A Probabilistic Framework for Space Carving

A. Broadhurst, T.W. Drummond and R. Cipolla
Department of Engineering
University of Cambridge
Cambridge, UK CB2 1PZ

Abstract

This paper introduces a new probabilistic framework for Space Carving [9]. In this framework each voxel is assigned a probability, which is computed by comparing the likelihoods for the voxel existing and not existing.

This new framework avoids many of the difficulties associated with the original Space Carving algorithm. Specifically, it does not need a global threshold parameter, and it guarantees that no holes will be carved in the model. This paper also proposes that a voxel-based thick texture is a realistic and efficient representation for scenes which contain dominant planes. The algorithm is tested using both real and synthetic data, and both qualitative and quantitative results are presented.

1. Introduction

Three-dimensional reconstruction has attracted considerable attention, and is still a very active topic for research. The goal of reconstruction is to recover the three-dimensional shape of a static scene viewed from a number of different cameras. Many different representations have been suggested, but still no single solution has been developed that covers all types and configurations of input images.

The earliest attempts [10] to solve the reconstruction problem posed the solution using image matching. These correlation based algorithms (e.g. [10, 7, 15]) attempted to solve the 2-D disparity field which described the relationship between two images. In this representation, when there is a change of depth, there is always an associated occlusion. But, in all these algorithms the factor limiting the complexity of the reconstruction is the depth parameterisation of space $z(x, y)$.

More recently, a second class of algorithms have focused on using Euclidean (or projective) representations for space. A common approach is to use piecewise planar polygons, which can be made significantly more realistic by using photographic and bas-relief texture. This was demonstrated by [5], and further developed by [1] who used layered depth images to efficiently represent 3-D scenes. The algorithm in this paper uses a layer-based implementation.

A third class of algorithms are the volumetric representations, which may use voxels [14, 13, 9, 4], or level sets [6]. The voxel-based algorithms make no attempt to model the continuity of shape, rather, they model the volume as an array of 3-D voxels. By not making assumptions about planarity and continuity, the voxel algorithms are able to cope with significantly more complex structures. The difficulty with these approaches, however, is that by ignoring the regularising assumptions they become more susceptible to noise.

In this paper a novel probabilistic framework for the space carving algorithm will be presented. Many ideas from the original Space Carving [9] framework will be retained but, significantly the existence of a voxel will not be a binary function. Instead each voxel will be assigned a probability for it existing in the model. This is different from the work of [14, 2], because the probability is used to describe the existence of the voxel, and is not used to infer opacity. It will be shown that by using a probabilistic framework, two major failings of the Space Carving algorithm can be avoided. The first advantage is that no global parameters are required. (Space Carving requires the user to specify a global variance.) The second advantage, is that it can be guaranteed that the algorithm will not carve holes in the model, which is a well known failing of the Space Carving algorithm.

2. Background

In the Space Carving [9] framework, space is represented by an array of voxels, which must enclose the entire scene. At each iteration the algorithm selects a voxel and projects it into all the images where it is visible. It then asks the question; could this voxel have been part of the model? If the answer is no, then voxel is removed. This process is repeated until the algorithm has exposed the 3-D shape of

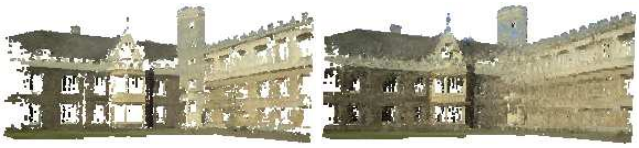

Figure 1. These images were generated using the original Space Carving algorithm [9] with two different thresholds. Notice how the windows have been incorrectly carved in both models, and how the threshold is too high in the right image to recover any detail.

the scene.

One difficulty with this algorithm is that if a voxel is removed in error, the algorithm will punch a hole all the way through the model. This erroneously removes further voxels, which may lead to a cascade effect. In practice a large region of the model may be removed. This paper will solve this problem by imposing the constraint that every ray must intersect at least one voxel.

A crucial part of the Space Carving algorithm is the consistency function, which is the mechanism that decides whether a voxel should be kept or discarded. Originally , only the centroid of a projected voxel was considered [9], and the voxel was kept if the RGB variance was less than a global threshold. A later refinement introduced R-shuffles [8] to allow for calibration error, but this still did not allow for any image noise, or lighting errors. The effect of noise can be reduced by using a statistical consistency function [3], but voxels may still be removed erroneously.

3. Framework

In this section the basic outline of the probabilistic framework will be described. There are many implementation issues which will need to be considered, but these can be left to the following section.

Indices	
i	Image index $\in \{1 \dots n\}$
x, y	Pixel co-ordinate index
k, l, m	Voxel index (m is depth)
Image data	
$\mathcal{I}^i$	Image i
$\mathcal{I}^i_{xy}$	Pixel x, y of image $\mathcal{I}^i$.
Geometric variables	
$\mathbf{P}^i$	Projection matrix for image $\mathcal{I}^i$
$\mathbf{H}^i_m$	Homography from plane m to $\mathcal{I}^i$
Statistical variables	
D	All available data $\{\{\mathcal{I}^i\}, \{\mathbf{P}^i\}\}$
$\exists_{klm}$	$\exists_{klm} \in \{0 =$ voxel removed, $1 =$ voxel exists$\}$
$\mathcal{V}_{klm}$	The model for a voxel which exists at k, l, m.
$\mathcal{W}^i_{klm}$	The missing voxel model, for image i.

Table 1. Notation used in this paper.

3.1. Modelling the projection of a voxel

This algorithm will use two models to describe a voxel. The first model will describe what the projection of a voxel looks like in the image, and a second model will describe what an image looks like when a voxel is removed. Later Bayes' rule will be used to infer which case is more likely.

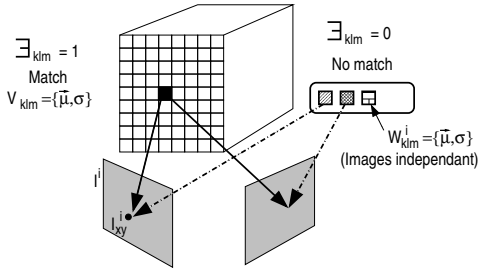

Figure 2. The model for a voxel existing ($\exists_{klm} = 1$), and not existing ($\exists_{klm} = 0$).

The first possibility is that the voxel indexed by klm is present in the model (denoted $\exists_{klm} = 1$), in which case the image data corresponding to the projection of the voxel will be described by the voxel model $\mathcal{V}_{klm}$. This is true for all views that are not occluded. Each voxel will be represented by a spherical Gaussian distribution in RGB space. This means that $\mathcal{V}_{klm}$ has four degrees of freedom that need to be estimated, which are $\{\mu_R, \mu_G, \mu_B, \sigma^2\}$.

The second possibility is that voxel klm does not exist (denoted $\exists_{klm} = 0$). In this case when the non-voxel is projected into each of the images, the image samples will be from *different* voxels. The difficulty here is that until the entire model is known, it will not be know which voxels these were. To get around this difficulty, it will be assumed that each sample is locally independent. This means a missing voxel can be represented by a set of independent models $\{\mathcal{W}^i_{klm}\}$ (one for each image). Again, each of the models will be represented by a spherical Gaussian $\{\mu_R, \mu_G, \mu_B, \sigma^2\}$.

3.2. How to make a decision about a voxel

The probability of a voxel existing $P(\exists_{klm} = 1 \mid D)$ is determined using Bayes' theorem. The voxel and independent models are used to compute the likelihoods of the data given the models. The prior probability that a voxel exists is denoted $P(\exists = 1)$.

$$P(\exists_{klm} = 1 \mid D) = \frac{P(D \mid \exists_{klm} = 1) P(\exists = 1)}{P(D \mid \exists_{klm} = 1) P(\exists = 1) + P(D \mid \exists_{klm} = 0) P(\exists = 0)} \quad (1)$$

The two likelihood terms are written as a function of the image data, and the model parameters. For the independent

model case these are $\{\mathcal{W}^i_{klm}\}$, and are marginalised:

$$P(D \mid \exists_{klm}=0) = \prod_i \int_{\mathcal{W}^i_{klm}} P(\mathcal{I}^i \mid \exists_{klm}=0, \mathcal{W}^i_{klm}) P(\mathcal{W}^i_{klm}) d\mathcal{W}^i_{klm} \quad (2)$$

A similar expression can be written for the case of a voxel existing. In this case, the voxel model $\mathcal{V}_{klm}$ is marginalised, which gives:

$$P(D \mid \exists_{klm}=1) = \int_{\mathcal{V}_{klm}} \prod_i P(\mathcal{I}^i \mid \exists_{klm}=1, \mathcal{V}_{klm}) P(\mathcal{V}_{klm}) d\mathcal{V}_{klm} \quad (3)$$

3.3. Visibility and occlusion reasoning

Equations (2) and (3) assume that every voxel is visible. This is only true for exterior voxels, as any other voxel may be occluded. Marginalising the existence of *all* closer voxels, however, is computationally infeasible, but an approximation can be made.

A voxel klm defines a ray to a camera $\mathcal{I}^i$, and only voxels along this ray effect visibility. This is a first order approximation. Of all the possibilities of these voxels existing or not, only two cases are of interest which are denoted $C^i_{klm} \in \{0 = visible, 1 = occluded\}$. The probability of a voxel being visible $P(C^i_{klm} = 0 \mid \{\exists\}, D)$ is computed by ensuring all the voxels along the path of the ray do not exist. Equation (4) assumes independence, which is the same first order approximation (as above).

$$P(C^i_{klm} = 0 \mid \{\exists\}, D) = \prod_{\mathbf{r} \in ray} P(\exists_{\mathbf{r}} = 0 \mid D) \quad (4)$$

In (3), the effect of occlusion was neglected. These equations can be rewritten for a particular set of visible images $\{C^i_{klm}\}$ by only including the visible images in the product:

$$P(D \mid \exists_{klm}=0, \{C^i_{klm}\}) = \prod_i \left(\int_{\mathcal{W}^i_{klm}} P(\mathcal{I}^i \mid \exists_{klm}=0, \mathcal{W}^i_{klm}) P(\mathcal{W}^i_{klm}) d\mathcal{W}^i_{klm} \right)^{C^i_{klm}} \quad (5)$$

The next step is to marginalise the visibility $\{C^i_{klm}\}$ by summing all the possible visibility assignments.

$$\begin{aligned} P(D \mid \exists_{klm}=0) &= \sum_{c^1 \in \{0,1\}} P(C^1_{klm}=c_1 \mid \{\exists\} D) \ldots \\ &\sum_{c^n \in \{0,1\}} P(C^n_{klm}=c_n \mid \{\exists\} D) P(D \mid \exists_{klm}=0, \{C^i_{klm}\}) \\ &= \sum_{c^1} \cdots \sum_{c^n} \prod_i P(C^i_{klm}=c^i \mid \{\exists\}, D) (\theta^i_{klm})^{C^i_{klm}} \\ \theta^i_{klm} &= \int_{\mathcal{W}^i_{klm}} P(\mathcal{I}^i \mid \exists_{klm}=0, \mathcal{W}^i_{klm}) P(\mathcal{W}^i_{klm}) d\mathcal{W}^i_{klm} \end{aligned} \quad (6)$$

Equation 3 can be modified in a similar manner:

$$\begin{aligned} P(D \mid \exists_{klm}=1) &= \sum_{c^1} \cdots \sum_{c^n} \prod_i P(C^i_{klm}=c^i \mid \{\exists\}, D) (\Theta^i_{klm})^{C^i_{klm}} \\ \Theta^i_{klm} &= \int_{\mathcal{V}_{klm}} P(\mathcal{I}^i \mid \exists_{klm}=1, \mathcal{V}_{klm}) P(\mathcal{V}_{klm}) d\mathcal{V}_{klm} \end{aligned} \quad (7)$$

To correctly marginalise the dependence of voxels, it would be necessary to integrate over all the visibility cases of all the other voxels. This is infeasible, and is replaced in this section by a first order approximation to marginalise the different visibility cases. This still has $O(2^n)$ complexity, so a brute force approach is still impractical, but a further $O(n)$ approximation is considered in Section 4.5.

3.4. Image sampling and independence

Consider the problem of estimating the likelihood of the data for a given voxel. This requires information from different images with different levels of detail to be combined.

Typically the resolution of the voxel array is similar to that of the images, but for a particular voxel it may be either slightly higher or lower than that of the voxel grid. The difficulty with this is that it is necessary to compute the fraction that each pixel contributes to the distribution of each voxel. This is difficult and time consuming. A better approach is to oversample the voxel array so that it can be guaranteed that each sample point is drawn from a single pixel. The oversampled plane corresponds to the front face of each voxel and is denoted S^{rs}_{klm}, where klm refers to the voxel, and rs are the co-ordinates of the sample point within that voxel.

There is a complication, however, in that pixels which project to a large area on the voxel will get counted many times. To solve this problem, the samples from each image are inversely weighted according to their area: $w^i = \frac{1}{detJ(\mathbf{A}^i_{xy})}$ where $\mathbf{A}^i_{xy}$ is the affine homography from the image to the front face of the voxel, and where $J(A)$ is its Jacobian. This means that a large pixel which spans several sample points will get a small weighting.

The voxel model has been assumed to be Gaussian, so $\mathcal{V}_{klm} = \{\vec{\mu}, \sigma\}$. The voxel model term in (7) can be marginalised as follows:

$$\begin{aligned} &P(\{\mathcal{I}\} \mid \mathcal{V}_{klm}, C^i_{klm}=0, \exists_{klm}=1) \\ &= \int\int \prod_{rs} P(S^{rs}_{klm} \mid C^i_{klm}=0, \exists_{klm}=1)^{w^i} P(\vec{\mu}) d\vec{\mu} P(\sigma) d\sigma \\ &= \int\int \prod_{rsi} \left(\frac{1}{(\sigma\sqrt{2\pi})^3} e^{-\frac{\|S^{rs}_{klm} - \vec{\mu}\|^2}{2\sigma^2}} \right)^{w_i} P(\vec{\mu}) d\vec{\mu} P(\sigma) d\sigma \end{aligned} \quad (8)$$

At this point the prior for $P(\vec{\mu})$ is assumed to be flat, and $P(\sigma) = \frac{1}{\sigma}$ is given a Jeffrey's prior. This is both mathemat-

ically convenient, and favours voxels with small variances, which agrees with the posterior data.

$$P(\{\mathcal{I}\} \mid \mathcal{V}_{klm}, C^i_{klm}=0, \exists_{klm}=1) = \frac{1}{2\sqrt{n}}\left(\frac{1}{\sigma_{\mathbf{x}}\sqrt{\pi n}}\right)^{3(n-1)}\Gamma\left(\frac{3(n-1)}{2}\right) \quad (9)$$

where $n=\sum w_i$ and $\sigma^2_{\mathbf{x}} = \sigma^2_{x_R}+\sigma^2_{x_G}+\sigma^2_{x_B}$ and $\sigma^2_{x_R} = \frac{1}{n}(\sum_{rsi} w_i x^2_R) - \bar{x}^2_R$. A similar expression can be written for the case of independent models (i.e. no match):

$$\begin{aligned} &P(\{\mathcal{I}\} \mid \{\mathcal{W}_{klm}\}, C^i_{klm}=0, \exists_{klm}=0) \\ &= \prod_i \iint \prod_{rs} P(S^{rs}_{klm} \mid C^i_{klm}=0, \exists_{klm}=0) P(\vec{\mu}) d\vec{\mu} P(\sigma) d\sigma \\ &= \prod_i \left(\frac{1}{2\sqrt{n}}\left(\frac{1}{\sigma^i_{\mathbf{x}}\sqrt{\pi n}}\right)^{3(n-1)}\Gamma\left(\frac{3(n-1)}{2}\right)\right)^{w_i} \end{aligned} \quad (10)$$

This can be expressed using log probabilities, with n being the number of visible images, as:

$$\log P(\{\mathcal{I}\} \mid \{\mathcal{W}_{klm}\}, C^i_{klm}=0, \exists_{klm}=0) = \sum_i w_i\left(a(n)\log\left(\sigma^i_{\mathbf{x}}\right)^2 + b(n)\right) \quad (11)$$

In summary, a voxel is defined by one of two models, depending on whether it exists in the model, or not. An expressions has been derived for estimating the parameters of both models, and for computing their likelihoods in a Bayesian framework. A technique for handling visibility and occlusion has been presented, but this is still computationally expensive, so a number of implementation issues will be addressed in the next section:

4. Implementation

In this section the implementation of the Probabilistic Space Carving algorithm will be discussed, but first, a number of practical issues need to be addressed:

4.1. Overview of the carving algorithm

The voxel array is processed using the plane sweep algorithm, starting with the layer of voxels closest to the viewer. After the probabilities for the first layer of voxels have been calculated, the probabilities are rendered into the alpha channel of each of the source images. When processing each of the subsequent layers, the probability of a voxel being occluded (in each image) is easily obtained by looking up the projected alpha value in the images. Again, after each plane is processed the rendered alpha values in each of the images are updated.

4.2. Processing layers for efficiency

Rendering a voxel array using triangles is very inefficient as there are 12 triangles per voxel. Silicon Graphics have overcome this problem in OpenGL, by defining a set of extensions for volume rendering. The voxel array is described as a set of texture mapped layers, with the alpha channel specifying the the opacity. Matching using layers is not a new concept in Computer Vision [14], however, in this paper the alpha channel will be used to store the probability of a voxel existing $P(\exists = 1 \mid D)$ not opacity.

4.3. Encoding probability

The alpha channel on most graphics platforms is only 8 bits wide, so an efficient encoding scheme is needed for storing the probability. This probability is computed using Bayes' rule (1), and can be expressed as:

$$P(\exists{=}1 \mid D) = \frac{1}{1+e^{(\log P(D\mid\exists=0) - \log P(D\mid\exists=1)+k)}} \quad (12)$$

where the prior $k = \log P(\exists{=}0) - \log P(\exists{=}1)$ is assumed constant. The alpha channel stores the difference between the log likelihoods and is insensitive to k.

4.4. Calculating visibility

The probability of a voxel being visible is given by (4). It is not necessary to compute the entire product for every voxel. After each plane has been processed the images can be used to cache the result of $P(C \mid \exists, D)$ for all planes up to the one being processed. This is a significant speed improvement.

Consider one pixel in an image. This pixel defines a ray in space, which intersects with one voxel in each layer. To find the current estimate for $P(C \mid \exists, D)$, the product along each ray is calculated. Since this ray is defined by a single image pixel, the α channel of the source image can be used to cache the accumulated probability $\prod P(C \mid \exists, D)$. After each new layer of voxels is carved, the cached values need to be updated. This is implementing using two alternating phases. In the first phase a layer of voxels is processed, and in the second phase the cached visibility values are updated (in the image plane). This is repeated for each layer.

4.5. Marginalising visibility

To calculate the probability of a voxel existing, the probability given each of the possible visibility assignments must be computed (see equation 7), and the visibility marginalised. This is very slow as it has complexity $O(2^n)$. Instead a local threshold β is introduced and view $\mathcal{I}^i$ is included in the integral if $P(C^i_{klm}=0 \mid \exists, D) > \beta$. The

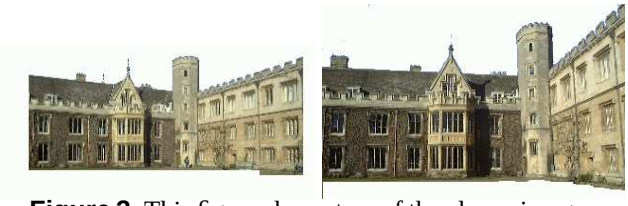

Figure 3. This figure shows two of the eleven images of the Master's Lodge, Trinity College, Cambridge

parameter β is then varied between $0 \ldots 1$, and the maximum likelihood value of $P(\exists \mid C, D)$ is found.

Since the number of images is usually significantly lower than the number of possible α values, it is a good idea to break the search up into a small number of β ranges. In this way one can guarantee the visibility to stay the same over each of the ranges.

This approximation selects the most likely cases for 2 views, 3 views, etc. All other visibility cases are assumed to be unlikely. This approximation can be checked as the total likelihood of all the visibility cases should be one.

4.6. Approximations for rendering

This section describes how an image from a new viewpoint is rendered. The best approach is to integrate along each ray marginalising the probabilities of each voxel using Bayes' rule. This assumes that there is one voxel visible along each ray, but cannot be implemented using OpenGL.

$$\mathbf{x}_{rgb} = \frac{\sum_{r \in ray} \mathbf{x}_r P(\exists_r = 1)}{\sum_{r \in ray} P(\exists_r = 1)} \qquad (13)$$

If Equation (13) is approximated by a maximum likelihood estimate, then an OpenGL implementation is possible. The model is repeatedly rendered using the α-test, with an increasing probability threshold. Alternatively, a conventional 3-D model can be generated by classifying each voxel as either existing, or not; by selecting the best voxel along each ray. This technique was used to generate Figure 4.

5. Results

The algorithm is demonstrated in both a qualitative and quantitative manner. Figures 4 and 5 were generated using the *Master's Lodge* [3] and the *Castle* [11] image sequences. The rendered views are new viewpoints, which were not used during the reconstruction process. The ground truth is shown in Figure 5 for comparison. Notice how the entire image including the background and trees has been reconstructed.

A quantitative analysis has been performed using both real and synthetic data. Figure 6 shows the performance of the original Space Carving algorithm and the probabilistic approach, by plotting the number of missing pixels (holes) against reconstruction error. Notice how the new approach never carves holes in the model, and how difficult it is to select an optimal threshold parameter in the original method.

Figure 4. This figure shows two rendered images from unseen viewpoints. The front view uses Bayesian rendering, and the top view is generated using a M.L. reconstruction. The model contains 63 planes with texture size 246×512.

Figure 5. This figure was generated using 16 images from the *castle* sequence. The model contains 116 planes with texture size 256×159. The left image is the new viewpoint, and the right image is the withheld view.

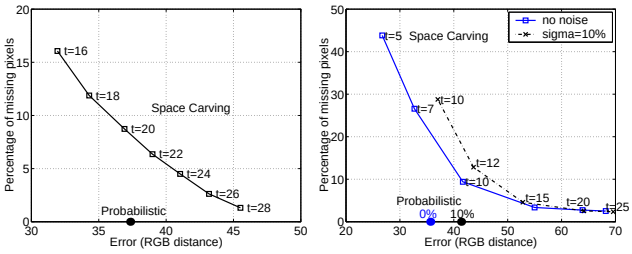

Figure 6. These two graphs compare the performance of the probabilistic approach with the original Space Carving algorithm. Each data point was generated by calculating the pixel error between an unseen ground truth image, and a reconstructed image. The right hand plot uses synthetic data, and shows the effect of noise. The left plot uses the *Master's Lodge* image sequence. The parameter t (0..255) is the global variance threshold of the original Space Carving [9] algorithm. The probabilistic framework has no parameters.

5.1. Discussion

Figure 6 shows how significantly the quality of the reconstruction is affected by the threshold parameter in the original Space Carving algorithm. This ambiguity is eliminated using the probabilistic approach. In particular, the algorithm never generates holes in the model. This has two side-effects. Where previously there would have been holes, the new algorithm renders using a low probability voxel. This can lead to higher errors in the rendered images around these regions. A second difficulty is that if the background is not accurately segmented, then the algorithm will not find holes where there should be holes. In many cases, such as the *Castle* sequence, background segmentation is not necessary.

5.2. Application: Thick texture

Many scenes can be broken up into two or three dominant planes [1, 12]. Typically, these planes contain additional detail. For example, a wall may contain bay windows or plants. These are difficult to represent using polygons, and bas-relief textures do not always capture all the occlusion relationships. In this example, two thin voxel arrays are used as *thick texture*.

Figure 7. This figure was generated using two thin voxel arrays (6 layers) positioned parallel to the walls.

6. Future Work

At present the algorithm has only been tested on image sequences that can be processed in a single sweep. In order to use more complex sequences the input images will have to be divided into subsets. Each orientation of the sweep plane will generate a different model, and a method is needed to combine the results to yield the maximum likelihood reconstruction.

7. Conclusion

In this paper a probabilistic framework for the Space Carving algorithm has been presented. In this framework there is no global variance parameter. The qualitative results show how the new approach does not carve holes in the model, although it may introduce a small amount of voxel clutter. Quantitative results have shown that the probabilistic approach generates reconstructed images with a lower intensity error than the original Space Carving algorithm.

References

[1] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 434–441, June 1998.

[2] J. S. D. Bonet and P. Viola. Poxels: Probabilistic voxelized volume reconstruction. In *Proc. 7th Int. Conf. on Computer Vision*, pages 418–425, Corfu, Greece, 1999.

[3] A. Broadhurst and R. Cipolla. A statistical consistency check for the space carving algorithm. In *Proc. British Machine Vision Conference*, volume I, pages 282–291, 2000.

[4] W. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *Proceedings of Vision Algorithms Theory and Practice Workshop*, pages 100–114, Corfu, Greece, 1999.

[5] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Special Interest Group on Computer Graphics*, pages 11–20, 1996.

[6] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDEs, level set methods, and the stereo problem. *Transactions on Image Processing. Special Issue on Geometry Driven Diffusion and PDEs in Image Processing*, 7(3):336–344, March 1998.

[7] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16:920–932, 1994.

[8] K. Kutulakos. Approximate N-view stereo. In *Proc. European Conference on Computer Vision*, pages 67–83, Dublin, Ireland, 2000.

[9] K. Kutulakos and S. M. Seitz. A theory of shape by space carving. *Int. Journal of Computer Vision*, 38(3):198–218, July 2000.

[10] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, vol.194:283–287, 1976.

[11] M. Pollefeys, R. Koch, M. Vergauwen, and L. V. Gool. Metric 3-D surface reconstruction from uncalibrated image sequences. *Proc. SMILE workshop (post-ECCV), Lecture Notes in Computer Science*, 1506:138–153, 1998.

[12] P.Torr, A. Dick, and R. Cipolla. Layer extraction with a Bayesian model of shapes. In *Proc. European Conference on Computer Vision*, volume II, pages 273–289, Dublin, Ireland, 2000.

[13] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *Int. Journal of Computer Vision*, 35(2):1067–1073, 1999.

[14] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *Proc. 6th Int. Conf. on Computer Vision*, pages 517–524, Bombay, India, Bombay, India 1998.

[15] O. V. Y. Boykov and R. Zabih. Fast approximate energy minimisation via graph cuts. In *Proc. 7th Int. Conf. on Computer Vision*, pages 377–384, Corfu, Greece, September 1999.