frame 8. Although the chimney must be aligned with the walls, there are not any vertices that can be used to define constraints in a process similar to the roof construction. A control-form approach similar to Façade could be used to allow the user to explicitly constrain the chimney's rotation in this case.

A scene-graph is described in this model by stacking primitives together to build a tentative model. Although the model requires an explicit description of relationships (for example, the roof must be stacked on top of the cube walls), the scene-graph includes an initial estimate of the underlying geometric transformations. The user would not be required to ensure that the transformations are correlated with respect to the reference images, however, because the estimation of these parameters is left to the photo-consistency system described in Chapters 4—7.

8.1.4 Estimating specular surfaces

The photo-consistency likelihood assumes perfectly diffuse surfaces to simplify the problem of estimating reflectance from multiple, wide-base line views. A large number of useful scenes can be reconstructed with this assumption: surfaces made of stone, concrete, brick, paper, cloth and unvarnished wood, for example, are amenable to this

approach. Specular surfaces—that is, surfaces that reflect light as a function of incident light angle to the surface normal—cannot be modelled by the image-based or surface-based likelihoods described in Chapters 4—7. The problem with specular surfaces is illustrated in Figure 8.5(a), where three cameras observe a 'shiny' scene-point, $\boldsymbol{p}$. In this example, the pixel $\boldsymbol{r}$ is closely aligned with the *mirror angle* of the point, and this observation adversely affects the average observation, which leads to high photo-consistency error when the composite colour is compared with all observations.

Recovering specular surfaces is a potential extension to both the image-based and surface-based likelihoods. Given that both likelihoods compare synthetic images of the hypothesised geometry with the reference images, modelling specular surfaces would require extending the number of rendering parameters to include light-source position and specular component for each texel in the scene-graph. The reflectance of the point $\boldsymbol{p}$ in Figure 8.5(b) would then be given by the hypothesised light position $\boldsymbol{l}$, light intensity, and specular co-efficient. These parameters are sufficient for a Phong reflection model [41] when used with the surface normal $\boldsymbol{n}$ and eye-vectors from $\boldsymbol{p}$ to the optical centre of each reference camera. The photo-consistency likelihood would then be given by the disparity between the observations against the hypothesised reflectance model. Further investigation might consider how this approach could be implemented in graphics hardware, and whether it can recover specular surfaces.