Computing Surface-based Photo-Consistency on Graphics Hardware

J. W. Bastian and A. J. van den Hengel
The University of Adelaide, Adelaide, South Australia
Centre for Sensor Signal and Information Processing, Mawson Lakes, South Australia
{john|anton}@cs.adelaide.edu.au

Abstract

This paper describes a novel approach to the problem of recovering information from an image set by comparing the radiance of hypothesised point correspondences. Our algorithm is applicable to a number of problems in computer vision, but is explained particularly in terms of recovering geometry from an image set. It uses the idea of photo-consistency to measure the confidence that a hypothesised scene description generated the reference images. Photo-consistency has been used in volumetric scene reconstruction where a hypothesised surface is evolved by considering one voxel at a time. Our approach is different: it represents the scene as a parameterised surface so decisions can be made about its photo-consistency simultaneously over the entire surface rather than a series of independent decisions. Our approach is further characterised by its ability to execute on graphics hardware. Experiments demonstrate that our cost function minimises at the solution and is not adversely affected by occlusion.

1 Introduction

Working in scene space on computer vision problems is a relatively new approach that confers many advantages over feature-based algorithms. A number of these techniques rely on pixelwise comparison across an image set to recover information. These comparisons are usually carried out to find evidence to support or reject an hypothesised set of parameters. Repeated application of this process with different configurations allows conclusions to be drawn about the parameter space. The key to these approaches is pixelwise comparisons: a mathematically straightforward but computationally expensive process that requires computing occlusions and comparing projections of points.

One application of the graphics-based technique is towards the recovery of static geometry from a set of reference images. Recovering geometry from images has, in general, focused on computing correspondences between features in the reference images. These features are used to calibrate the cameras and compute a reconstruction. The quality of the camera calibration crucially depends on the accuracy of these features. In contrast, the graphics-based approach compares synthetic images of a hypothetical scene with the reference images. There are a number of advantages to this approach, including the ability to recover scene shape without explicitly determining dense correspondences and its capacity to consider global structure when making local changes.

Space carving-class algorithms, for example, use a photo-consistency cost-function to validate a hypothesised surface by considering the projection of each scene point into the reference images [3] [11] [5] [4]. They begin with a conservative, initial hypothesis that is provided by the user and guaranteed to encompass the true geometry. A carving iteration validates every point on this surface by comparing its projection with the reference views. A point is considered valid if its projection fits a particular light model in all images where it is visible; such points are said to be *photo-consistent*. Inconsistent points are thought to come from disparate points on the reference surface. Removing inconsistent points creates a new hypothesised surface, which in turn changes its visibility and therefore requires further image-based comparisons until it converges to a state where no further points are removed.

Smelyansky *et al* use a similar basis to recover a height-map and make small changes to camera parameters from an initial feature-based estimate [12]. Their approach considers entire images rather than determining photo-consistency for each point, but they do not model occlusion and can therefore compute the derivative image with respect to scene and camera parameters. New views are generated by applying the image derivative to the last synthetic image.

Both space carving and Smelyansky's approaches treat reconstruction as a problem of colouring a hypothesised scene and comparing the results with the reference images. Computing the photo-consistency and generating new views involves finding the unoccluded projection in each image of a sub-set of the surface. Determining occlusions can be expensive, particularly if the scene structure is time-consuming to traverse or it is not possible to optimise the number of visibility tests. Searching detailed volumetric structures can be time-consuming because a large number of volume elements must be scanned to confirm that no opaque regions are closer to the camera than a given point.

Generating synthetic images from geometry described by triangles has received considerable attention from hardware vendors to the point where powerful commercially available graphics hardware—capable of processing millions of textured triangles per second—is relatively inexpensive [13]. Modern Graphics Processing Units implement a pipeline organised in a series of transformations, clipping and rasterisation steps. Surfaces represented by triangles are sent to the pipeline as a stream of vertices where they are transformed and clipped by a vertex processor. The transformed triangles are rastered into a collection of *fragments*. This process linearly interpolates vertex attributes—such as texture coordinates computed by the vertex processor–and binds the new values to each fragment. The fragment processor uses these attributes to apply colour to the frame-buffer; this may involve referring to texture elements (*texels*) from the currently bound textures. The fragment processor is capable of performing operations on the fragment attributes with the values sourced from the frame-buffer or other texture units.

This paper presents a method that uses graphics hardware's ability to create synthetic images from geometry to reason about the *inverse* problem of finding geometry when given a set of reference images. Rather than using a volumetric model like space-carving based algorithms, our approach is more like Debevec's Façade by using a scene-graph to constraint the reconstruction process [6]. The scene-graph encapsulates known geometric primitives with unknown parameters. It constrains the recoverable set of scenes, which is useful because an infinite set of scenes can be photo-consistent with a finite set of images [8].

The scene-graph approach effectively correlates what the system expects to see with what is actually visible in the reference images. This has two advantages over voxel-based approaches. The key advantage is that our approach is tractable in large architecture-based scenes that would otherwise require a prohibitively large number of voxels. The second advantage is that it generates spatially coherent reconstructions because photo-consistency is measured over a large surface rather than making hundreds of independent photo-consistency decisions.

The scene graph's primitives and their relationships are provided by the user. Both Façade and our approach aims to find the transformations within the scene-graph that best fit the primitives to the reference views. Façade finds these transformations by minimising the reprojection error of the scene-graph edges against its corresponding edges marked by the user in the reference images. Rather than only considering edges, we exploit the photo-consistency constraint by texturing the reference images over the scene's surface and measuring how well this textured scene-graph recreates the reference views. This has three advantages over Façade. Firstly, the user is not required to correlate edges in the reference views with edges in the scene-graph. More importantly, reprojection error is measured over the *entire* surface of the scene-graph rather than just its edges. This would allow reconstructions in views where there are insufficient primitive edges, but sufficient surface that can be used to measure photo-consistency. (A close-up of the edge of a building is an example of this case.) The most significant advantage, however, is our algorithm's ability to model occlusion, assign radiance to the hypothetical surface and compute the surface's photo-consistency metric *entirely on graphics hardware*. This leverages the inherent parallelism from an otherwise idle computational resource.

We have previously investigated using a scene graph to transfer pixels from one reference image to another as a measure of photo-consistency [1]. This approach projected one image onto the scene and rendered the result from a second view. The set of pixels visible to both cameras was used to determine the photo-consistency of the hypothesised parameters; this process was repeated for all stereo pairs of reference views. We discovered that the disadvantage of this approach was its tendency to push the scene parameters so the surface moved outside the visual hull. Since the photo-consistency metric can only compute the consistency of visible pixels, surface elements outside the visual hull cannot contribute to the photo-consistency error and therefore the scene cannot be invalidated. This visibility bias cannot be overcome by penalising occluded pixels because we assume the scene has significant occlusion. Furthermore, surface based visibility constraints cannot be applied in this framework because only the *projections* of surface points were compared and thus there was no mechanism to monitor the projection of a scene-point beyond a given stereo pair. We reasoned that the next step was to consider colour consistency over the scene surface while simultaneously using all reference images. This would allow constraints on the visibility of objects—for example, imposing a constraint that every point on the surface must be seen by at least two cameras. This penalty would invalidate parameters that push the geometry outside the visual hull.

This paper describes an approach to find parameters for a user-supplied scene-graph that best fit the reference images. Our approach uses photo-consistency to measure the fitness of a hypothesised scene configuration. It works by assigning radiance to the scene's surface and comparing the average radiance from all views with the contribution from *each* view. Space-carving also uses the idea of photo-consistency to evolve a surface hypothesis, but it considers

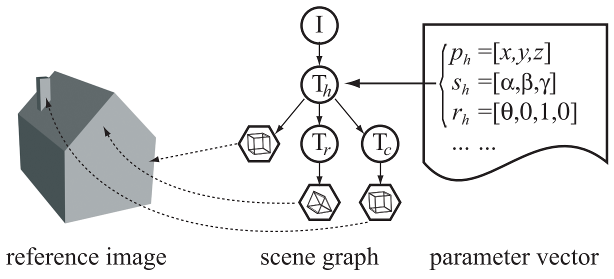


Figure 1. A reference image and its scene-graph provided by the user.

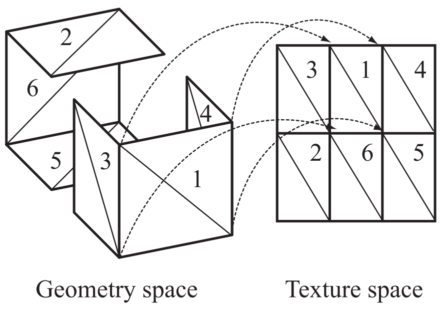


Figure 2. Texturing

only a single voxel at a time, and therefore makes a decision independent of its effect on the rest of the geometry. Our approach is different: it considers the global effect of changing a scene parameter because photo-consistency is simultaneously measured over the entire surface. Repeated application of our approach to different scene configurations allows us to draw conclusions about the scene's shape. Computing the photo-consistency over the entire surface is made tractable by exploiting graphics hardware's ability to generate synthetic images from geometry.

2 The Scene Graph

The input to the system is the set of reference images and an acyclic parameterised scene graph provided by the user. The scene graph describes how the reference geometry is constructed from primitive shapes such as boxes, pyramids and spheres. Geometry nodes are connected to a series of parameterised transformation nodes such as translation, rotation and scale. Each geometry node has a model-view matrix to transform its local coordinate system to the world coordinate system; it is given by the post-multiplication of the transformations along the path from the root node to the given geometry node. The model-view matrix is dependent on the scene-graph parameter vector, for associated with each transformation node is a set of parameters to model the unknown transformations. Not all transformation parameters are necessarily unknown—a primitive might be constrained to a plane or rotate about only one axis, for example. In these cases, the structure of the graph places constraints on the relationships between as their stacking order and alignment. Furthermore, the parameters may be constrained by expressions in terms of other parameters such as limiting the wings of an air-plane to some constrained set of suitable ratios.

Figure 1 is an example scene-graph for a house reference view. It consists of a box to model the house's base, a prism to model the roof and a box to model the chimney. The roof and chimney primitives are connected to the house base's transform node so they will inherent any changes to the base's transform. The transform applied to the roof is given by IT_hT_r, where T_h is the transform applied to the base and T_r is the roof's local transformation. The house base transformation could be described by scale, translation and rotation about the vertical axis while the roof transform would only require one scale factor to represent the height of its apex. The roof transform would also have a translation using the house's vertical scale to ensure its base is always coplanar with the top of the house base.

Each geometry node in the scene-graph is a renderable, fixed-size triangle mesh. Associated with each vertex is a position in its local coordinate system and a coordinate in texture space. A mesh's texture map is somewhat arbitrary (since there is not always a "good solution" [14]); the only constraint is that no two triangles in texture space can overlap. A triangle mesh can be defined to use any number of textures spaces. This property would be particularly important for meshes that are quite large in the reference views. An example texture map used in our system is illustrated in Figure 2: the advantage of this approach is it uses one texture space, yet the entire texture space is used.

Each triangle mesh has a model-view matrix defined by the path from the roof node to the given geometry node. Throughout the rest of the paper, we will use the term 'world triangle' to refer to the triangle in the local coordinate system transformed by the model-view matrix and 'texture triangle' to denote the triangle defined in texture space. 'Triangle' simply refers to the structure containing both triangles in the scene graph.

3 The Photo-Consistency Metric

The set of textures used in the scene-graph defines the structure's surface. The challenge presented by the photo-consistency constraint is to find suitable texture images and scene-graph parameters so the model generates the refer-

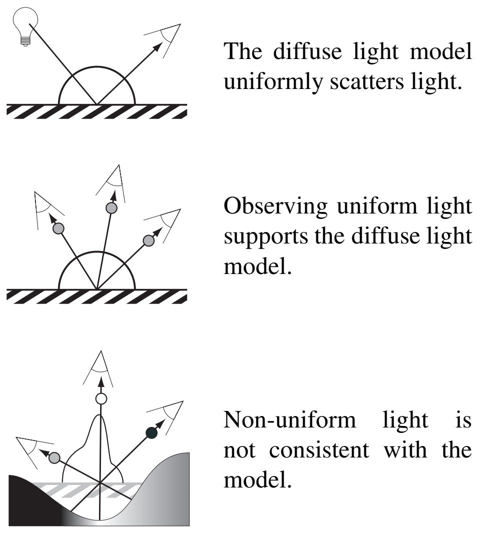

The diffuse light model uniformly scatters light.
Observing uniform light supports the diffuse light model.
Non-uniform light is not consistent with the model.

Figure 3. Photo-consistency is a property held by real scenes and is used to verify a hypothetical surface.

ence images when imaged from the reference camera positions. By definition, the generative model has photo-consistent textures. A point in texture space (or, *texel*) is photo-consistent if can be assigned a radiance function that models the point's projection in all reference views. This vector is usually a single colour to fit the diffuse light model, although more complicated models such as Phong [7] and Blinn [2] could be used.

If we restrict the class of reference scenes to those that can be modelled by only diffuse light, then the unoccluded projection of a point should be similar colour in every image. Consequently, a point that does not project to a similar colour in each image is said to be *inconsistent* and reasoned to not be part of the reference scene (see Figure 3). We exploit this colour constraint by constructing a view-dependent surface texture for every camera and comparing its consistency with the composite texture created by averaging each texel in the images where it is visible.

4 Computing Photo-Consistency

Given a reference image and hypothesised scene configuration, a view-dependent texture is created by back-projecting the reference image onto the scene. Every surface point visible in the reference image is assigned the radiance under its projection in the reference image; otherwise the point is classed as occluded and its radiance is undefined. Given a view-dependent texture from each reference image, the *composite* is computed by averaging the radiance assigned to each point. This process results in two maps: the first is the average radiance of each point's projection in all images that can see it, and the second map identifies the number of cameras that contributed to the point's average radiance.

If the geometry and scene parameters are correct, then every visible point will have a similar colour to every reference view. Consequently, inconsistencies between the composite texture and a reference view suggests the scene parameters are incorrect. The consistency of the composite texture to a view-dependent texture is given by the distance in colour space of every corresponding texel. This metric is used to drive an optimisation process to find the set of scene-parameters that maximise consistency with respect to all images. An overview of this process is illustrated in Figure 4.

4.1 View-Dependent Textures

The view-dependent texture is created by transferring radiance from the reference image to every triangle in the texture. This is implemented on graphics hardware by rendering each texture triangle and using its world triangle projected into the reference image as its texture map. The texture coordinate is the world coordinate transformed by $T = HPM$, where P is reference camera's projection, M is the model-view matrix associated with the triangle and H is the homography to map image space to texture space. The world triangle's projection in the reference image is mapped to the texture triangle by

$$\mathcal{T}(c_i) \longleftarrow r_{\mathcal{I}}(HPMv'_i). \qquad (1)$$

Here, $r(.)$ is a radiance function that returns $[p, \alpha = 1]$ (where p is the pixel on the reference image $\mathcal{I}$) if the point v_i is not occluded, and red = green = blue = α = 0, otherwise. Occlusion is with respect to the hypothetical scene structure and reference view. It is implemented in graphics hardware by rendering the scene from the reference viewpoint and binding its depth buffer as a shadow map [10]. An example of a view-dependent texture is illustrated in figure 5.

4.2 The Composite Texture

The composite texture is built from the set of view-dependent textures by blending each texture layer in the frame-buffer. The composite colour for a texel at $[u, v]^\top$ is the average colour from the set of corresponding texel $\{\mathcal{T}_i(u, v) \mid \mathcal{T}_i(u, v)_\alpha \neq 0\}$. Computing the composite texture in graphics hardware is readily implemented with $2n + 1$ render passes. Each texture is rendered to the frame-buffer to count the number the number of samples per texel, where each pass scales the texture's α-channel by n^{-1}. The

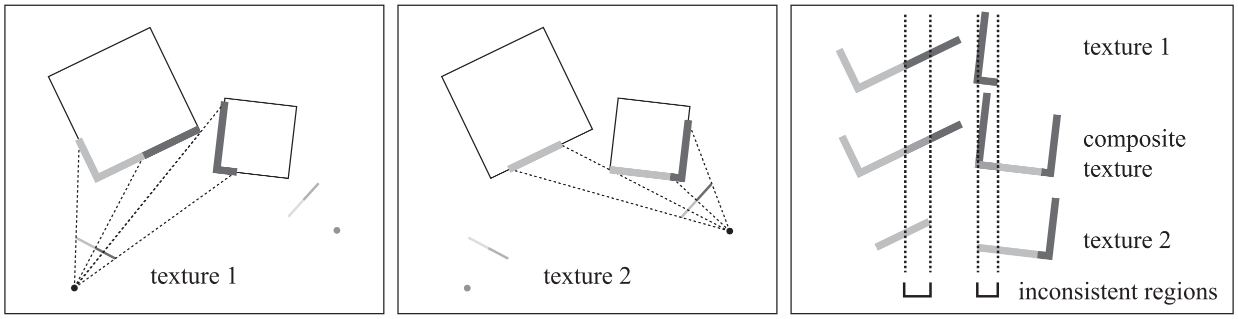


Figure 4. Photo-consistency is determined by constructing a view-dependent texture for each view and comparing it with the composite texture.

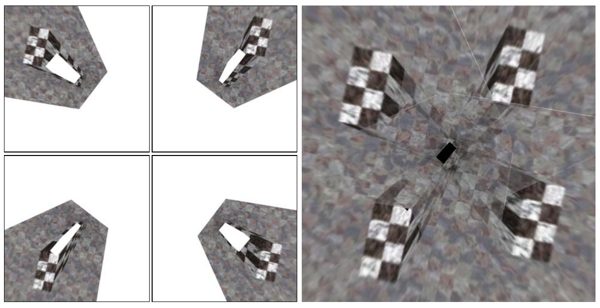


Figure 5. The view-dependent textures and the composite texture.

frame-buffer's α channel is then inverted with a fragment program; and, finally, the textures are rewritten to the frame-buffer and blended with the source and destination weights taken from the frame-buffer α-channel (as α and $1 - \alpha$ respectively) [9]. Figure 5 is an example composite texture from four provided view dependent textures.

4.3 Photo-Consistency and the Visibility Constraint

The composite texture is simultaneously the union of points seen in only one reference image and the average observation for points seen in multiple images. To determine the photo-consistency of the composite texture with respect to the reference images, it is compared with each view-dependent texture. The consistency of a texel is given by the average difference between the composite texture and each view-dependent texture; but a penalty is applied if less than an *a priori* number of views can see the texel. Assuming every scene point is seen by at least two views prevents the optimiser from pushing the geometry outside the visual hull.

4.4 Removing Error from Sealed Surfaces

While the visibility constraint penalises geometry that is not visible by sufficient number of cameras, some parts of the surface are not visible by design. For example, the top of the house base is coplanar with the bottom of the roof for the scene in Figure 1—these surfaces are hidden because they are part of primitive objects that are stacked to build more complex shapes. If left unchecked, these self-occluding surfaces erroneously contribute to the occlusion penalty and could lead to some tolerance of photo-inconsistency to minimise self-occlusion. Removing the occlusion penalty from coincident surfaces will treat the scene as a contiguous surface, giving an accurate understanding of the surface's occlusion.

Coincident surfaces are identified by rendering the scene for each triangle in the scene-graph from its point of view. At each iteration, the scene is transformed so a given triangle maps to its texture triangle in along the X/Y plane. This map is decomposed into a transformation that maps the world triangle to the X/Y plane followed by an affine homography to further map its world triangle to its texture triangle. With clipping planes configured to clip to the edge of the textured triangle, points along the triangle coincident with the scene-graph are marked by rendering the scene and accepting fragments with $z = 0$.

4.5 Accumulating Photo-Consistency

A photo-consistency map is generated for each iteration to compare a view-dependent texture with the composite

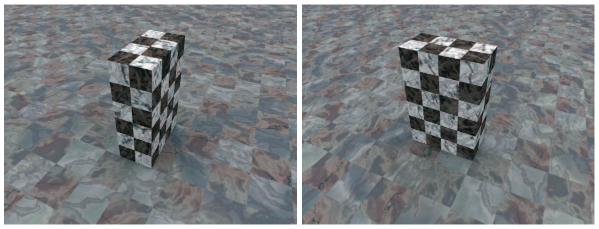


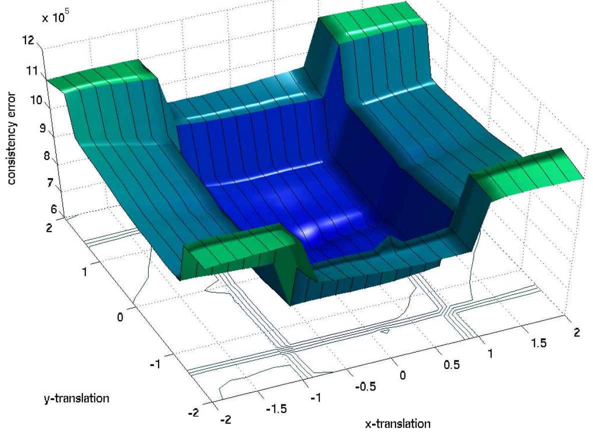
Figure 6. Two reference images used in the synthetic tests.

texture. Each texel in this map represents the consistency of its corresponding surface element with respect to a single reference view. The average consistency over the set of reference views is accumulated in the frame-buffer by blending each view-dependent map toegther. The per-pixel visibility penalty, applied to every texel by the process in Section 4.3, is removed from coincident surfaces by the process in Section 4.4 while rendering the scene with material $\alpha = 0$. The result stored in the frame-buffer's α channel is the per-texel photo-consistency; it is copied to system memory and summed to give the photo-consistency for the texture. This process is repeated and summed for all textures in the scene-graph.

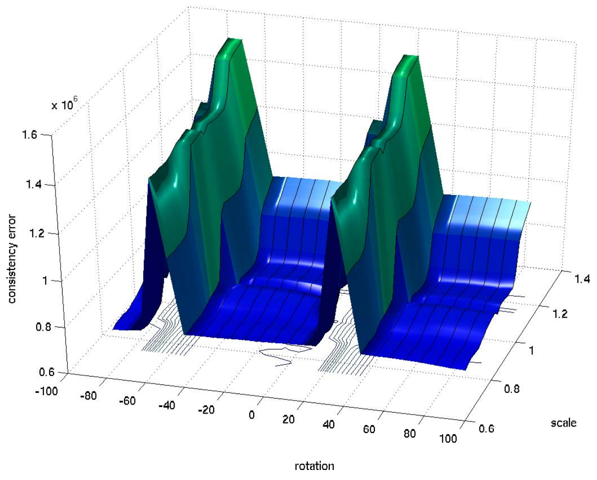
5 Results

The photo-consistency cost function was implemented with OpenGL 1.5 to determine how it behaves when the scene-graph parameters are changed. Synthetic tests were used to provide ground truth as we are still evaluating whether this approach is valid. We constructed a synthetic test involving four cameras and a box on a plane where the cameras orbit the box. Every point on the box is visible by at least two cameras. The reference images were generated with the Persistence of Vision ray-tracer; two of these images are illustrated in Figure 6.

The test scene has four parameters: the box's location along the plane, its rotation, and scale. (Scale can be represented as a single parameter since the box's texture suggests its dimensions are $4s \times 6s \times 2s$, where s is the size of the squares on its surface.) We defined the world coordinate system so the box is at its origin, has zero rotation about the y-axis and a dimension of $2 \times 3 \times 1$ units. We assume the cameras have been calibrated (although changing camera parameters is certainly possible in our system and will be investigated in the future). We performed a number of tests to determine how the photo-consistency metric behaves as scene parameters change. Some of the parameters are assumed to be known in these tests so that the graphs represent cross-sections of four-dimensional space.



(a) Translation with known rotation and scale. Truth is $x = 0$ and $y = 0$.

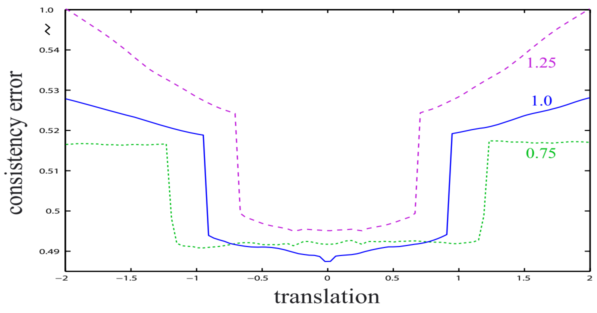


(b) Rotation vs scale with known translation. Truth is at $\theta = 0$ and $s = 1$.

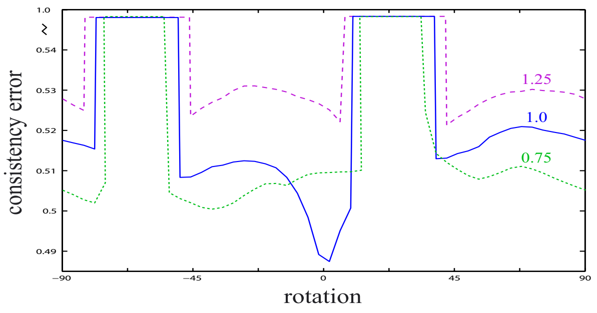
Figure 7. Exploring the photo-consistency metric over a range of scene parameters.

Figure 7(a) illustrates the effect of changing the box's translation while assuming rotation and scale are correctly known. Figure 7(b) illustrates the effect of changing the box's rotation and scale while assuming the translation is correctly known. The peaks in both graphs are caused by the visibility constraint; both graphs are minimised at the true solution. Figure 8(b) illustrates three cross-sections taken from the rotation/scale graph to clearly demonstrate the effect on rotation at various scales and that the cost is minimised at the true solution.

Figure 8(a) illustrates the effect of translation along one axis when scale is incorrectly known. The photo-

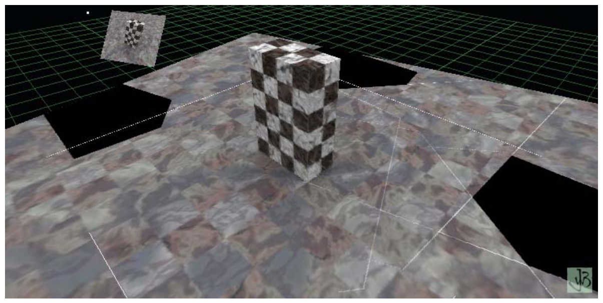


(a) Three cross-sections from Figure 7(a)

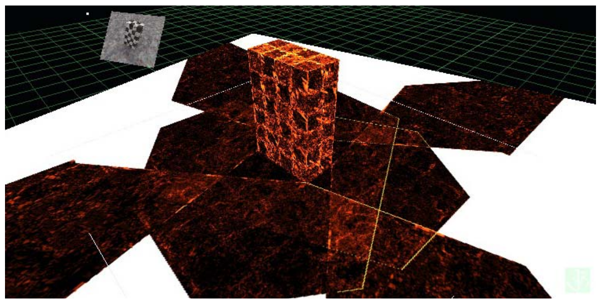


(b) Three cross-sections from Figure 7(b)

Figure 8. Photo-consistency cross-sections.



(a) A novel view of the reconstructed scene. (One of the reference cameras is visualised in the top left corner.)



(b) The photo-consistency error surface seen from the same view-point as 9(a). Increasing shades of red represents increasing photo-inconsistency.

Figure 9. The optimised scene configuration.

consistency metric is still minimised at truth (scale=1). The graph when the box's scale is under-estimated is linear when the box is visible because there is less texture variation over the box's surface.

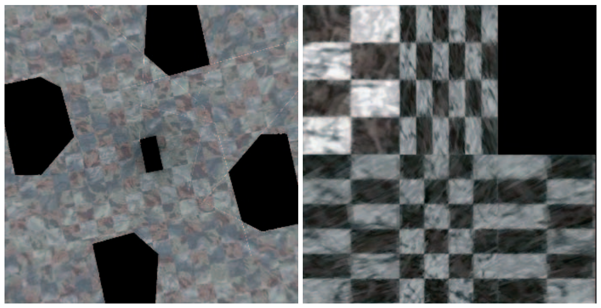
5.1 Synthesising Novel Views

The reconstructed scene can be imaged from novel view points by applying the composite texture over the surface. Figure 9(a) is an example novel view where the scene is textured with its composite texture. The black holes in the ground texture are regions not seen in any of the reference views. Figure 9(b) is the same view point but textured with its error surface. Here, black regions represent areas of high photo-consistency, white represents regions of low photo-consistency, and shades of dark red through to yellow represents increasing inconsistency.

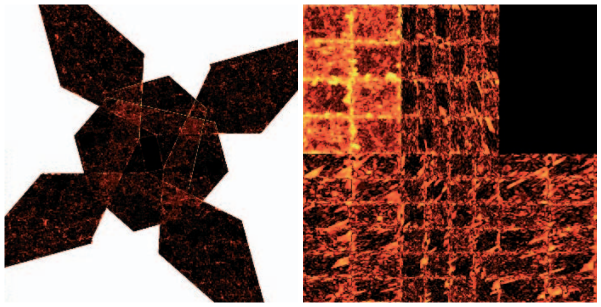
The unwrapped composite and photo-consistent texture space is illustrated in Figures 10(a) and 10(b) respectively. Figure 10(b) indicates that the cube's texture space has a large, seemingly photo-consistent region in the top-right of its texture space—this region comes from the surface that is coplanar with the ground-plane and is therefore always considered photo-consistent. In contrast, the ground-texture has more photo-inconsistent regions in Figure 10(b) than what its composite texture suggests: for example, the corners of the ground texture are clearly the grey chequered ground, yet the corresponding regions in the photo-consistency texture suggest these regions are inconsistent. This inconsistency is caused by enforcing the visibility constraint since these regions are only visible in one camera.

5.2 Texture Aliasing

Our experiments show that the best reconstruction has apparently significant levels of photo-inconsistency (illustrated in Figure 10(b)). This is true even when the known parameters are used; it is caused by aliasing in texture-space as the reference images are back-projected onto the surface. The error introduced through aliasing may affect complicated scene-graph configurations. This is not evident in our experiments, largely because the size of the box in the images is relatively large. Experiments with blurring the reference images somewhat mitigated the aliasing but did not dramatically change the shape of the consistency manifold. Future work would investigate the effect of aliasing on more complicated scenes and develop better sampling algorithms to limit its effect.



(a) A reference image and its scene-graph provided by the user.



(b) The reconstruction error texture.

Figure 10. The textures from the optimised scene configuration.

6 Conclusion

There are a number of advantages to using a graphics-based approach towards recovering scene information. The key advantage is that changing parameters has a global consequence to the surface's photo-consistency metric. This approach shifts the problem from image-processing into one where changes are made in scene-space. Previous work with photo-consistency evolved a volumetric structure by considering one voxel at a time. Consequently, decisions based on a voxel's photo-consistency are independent from decisions made to its neighbours. Our approach considers photo-consistency over the entire surface. We consequently recast the problem in terms of finding a scene-graph parameter vector that best generates the reference images. This has two advantages: firstly, reconstructions are spatially coherent because photo-consistency is measured over the entire surface; and, secondly, our approach can be accelerated using graphics hardware. Whereas our previous cost-function was unable to properly account for occlusion, the cost-function presented in this paper does not push geometry outside the visual hull, provided that a conservative estimate of a surface's visibility can be made. Our experiments demonstrate that our cost-function minimises the scene-parameters for simple scenes.

References

[1] J. W. Bastian and A. J. van den Hengel. Computing image-based reprojection error on graphics hardware. In *Proc. of the 7th Biennial Australian Pattern Recognition Society Conference*, 2003.

[2] J. F. Blinn. Models of light reflection for computer synthesized pictures. *ACM Computer Graphics SIGGRAPH*, 19(10):542–547, 1977.

[3] W. B. Culbertson, T. Malzbender, and G. G. Slabaugh. Generalized voxel coloring. In *Workshop on Vision Algorithms*, pages 100–115, 1999.

[4] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. Technical Report TR692 Computer Science Dept., U. Rochester, 1998.

[5] K. N. Kutulakos and S. M. Seitz. What do photographs tell us about 3d shape? Technical Report TR692, Computer Science Dept., U. Rochester, 1998.

[6] C. J. T. P. E. Debevec and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series):11–20, 1996.

[7] B. Phong. Illumination for computer generated images. *Communications of the ACM*, 18:311–317, 1975.

[8] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(26):314–319, 1985.

[9] T. Porter and T. Duff. Compositing digital images. In *Proc. Computer Graphics and Interactive Techniques*, 1984.

[10] M. Segal, C. Korobkin, J. Foran, and P. Haeberli. Fast shadows and light effects using texture mapping. In *Proc. Computer Graphics and Interactive Techniques*, 1992.

[11] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *Proc. CVPR*, pages 1067–1073, 1997.

[12] V. N. Smelyansky, R. D. Morris, F. O. Kuehnel, D. A. Maluf, and P. Cheeseman. Dramatic improvements to feature based stereo. In *Proc. European Conference on Computer Vision*, 2002.

[13] C. J. Thompson, S. Hahn, and M. Oskin. Using modern graphics architectures for general-purpose computing: a framework and analysis. In *Proc. Computer Graphics and Interactive Techniques*, 2002.

[14] A. H. Watt. *3D Computer Graphics*. Addison Wesley, 3rd edition, 1999.