A FIRST ORDER PREDICATE LOGIC FORMULATION OF THE 3D RECONSTRUCTION PROBLEM AND ITS SOLUTION SPACE

MARTIN ROBINSON

Intelligent Real-Time Imaging and Sensing Group
School of ITEE, The University of Queensland, Australia QLD 4072
martin.robinson@defence.gov.au

KURT KUBIK

Intelligent Real-Time Imaging and Sensing Group
School of ITEE, The University of Queensland, Australia QLD 4072
kubik@itee.uq.edu.au

BRIAN LOVELL

Intelligent Real-Time Imaging and Sensing Group
School of ITEE, The University of Queensland, Australia QLD 4072
lovell@itee.uq.edu.au

This paper defines the 3D reconstruction problem as the process of reconstructing a 3D scene from numerous 2D visual images of that scene. It is well known that this problem is ill-posed, and numerous constraints and assumptions are used in 3D reconstruction algorithms in order to reduce the solution space. Unfortunately, most constraints only work in a certain range of situations and often constraints are built into the most fundamental methods (e.g. Area Based Matching assumes that all the pixels in the window belong to the same object). This paper presents a novel formulation of the 3D reconstruction problem, using a voxel framework and first order logic equations, which does not contain any additional constraints or assumptions. Solving this formulation for a set of input images gives all the possible solutions for that set, rather than picking a solution that is deemed most likely. Using this formulation, this paper studies the problem of uniqueness in 3D reconstruction and how the solution space changes for different configurations of input images. It is found that it is not possible to guarantee a unique solution, no matter how many images are taken of the scene, they're orientation or even how much colour variation is in the scene itself. Results of using the formulation to reconstruct a few small voxel spaces are also presented. They show that the number of solutions is extremely large for even very small voxel spaces (5x5 voxel space gives 10 to 10^7 solutions). This shows the need for constraints to reduce the solution space to a reasonable size. Finally, it is noted that because of the discrete nature of the formulation, the solution space size can be easily calculated, making the formulation a useful tool to numerically evaluate the usefulness of any constraints that are added.

Keywords: 3D Reconstruction; Solution Space; Uniqueness; Voxels; Logic

2   *Martin Robinson, Kurt Kubik, Brian Lovell*

1. Introduction

What do you see when you look at an apple? Well, hopefully you see an apple sitting in front of you. But how can you be sure that there is an apple there at all? Well, you could reach out and grab it, but assume for one moment that you only have your sense of vision.

In order to be sure that there is an apple there at all, there needs to be a unique solution to your vision task. That is, there needs to be only one possible three dimensional space corresponding to the two images which you see through your eyes.

Unfortunately, this is impossible, there will always be more than one solution to this problem. Simply think of the headsets in Virtual Reality systems. These place small LCD displays in front of each eye. If the correct images are fed into these then the user can be given the false impression that an apple is hovering in the air a few meters away. So, when you look at that apple, how do you know whether it is really there or if there are two little displays in front of your eyes?

Thus, we already have two solutions for the same vision perception. There can easily be more, hundreds or thousands of potential solutions, the exact number depending on the resolution of the 3D space (assuming a discrete spatial framework) and the exact images perceived. Two stereo images of an apple will have a considerably smaller solution space than, say, two stereo images showing only blackness.

Of course, these other solutions rarely trouble us in our day to day lives because we know from prior experience that if we see two stereo images of an apple, then there is most probably an apple sitting in front of us. Unfortunately, these other solutions do trouble us in the field of 3D computer vision. The problem of reconstructing a 3D object or scene from many 2D images is made difficult because the problem is an ill-posed one. The possible solution space for nearly all problems is huge!

In order to reduce the solution space for the 3D reconstruction problem, many constraints have been researched and implemented that attempt to reduce the solution space and improve the final reconstructed solution. Many constraints have arisen out of research into human 3D vision or through observations about the environment around us (e.g. The smoothness constraint, which forces any reconstructed surfaces to be smooth).

Many of the problems associated with 3D reconstruction methods come about because constraints and assumptions are used that do not hold for all situations (e.g. The smoothness constraint typically holds over the surface of a single object, but there are usually discontinuities in depth between multiple objects) . This problem is made worse by the fact that there are constraints and assumptions built in at a fundamental level in the most commonly used matching techniques. For example, the concept of matching corresponding pixels in order to calculate the 3D positions of the point they represent ignores monocular regions and the 3D information that

they can contribute (e.g. linear perspective, occultation, shading, shadow, texture[7]). As another example, windowing techniques used to match pixels assume that all the pixels inside the window belong to the same population of pixels (i.e. the same object). This problem can be eased through the use of Transform based methods [18] but the basic problem remains.

This paper presents a formulation of the 3D reconstruction problem using a voxel (3D volume elements with opacity and colour) framework and first order predicate logic equations. Since this paper is attempting to investigate only the theoretical properties of the 3D reconstruction problem, this formulation is purely theoretical. It does not take into account photometric distortions that yield corresponding points with quite different gray values or colours. Instead, it assumes that each pixel in the images has exactly the same colour as the voxel it projects upon.

2. Background and Related Work

2.1. *Volumetric Framework*

A digital image discretises the image space it occupies into areas called pixels. These pixels have position, size, shape and colour. A Voxel is simply a three-dimensional pixel. It is a volume in 3D space that is assigned a colour and an opacity. For a good overview of existing voxel-based reconstruction methods, see either Slabough's [16] or Dyer's [5] work.

A discrete or voxel representation of 3D space is convenient and commonly used when looking at stereo image matching. Marr and Poggio [12] formulated one of the earliest stereo matching algorithms using this representation. More recently, Intille and Bobik [8] [9] [2] created a stereo image representation called the *disparity-space image* or DSI.

Seitz and Dyer [15] also used a voxel representation of space, but they focused on many input images, rather than restricting themselves to stereo images. By placing constraints on the positioning of the camera to eliminate the problem of occlusions, Seitz and Dyer present an algorithm for reconstructing a 3D scene by finding *colour invariant* voxels using a technique called Voxel Colouring. The voxel space is traversed outward from the centre of the camera volume and each voxel is projected onto the input images. If the voxel projects upon inconsistent colours in the images, then it is deemed to be transparent. Otherwise the voxel is labelled with the colour of its projections and added to the set of colour invariant voxels. In this way a solution called the *photo hull* is built up. All the voxels on the surface of the photo hull are colour invariant and all the possible solutions of the 3D reconstruction problem (i.e. all photo-consistent shapes) are contained within the photo hull.

Kutulakos and Seitz [10] proposed a similar technique called Space Carving, so named because the colour inconsistent voxels are "carved" away to leave the photo hull behind. This technique places no restrictions on the placement of the cameras but it does place restriction on how many cameras are used at each iteration. The algorithm progresses by sweeping a horizontal plane through the voxel space,

carving away the colour inconsistent voxels as it goes. Only cameras behind the plane (i.e. those that have been carved out) are used to carve away new voxels.

Culbertson and Malzbender [4] built on Voxel Coloring and Space Carving to produce Generalized Voxel Coloring. This places no constraints on the cameras and uses all the cameras at every stage of the algorithm. On a side note, our paper paper uses the same concept of classifying voxels into *Surface*, *Transparent* and *Inside* voxels as was done in [4].

There have been many other algorithms developed that carve away colour inconsistent voxels to find the photo hull. Notably, Slabaugh, Malzbender and Culbertson [17] have created a Voxel Colouring algorithm that works on an infinite or semi-infinite sized voxel space. Saito and Kanade [14] have developed a variation that works on a projective grid space. Finally, De Bonet and Viola [3] attempt to bridge the gap between 3D computer vision and tomography by allowing their voxels to have both colour and variable opacity.

Similar to the idea of the photo hull is the *visual hull* [13]. A visual hull is constructed by calculating the intersection of all the silhouettes in all the input images. It too contains all of the solutions to the 3D reconstruction problem within its volume.

All of the techniques above aim to calculate a single shape (the photo or visual hull) that is the union of all the possible solutions of the 3D reconstruction problem. However, these hulls contain no information on the size of the solution space or the possible shapes that lie within the hull. This information can give valuable insights into the nature of the 3D reconstruction problem. For example, the size of the solution space can be a useful indicator of the quality of reconstruction. Obviously, the smaller the solution space, the more likely you are to pick the true shape. The formulation presented in this paper reconstructs the entire solution space, not just the photo or visual hull.

2.2. *Logical Formulation*

There is no known (to this author) formulation of low level 3D reconstruction that uses logical equations. Logic and Knowledge Based Modelling is used as a high level tool to classify individual object in the disparity maps and models calculated using traditional methods. Using a priori information, these maps and models can be improved by altering the representation of each classified object to better fit the known models [6] [11]. However, there has been no attempt to model the problem of 3D reconstruction using a logical formulation from the ground up, as is done in this document.

2.3. *Analysis of the Solution Space*

Kutulakos and Seitz [10] present a similar (to this paper) analysis of uniqueness in their paper on Space Carving. They acknowledge that trivial solutions like that

described in Section 5.3 and Figure 7 prevent the solution space from ever containing a single unique solution. They state that the implementation of free-space constraints (i.e. Constraining the solution to lie within a certain volume) can eliminate this problem. However, as our results in Section 6 show, this constraint does not do enough to reduce the solution space to manageable proportions.

Baker, Sim and Kanade [1] recently wrote an interesting paper asking when the shape of a scene was unique given its light-field. The light-field is a continuous function that specifies the radiance of light from the scene (see [1] for more details). They conclude that unless there are extended regions of constant intensity in the scene, there will always exist (theoretically at least) a single unique solution to the 3D reconstruction problem. One major difference between Baker et al.'s analysis and ours is that we are looking at the problem from a discrete point of view. Another is that the results in this paper show how ill-posed the 3D reconstruction problem is, and how difficult it is to reduce the solution space down to a reasonable number of solutions, let along a single unique one.

3. First Order Logic framework

This section formulates the problem of 3D reconstruction using a voxel framework and first order predicate logic.

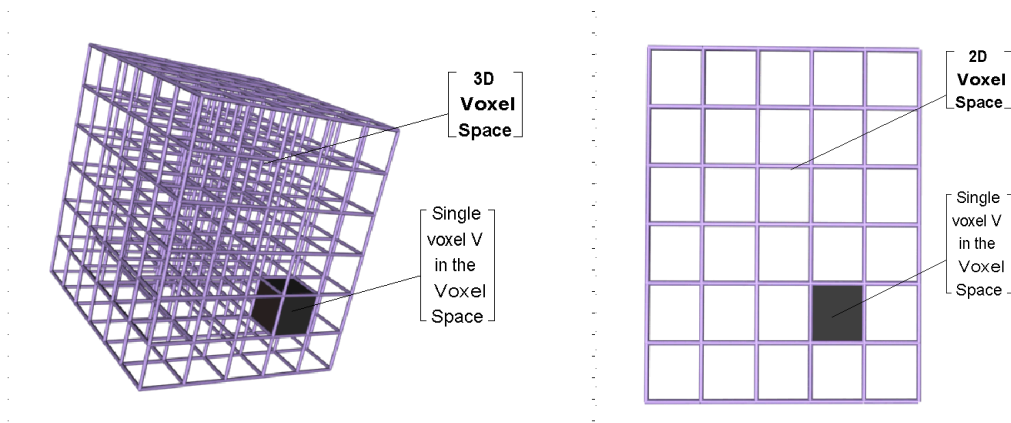Consider a single voxel V in the 3D voxel space S.

$$\exists V : V \in S$$

Fig. 1. 3D and 2D Voxel Space

This voxel projects onto m pixels (p) in n images (I) of the scene. These images are a subset of the total number of images in the scene. If I is defined as the total

6   *Martin Robinson, Kurt Kubik, Brian Lovell*

set of images, and $p\_project(i, V)$ returns the pixel that voxel V projects onto in image i, then the set of pixels and images that a voxel projects onto can be defined as:

$$project_V = \{p \mid \exists i : i \in I : p = p\_project(i, V)\}$$

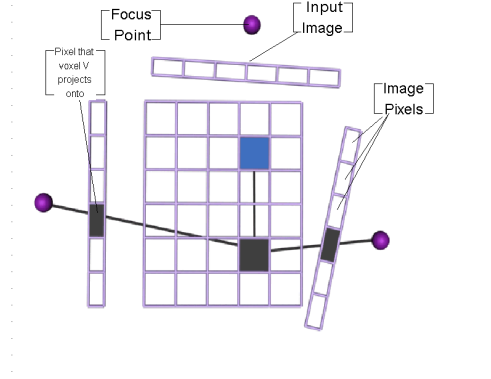$$projectImages_V = \{i \mid \exists p : p \in project_V : p \in i\}$$

Fig. 2. Each voxel projects onto m pixels in n images

Now if each of these pixels are projected out from the focal point of the corresponding image, each of these projected lines form that pixel's *light ray*. Note that there is a one to one relationship between each pixel and its ray, so from now on in this document, the symbol p will refer to both to a pixel and its ray.

The set of voxels that a pixel's ray passes through is given the symbol R_p

$$R_p = \{v \mid p \in project_v\}$$

A pixel's ray passes through three sets of voxels. There is the set of voxels before the voxel V

$$B_p^V = \{v \mid v \in R_p \wedge depth(v, p) < depth(V, p)\}$$

Where $depth(v, p)$ returns the number of voxels between v and the origin of p (the actual pixel). Then there is the set of voxels after the voxel V

$$A_p^V = \{v \mid v \in R_p \wedge depth(v, p) > depth(V, p)\}$$

Finally there is the set of the voxels equal in depth to voxel V. This is of course a set containing only V.

$$At_p^V = \{v \mid v \in R_p \wedge depth(v, p) = depth(V, p)\}$$
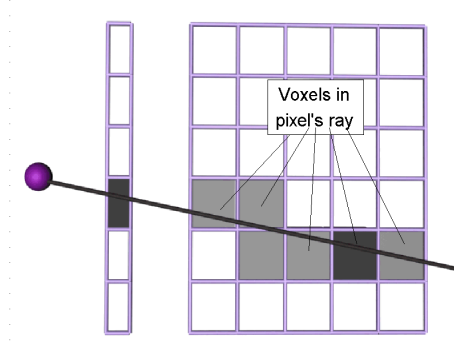$$At_p^V = \{V\}$$

Fig. 3. The set of voxels that a pixel's ray passes through

Of course, not every ray can travel on unimpeded forever. With respect to voxel V, each ray can have one of three properties:

(1) The ray can stop (i.e. project upon a filled voxel) **before** voxel V

$$before(V,p) \iff \exists v_1 : v_1 \in B_p^V : color(v_1) = color(p)$$
$$\wedge \left(\forall v_2 : v_2 \in B_p^{v_1} : color(v_2) = TRANSPARENT \right)$$

(2) The ray can stop **at** voxel V

$$at(V,p) \iff \exists v_1 : v_1 \in At_p^V : color(v_1) = color(p)$$
$$\wedge \left(\forall v_2 : v_2 \in B_p^{v_1} : color(v_2) = TRANSPARENT \right)$$
$$at(V,p) \iff color(V) = color(p) \wedge \left(\forall v : v \in B_p^V : color(v) = TRANSPARENT \right)$$
$$(see\ below\ for\ the\ definition\ of\ the\ set\ Transparent)$$

(3) The ray can stop **after** voxel V

$$after(V,p) \iff \exists v_1 : v_1 \in A_p^V : color(v_1) = color(p)$$
$$\wedge \left(\forall v_2 : v_2 \in B_p^{v_1} : color(v_2) = TRANSPARENT \right)$$

(4) The ray does not stop in the voxel area

$$background(p) \iff \forall v : v \in R_p : color(v) = TRANSPARENT$$

Each voxel in the 3D space S can be assigned to three possible sets determined by the properties of the rays passing through them.

(1) A voxel is judged to be a *transparent* voxel (i.e. not filled) if all the rays passing through it do not stop **at** and at least one stops **after** that particular voxel

$$Transparent(v) \iff \forall p : v \in R_p : (before(v,p) \vee after(v,p) \vee background(p))$$
$$\wedge (\exists p : v \in R_p : (after(v,p) \vee background(v,p)))$$
$$Transparent(v) \iff color(v) = TRANSPARENT$$

8   *Martin Robinson, Kurt Kubik, Brian Lovell*

(2) A voxel is judged to be a *surface* voxel (i.e. filled with a known colour) if all the rays passing through it do not stop **after** and at least one stops **at** that particular voxel

$$Surface(v) \Longleftrightarrow \forall p : v \in R_p : (before(v,p) \lor at(v,p)) \land (\exists p : v \in R_p : at(v,p))$$

$$Surface(v) \Longleftrightarrow color(v) \neq TRANSPARENT$$

(3) A voxel is judged to be an *inside* voxel (i.e. either enclosed inside an object or not seen by any images) if all the rays passing through it stop **before** that particular voxel

$$Inside(v) \Longleftrightarrow \forall p : v \in R_p : before(v,p)$$

Each voxel is one of these three types.

$$\forall v : v \in S : (Transparent(v) \lor Surface(v) \lor Inside(v))$$

4. A Simplified Voxel Space

In order to understand the problem better, it was decided that the equation system be solved for a simple version of the 3D reconstruction problem. Solving such a reduced problem should, in theory, be simple and it should give a greater understanding into the uniqueness, stability and solvability of the problem as a whole.

The voxel space is scaled down from three dimensions to two. This has the additional advantage that it is easy to present a visualisation of the voxel space. The focal point of each image is set at infinity, so all the pixel rays are perpendicular to the image planes.

5. Uniqueness

The ideal situation would be if we could obtain a unique solution to our problem of 3D reconstruction. But, unfortunately, the results obtained even for very small voxel space (e.g. 2x2, 1x1) are far from unique. See Figure 4 for a few sample results.

5.1. *Is it possible to get a unique solution?*

The answer to this questions is yes. It is possible to get a unique solution. But it is only possible to get a unique solutions for certain input images. Consider the input images in Figure 5. None of the input images have a common pixel colour, so the only possible solution is a totally transparent voxel space (i.e. a null space).

Unfortunately, this is not really useful to us. Is it only possible to get a unique solutions if that solution is totally transparent? It is easy to see that, given any number of input images and any size voxel space (except an infinite size), a possible solution will always be a totally transparent voxel space. From the problem formulation, a pixel's ray will pass all the way through the voxel space if all the voxels in the ray are transparent.

$$background(p) \Longleftrightarrow \forall v : v \in R_p : color(v) = TRANSPARENT$$

A First Order Logic Formulation of the 3D Reconstruction Problem   9

$$V\ IS\ \begin{vmatrix} T & T \\ T & T \end{vmatrix} OR \begin{vmatrix} T & T \\ 0 & T \end{vmatrix} OR \begin{vmatrix} T & T \\ 0 & 1 \end{vmatrix} OR$$
$$\begin{vmatrix} T & 1 \\ T & T \end{vmatrix} OR \begin{vmatrix} T & 1 \\ T & 0 \end{vmatrix} OR \begin{vmatrix} T & 1 \\ 0 & X \end{vmatrix}$$

$$V\ IS\ \begin{vmatrix} T & T \\ T & T \end{vmatrix} OR \begin{vmatrix} T & 1 \\ T & T \end{vmatrix} OR \begin{vmatrix} T & 1 \\ T & 3 \end{vmatrix}$$

$$V\ IS\ \begin{vmatrix} T & T & T \\ T & T & T \\ T & T & T \end{vmatrix} OR \begin{vmatrix} T & T & 1 \\ T & T & T \\ T & T & T \end{vmatrix} OR$$
$$\begin{vmatrix} T & T & 1 \\ T & T & T \\ T & T & 5 \end{vmatrix} OR \begin{vmatrix} T & T & 1 \\ T & T & 4 \\ T & T & T \end{vmatrix} OR$$
$$\begin{vmatrix} T & T & 1 \\ T & T & 4 \\ T & T & 5 \end{vmatrix}$$

Note on symbols used:
T stands for a *Transparent* voxel.
X stands for an *Inside* voxel (a voxel that is not seen by any camera).
Any number stands for a filled voxel with that colour.

Fig. 4. Some sample results
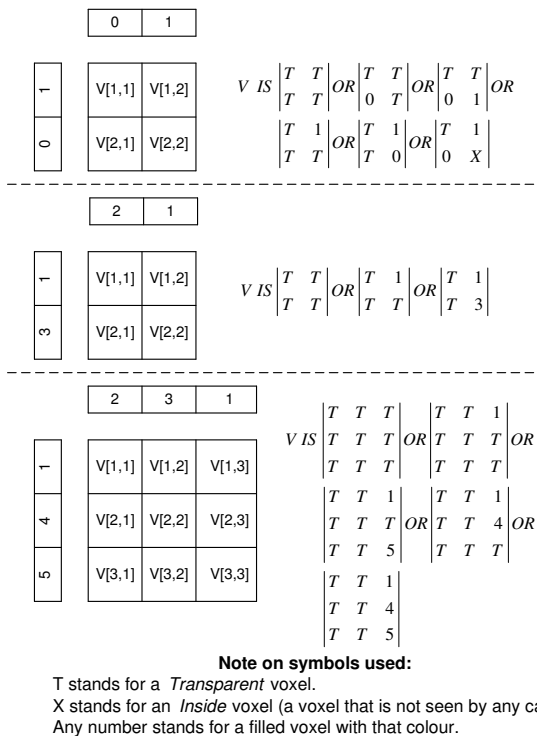

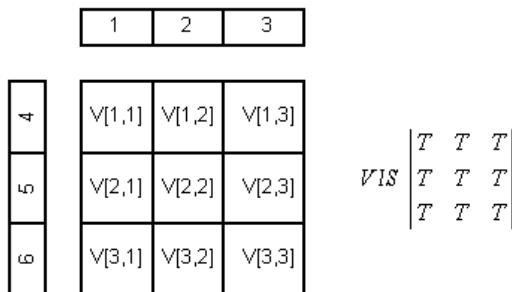$$VIS\ \begin{vmatrix} T & T & T \\ T & T & T \\ T & T & T \end{vmatrix}$$

Fig. 5. Example input images that give a unique solution

Because this equation does not depend on the colour of pixel p, it is always possible that *all* pixels are *background* pixels and the voxel space is totally transparent.

So, since the totally transparent solution is always a possible solution, it is impossible to get a unique solution unless that solution is the totally transparent solution. However, it is not practical to include a totally transparent voxel space

as a possible solution. Who would want to reconstruct an empty scene? So let us temporarily (for the rest of Section 5 only) add the following constraint to our formulation.

$$\forall pixel : background(pixel) \rightarrow false$$

This constraint ensures that all pixel rays stop inside the voxel space and that a totally transparent voxel space is never a possible solution.

With this new constraint in place, is it now possible to obtain a unique and useful solution? Well, it is easy to give an example set of input images that will result in a unique solution (see Figure 6), but is this really useful? In any practical problem, the only parameters of the 3D reconstruction problem the user has any control over are the number, positions and orientation of the cameras. So what is really needed is a configuration of input images that will always give a unique solution, no matter what the 3D scene or pixel colours are.
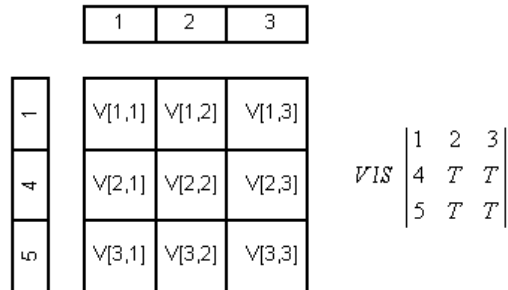

Fig. 6. Example input images that give a unique solution (with $\forall pixel : background(pixel) \rightarrow false$ constraint)

5.2. *Is there a image configuration that guarantees a unique solution?*

It is easy to see that there is no image configuration that guarantees a unique solution for any input image combination, even with the constraint of no background pixel rays introduced in Section 5.1. Consider the case where all the input images used in a reconstruction are black. No matter how many images you have, in whatever position and orientation, there will always (except for a voxel space with only one voxel in it) be more than one solution.

The reason why a totally black set of input images will never yield a unique solution is because of the lack of colour variation. Two pixel rays with different colours will always yield less solutions than two pixel rays with identical colours. So the more distinct colours in a set of input images, the smaller the solution space.

This immediately raises the question, is it possible to obtain a unique solution given sufficient colour variation in the input images. Since at this point there is no information as to how much colour variation is "sufficient", we can assume that there is maximum colour variation in the images. i.e. there is as many pixel colours as there is pixels.

5.3. *Given maximum colour variation, can a unique solution be guaranteed?*

Assume we have a very large, but finite, number of images. These images can be placed anywhere in the voxel space at any orientation. Now, assume that there is an equal number of 3D surfaces. Each surface has a corresponding image, and each surface is placed in the voxel space just in front of that image. The surfaces are one voxel thick, with the same width and breadth as their corresponding images. The voxel colours of these surfaces are exactly the same as the pixel values of their corresponding images. Figure 7 shows a graphical representation of this configuration. The filled-in voxels represent the surfaces in front of each camera.
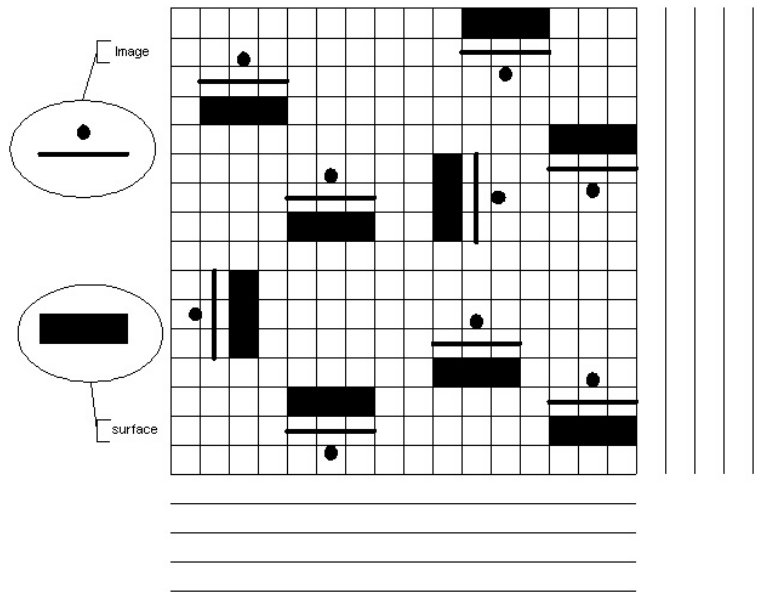

Fig. 7. Whatever the image pixel values, these surfaces will always be a possible solution

No matter what the pixel values of the input images were, these surfaces would be a valid and therefore possible solution. So, unless this is exactly the scene that you wish to reconstruct, it is impossible to obtain a unique solution, no matter how many images you have, their orientations or the amount of colour variation in the

scene.

This problem is exactly the flaw in the human vision system that can and is often used to provide us with Virtual Reality, or VR. In many VR environments, a user is given a headset that contains two small LCD displays that are placed directly in front of the users eyes when he/she puts the headset on. Once the displays are fed the correct images, the users sense of sight is fooled into believing that the user is in a virtual 3D environment.

6. Results of Solving Formulation for a Sample Input

This section shows the results of solving the formulation for a few different sets of input images.

6.1. *Input Image Configurations*

The testing was done using a total of eight images, spaced evenly around the voxel space, all pointing inward. The basic set up is shown in Figure 8.
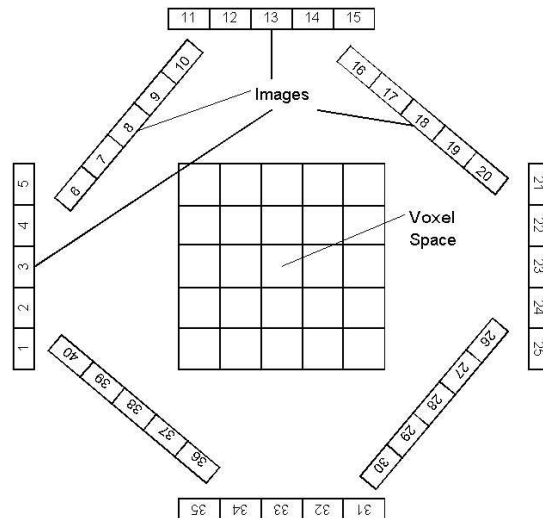

Fig. 8. Image Configuration (numbers are pixel numbers, not colours)

The numbers in the image pixels are pixel numbers, not colours. The pixels are numbered in the order in which they are processed. Naturally enough, the order in which they are processed affects the results. For example, if the first five pixels to be added are from five different images, you would expect the number of possible solutions using these five pixels to be less than if the first five were from the same image. However, after all the pixels have been processed, the final number of solutions will be the same, no matter what the processing order.

As can be seen from Figure 8, the images and voxel space used are very small. Only 2D spaces are used and the voxel space is only 5x5. This is done because of the processing times needed. Even for these small voxel space sizes some of the solution spaces are huge (over 10^7 solutions). For larger voxel spaces the growth in the number of solutions is exponential, because for each new voxel added, every existing solution will produce (on average) $\frac{n}{2}$ new solutions, where n is the number of possible states for that voxel. The number of possible states is the number of different colored pixel rays passing through the voxel plus one, since the voxel can also be transparent.

Only rendered images are used. This way there are no photometric distortions or lens-induced aberrations of real cameras that yield corresponding points with quite different colours. For convenience, the images are rendered with infinite focal lengths, so that they record a parallel projection of the scene. Finally, any pixel that does not project upon a voxel in the scene is removed from consideration. This is why, in Figure 11, the plots for *Shape* only go up to $x = 35$, not 40.

6.2. *Voxel Spaces*

The voxel space is a 5x5x1 array of voxels. The following voxel configurations were tested.

6.2.1. Rect *(Rectangle)*

This voxel configuration is a simple rectangle the exact same size of the voxel space. Different *Rect* configurations have different texture or colour information. *Rect1* gives all the surface voxels a unique colour. *Rect2* has each colour occurring in blocks of two voxels, and so has half the number of different colours as *Rect1*. *Rect16* gives all the surface voxels the exact same colour. *Rect Bland* is similar to *Rect1* except there is a large area with a constant colour. *Rect Repeat* is similar to *Rect Bland* except that the large area is covered in a repeating texture. See Figure 9 for the different voxel configurations under this class.

6.2.2. Shape *(Strange Looking Shape)*

The different configurations of *Shape* are similar to *Rect*, except that the surface voxels form a strange looking shape. Note that there is no *Shape16* corresponding to *Rect16* since any images of these two configurations would be exactly the same.

6.3. *Results*

These experiments aim to calculate the solution space using the logical formulation and the image's pixel values and position/orientation information. No constraints are added, so the solution space is comprised of all the possible solutions that satisfy the logic equations set out in Section 3.
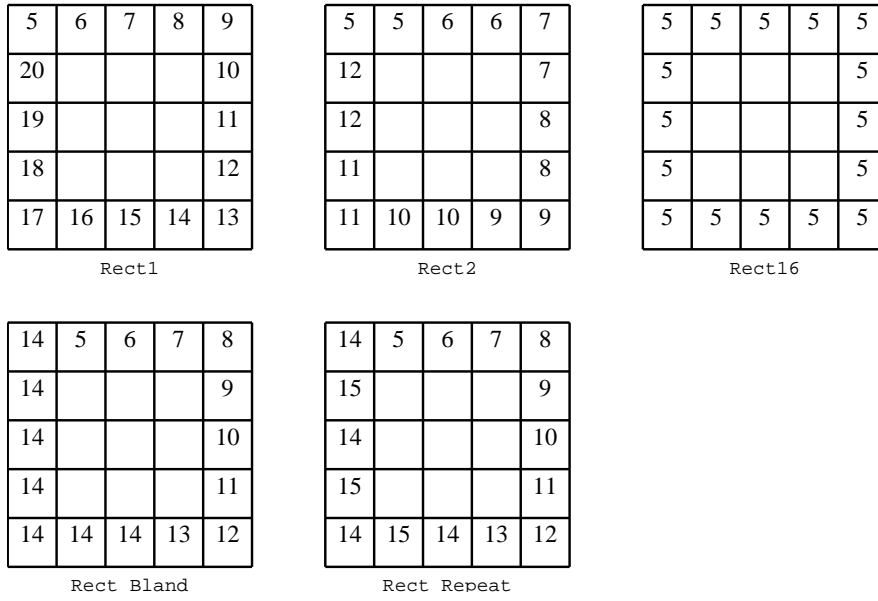
14   *Martin Robinson, Kurt Kubik, Brian Lovell*

5	6	7	8	9
20				10
19				11
18				12
17	16	15	14	13

Rect1

5	5	6	6	7
12				7
12				8
11				8
11	10	10	9	9

Rect2

5	5	5	5	5
5				5
5				5
5				5
5	5	5	5	5

Rect16

14	5	6	7	8
14				9
14				10
14				11
14	14	14	13	12

Rect Bland

14	5	6	7	8
15				9
14				10
15				11
14	15	14	13	12

Rect Repeat

Fig. 9. *Rect* Voxel Configurations

		5		
		6	7	
17	18	19	8	9
16				10
15	14	13	12	11

Shape1

		5		
		5	6	
11	11	12	6	7
10				7
10	9	9	8	8

Shape2

		5		
		13	6	
13	13	13	7	8
13				9
13	13	12	11	10

Shape Bland

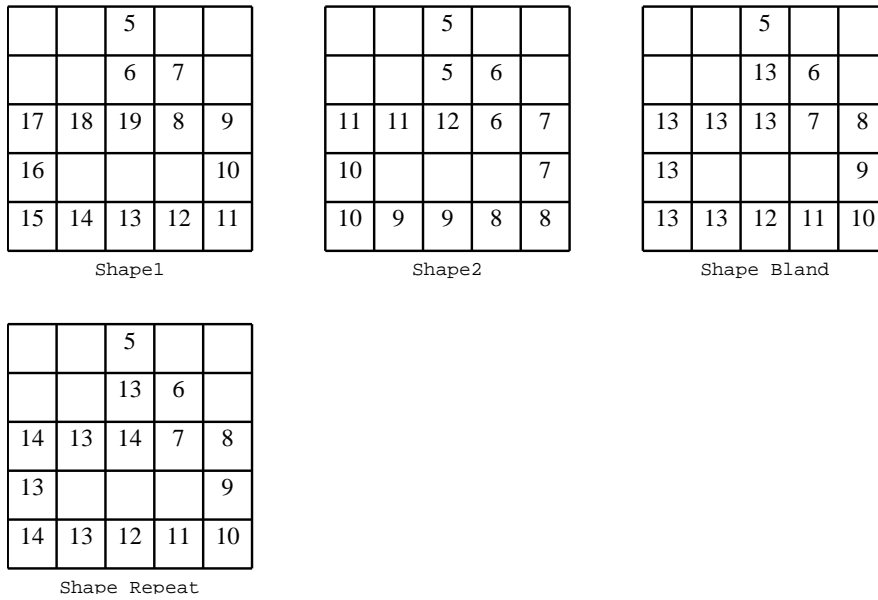|		5		
		13	6	
14	13	14	7	8
13				9
14	13	12	11	10

Shape Repeat

Fig. 10. *Shape* Voxel Configurations

The results are set out in plots of *Number of Possible Solutions* versus *Number of Pixels*. So if *Number of Pixels* = 5 and *Number of Possible Solutions* = 2000, this means that, after the program had taken into account pixels 1-5, there were 2000 solutions that projected correctly onto these pixels and satisfied the logical formulation.

The first thing to point out about the results is that the number of possible solutions starts at one. This may seem somewhat counter-intuitive. Surly when there is no pixel information available, the solution space will be at its maximum? However, our formulation states that any pixel not seen by a pixel is an *inside* pixel, and if there is no pixel information available, then the only possible solution is that where all the voxels are inside voxels. This fits in well with the goal of 3D reconstruction. When reconstructing the shape of an object, the colour of the voxels not seen by the camera are inconsequential, and therefore these voxels should not add to the size of the solution space.

Looking at the results given in Figure 11, it can be seen how each new pixel affects the solution space. After the first image is processed (the first image contains little meaningful 3D information on its own), subsequent pixels added tend to further constrain the solution space. This trend breaks down when there is little or no colour variation in the input images. The plot in Figure 12 shows what happens in this situation. All the surface voxels in *Rect16* have the same colour, so instead of the solution space being mostly constrained with the addition of new pixels, the size of the solution space grows exponentially with each new pixel.

The results for the differing voxel configurations show the effect of a few differing sources of ambiguity (Figure 11). Bland regions are generally considered as the biggest obstacle in reconstructing a scene and this is reflected in the fact that both the *Rect Bland* and *Shape Bland* voxel configurations gave the largest solution spaces. *The Rect Repeat* and *Shape Repeat* plots clearly show the ambiguity caused by repeating texture. Even though these voxel configurations contain more unique colours than *Rect2* and *Shape2* respectively, they produce larger solution spaces. Finally, it is worth noting that the plots produced by *Rect1* and *Shape1* show the smallest solution spaces that you can expect from these shapes using the current image configuration, since every surface voxel in these two voxel configurations has its own unique colour.

These results show how large the solution space can be even for very small voxel spaces. Obviously, solving the 3D reconstruction problem using no constraints will almost never give a solution space small enough so that the scene geometry can be extracted with any degree of accuracy.

7. Conclusions

Without any knowledge of the 3D space with which to apply constraints, it is impossible to guarantee a unique solution to the 3D reconstruction problem, no matter how many images are taken of the scene, they're orientation or even how

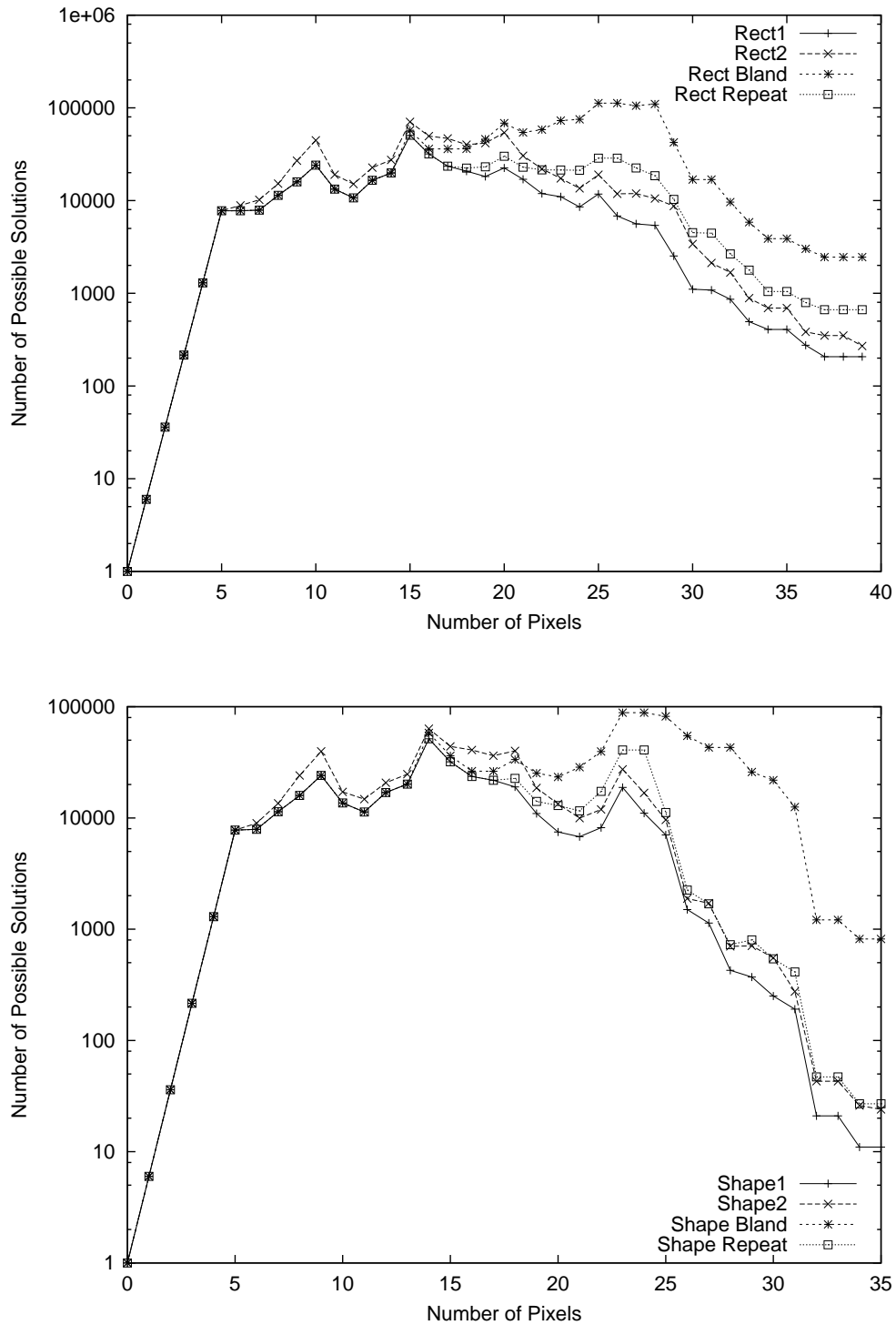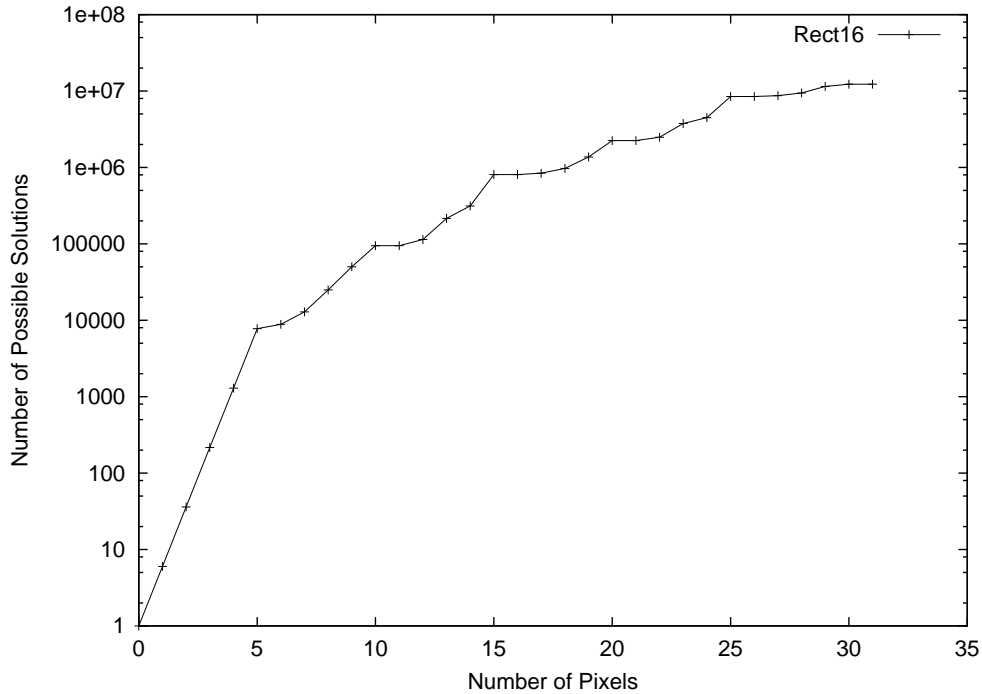16   *Martin Robinson, Kurt Kubik, Brian Lovell*

Fig. 11. Results of solving the logical formulation for *Rect* and *Shape*

Fig. 12. Results of solving the logical formulation for *Rect16*

much colour variation is in the scene itself.

The formulation presented in this paper allows the calculation of the entire solution space, without any constraints to artificially limit this space. The results from reconstructing even small voxel spaces (5x5 voxels) show that the solution space is huge, with the number of solutions ranging from 10 to 10^7. These results also clearly show the ambiguities caused by both bland regions and repeated textures.

Given these results, it is easy to see why constraints are needed in order to reduce the solution space to a more reasonable size. The next step is to implement additional constraints on top of the base formulation. Because of the general nature of the formulation, this can be easily done, since any constraint simply removes those solutions that do not conform from the solution space.

The formulation can also be used to provide a quantitative measure of the effectiveness of any constraint. Because of the formulation's discrete nature, the exact size of the solution space can be easily found by counting the number of possible solutions. Since the purpose of a constraint is to reduce the solution space as much as possible, this size could be used as a quantitative measure of the effectiveness of any constraint. This can be further investigated when additional constraints are implemented and tested.

18   *Martin Robinson, Kurt Kubik, Brian Lovell*

References

1. Simon Baker, Terence Sim, and Takeo Kanade. When is the shape of a scene unique given its light-field: a fundamental theorem of 3d vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1), January 2003.
2. A.F. Bobick and S.S. Intille. Large occlusion stereo. *IJCV*, 33(3):1–20, September 1999.
3. J. De Bonet and P. Viola. Roxels: Responsibility weighted 3D volume reconstruction. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume I, pages 418–425, Los Alamitos, CA, September 20–27 1999. IEEE.
4. W. Bruce Culbertson, Thomas Malzbender, and Gregory G. Slabaugh. Generalized voxel coloring. In *Workshop on Vision Algorithms*, pages 161–169, 1999.
5. C. R. Dyer. Volumetric scene reconstruction from multiple views. In L. S. Davis, editor, *Foundations of Image Analysis.* Kluwer, 2001.
6. Oliver Grau and Ralf Tnjes. Knowledge Based Modelling of Natural Scenes. In *European Workshop on Conbined Real and Synthetic Image Processing for Broadcast and Video Production*, November 1994.
7. J. Hsu, Z. Pizlo, C. Babbs, D. Chelberg, and E. Delp. Design of studies to test the effectiveness of stereo imaging, truth or dare: is stereo viewing really better. In *Proc. SPIE 1994, vol. 2177*, pages 211–220, 1994.
8. S.S. Intille and A.F. Bobick. Disparity-space images and large occlusion stereo. In *ECCV94*, pages 179–186, 1994.
9. S.S. Intille and A.F. Bobick. Incorporating intensity edges in the recovery of occlusion regi ons. In *ICPR94*, pages A:674–677, 1994.
10. Kiriakos Kutulakos and Steven Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.
11. Claus-E. Liedtke, O. Grau, and Stefan Growe. Use of explicit knowledge for the reconstruction of 3-d object geometry. In *Computer Analysis of Images and Patterns*, pages 580–587, 1995.
12. D. Marr and T.A. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, October 15, 1976, October 1976.
13. Wojciech Matusik, Chis Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. *SIGGRAPH*, pages 369–374, 2000.
14. H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99)*, pages 49–54, Los Alamitos, June 23–25 1999. IEEE.
15. S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR97*, pages 1067–1073, 1997.
16. Greg Slabaugh, Bruce Culbertson, Tom Malzbender, and Ron Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *VG2001*, 2001.
17. Gregory G. Slabaugh, Tom Malzbender, and Bruce Culbertson. Volumetric warping for voxel coloring on an infinite domain. In *SMILE2000*, pages 109–123, 2000.
18. Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV (2)*, pages 151–158, 1994.