2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Discovering and Mapping Complete Surfaces With Stereo

Robbie Shade and Paul Newman

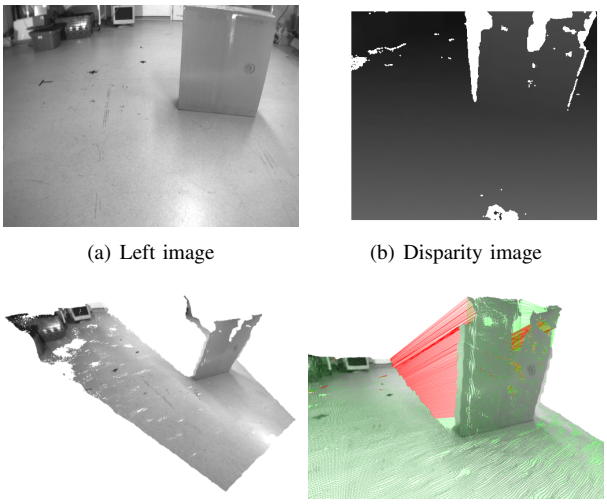
Abstract— **This paper is about the automated discovery and mapping of surfaces using a stereo pair. We begin with the observation that for any workspace which is topologically connected (*i.e.* does not contain free flying islands) there exists a single surface that covers the entirety of the workspace. We call this surface the covering surface. We assume that while this surface is complex and self intersecting every point on it can be imaged from a suitable camera pose and furthermore that it is locally smooth at some finite scale – it is a manifold. We show how by representing the covering surface as a non-planar graph of observed pixels we are able to plan new views and importantly fuse disparity maps from multiple views. The resulting graph can be lifted to 3D to yield a full scene reconstruction.**

I. Introduction

In this paper we consider the problem of workspace surface discovery using an actuated stereo camera. Our goal is to generate a sequence of control commands which will servo the camera to take views of the workspace which will guarantee that we image every point on every visible surface of every object in the workspace. With this complete coverage we can construct a single covering surface $\mathcal{S}$ of the entire workspace. This work is motivated by a desire to construct complete and detailed maps of unknown workspaces — a task which has an important role to play in automated inspection. We emphasise that the aim is to obtain surface coverage of a arbitrarily complex workspace using a stereo camera - not the exploration of free-space.

We make few assumptions in this work but they need to be made explicit. Firstly we require that we have accurate information on the pose of the camera for all time – a reasonable assumption given we are using a stereo pair and the state of the art in structure from motion and SLAM techniques[1]. Although the surface $\mathcal{S}$ will be complex and self intersecting we place just two restrictions on the workspace itself. Firstly we require that it must be topologically connected (*i.e.* devoid of floating objects). Secondly, for practical reasons we require that there exists at least one camera pose for every point $\mathbf{p} \in \mathcal{S}$ such that the $\mathbf{p}$ is four connected when projected into a stereo disparity map. Put simply this implies that there are no prominent knife edges which resolutely remain one dimensional however close the camera gets to them.

Our approach can be summarised as follows. We construct a non-planar graph consisting of observed pixels as vertices. Neighbouring pixels in a stereo disparity image will be connected with an edge weighted in proportion to the Euclidean distance in $\mathbb{R}^3$ between vertices - spatial positions having been triangulated from stereo. As new views are taken the graph is grown such that at any time the edge between two vertices will correspond to the smallest observed Euclidean distance between corresponding pixels in any disparity image to date. By detecting and ranking the severity of one dimensional rifts in this graph (collections of edges with large weights in close proximity) we can plan a continuous trajectory through space to guide the camera to a new pose, the view from which will enable some if not all of the scene detail hidden by the rift to be filled in. The process can continue until no significant rifts remain. In what follows we will carefully explain the detail behind this scheme and cover some of the intricacies required to maintain a consistent graphical structure.



(a) Left image (b) Disparity image

(c) Point cloud (d) Graph detail

Fig. 1. The results of constructing our graph structure from a typical stereo image pair and disparity image. (a) is the input image $\mathcal{I}^L$ (the corresponding $\mathcal{I}^R$ is not shown). (b) shows the result of applying stereo processing to $\mathcal{I}^L$ and $\mathcal{I}^R$, the disparity image $\mathcal{D}$. Darker pixels are further from the camera. (c) the point cloud which is obtained by triangulation of valid disparity values. (d) detail of the graph structure - edges have been coloured according to weight: low weight edges are green, while high weight edges are red.

II. Related Work

There is a wide ranging and extensive literature on 3D mapping and reconstruction with mobile robots. A typical approach is to use a 2D occupancy grid as the map of the world [2] which necessarily requires discretisation of the world into quantised cells. This in turn risks oversimplification of structure of the world, and does not allow for overhanging obstacles such as bridges. Later work such as the multi-level surface maps of [3] extend this idea, allowing for multiple traversable levels in each cell of the grid.

The authors are with the Oxford University Mobile Robotics Group {rjs,pnewman}@robots.ox.ac.uk

A common exploration strategy on such maps is frontier based exploration [2] [4] - the robot is directed towards the closest frontier between explored and unexplored grid cells. More recently [4] uses a maximum-likelihood occupancy grid generated from stereo data, and frontier based exploration strategies to explore an office environment.

The gap navigation tree (GNT), introduced by [5] and [6], is a data structure designed to maintain a graph of gaps - depth discontinuties with respect to the heading of the robot. The authors show that by chasing down gaps, while maintaining the graph structure by adding and removing gaps as necessary, locally optimal navigation can be achieved. [7] introduces a path planning exploration algorithm based on eliminating these gaps using visibility maps. Exploration of a simple bounded environment by multiple observers is demonstrated.

On a larger scale, the visibility-based fusion of stereo data by [8] quickly combines stereo depth maps from multiple viewponts. Using consideration of occlusions and free-space violations to find accurate depths for each pixel, they then apply a triangular mesh to the resulting point cloud to produce the final model in real-time. Similarily [9] overcomes the 2D limitations of occupancy grids by using dynamic 3D octrees and planar polygon fitting the resulting data. Semantic labelling of terrain, with different path planners for different environments, leads to a real-time system for navigation.

Our work has interesting parallels with the Gap Navigation Tree of LaValle in which it is shown that a scheme that requires the chasing down of discontinuities perceived depth in a 2D environment provably enforces complete exploration. One could see the graphical approach we adopt here as a refactorisation of this approach and a generalisation to 3D.

III. Notation

Notation conventions followed in this paper are as follows. A 6 DoF camera pose at time i is

$$\mathbf{c}_i = [x, y, z, \theta_r, \theta_p, \theta_q]^T$$

We will refer to points in $\mathbb{R}^3$ as

$$\mathbf{p} = [p_x, p_y, p_z]^T$$

Left and right images from the stereo camera are $\mathcal{I}^L$ and $\mathcal{I}^R$. A pixel at image row and column $[r, c]^T$ is addressed as $\mathcal{I}^L_{r,c}$. An image seen at camera pose $\mathbf{c}_i$ is ${}_i\mathcal{I}^L$.

IV. Graph Construction

A. Disparity from stereo

Stereo was chosen as the sensor modality for this research as it provides dense point clouds and high resolution source images at high speed. A stereo camera provides left and right images $\mathcal{I}^L$ and $\mathcal{I}^R$. Corresponding pixels are identified using a local window-based matching technique resulting in a *disparity image* $\mathcal{D}$.

Every pixel of a disparity image stores a floating point value $d = \mathcal{D}_{r,c}$, the *disparity* of $\mathcal{I}^L_{r,c}$. This disparity is the horizontal displacement between pixels $\mathcal{I}^L_{r,c}$ and $\mathcal{I}^R_{r,c-d}$ which are images of the same point in $\mathbb{R}^3$.

Some pixels do not have a disparity score due to poor matching confidence or occlusions and are assigned an error value. We will use the notation $\mathcal{D}_{valid}$ to refer to the subset of pixels in $\mathcal{D}$ which have valid disparities.

Figs. 1(a) and 1(b) show an example $\mathcal{I}^L$ and the resulting $\mathcal{D}$. Pixels $\notin \mathcal{D}_{valid}$ are coloured white. The implementation details of the stereo algorithm are described in Section IX.

B. Point cloud from disparity

With knowledge of the camera parameters (importantly focal length f and baseline b) we can back project a pixel in $\mathcal{D}$ and triangulate the position in $\mathbb{R}^3$. The back projection function at camera pose $\mathbf{c}_i$ is:

$$\{\pi_i(r, c) : \mathbb{R}^2 \rightarrow \mathbb{R}^3\} = \left[\frac{b.c}{{}_i\mathcal{D}_{r,c}}, \frac{b.r}{{}_i\mathcal{D}_{r,c}}, \frac{f.b}{{}_i\mathcal{D}_{r,c}}\right]^T \quad (1)$$

Applying this function to all pixels in ${}_i\mathcal{D}_{valid}$ gives us a point cloud:

$$\mathbf{P} = [\pi_i(r, c) : \forall [r, c] \in {}_i\mathcal{D}_{valid}] \quad (2)$$

This resulting point cloud is unorganised - it is simply a list of points in space (Fig. 1(c) shows an example). However, the source images which gave rise to this point cloud encode a great deal of information about the relationship between pixels and we aim to include this in the construction of our graph structure.

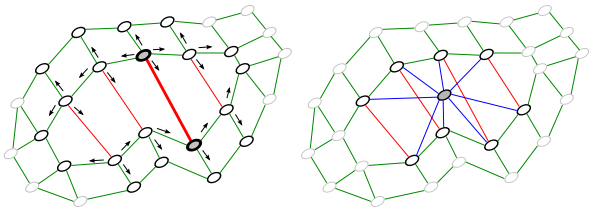
Two neighbouring pixels in an image with similar disparities suggest that the two pixels are images of neighbouring points on a surface in the world. Areas of a disparity image with smooth gradients indicate smooth surfaces in the world while discontinuties - edges in the disparity image - highlight physical discontinuties in space.

An undirected graph, $\mathcal{G} = [\mathbf{V}, \mathbf{E}]$, consists of a set of vertices, $\mathbf{V}$, and a set of edges, $\mathbf{E}$, which describe the connectivity between vertices. In the case of a single disparity map from camera pose $\mathbf{c}_i$, every pixel $\mathbf{p} \in \mathbf{P}$ results in a graph vertex v.

A vertex v, has a 3D position $v_\mathbf{x}$, a surface normal $v_\mathbf{n}$, and a colour $v_{r,g,b}$. Edges are constructed by considering the pixels which surround v in ${}_i\mathcal{D}$. If v was seen at ${}_i\mathcal{D}_{r,c}$, then we consider the neighbouring pixels ${}_i\mathcal{D}_{r\pm1,c\pm1}$. If one or more of these pixels has a valid disparity and hence a corresponding vertex u, then an edge is created connecting v and u. Every edge e has a weight e_w associated with it - in this case it is simply the Euclidean distance in $\mathbb{R}^3$ between the two end vertices:

$$e_w = ||v_\mathbf{x} - u_\mathbf{x}||_2 \quad (3)$$

Fig. 1 shows the stages of constructing such a graph from a typical stereo image pair. 200ppiImages $\mathcal{I}^L$ (Fig. 1(a)) and $\mathcal{I}^R$ are used to create $\mathcal{D}$ (Fig. 1(b)). $\mathcal{D}$ is converted to a point



(a) Input graph (b) Resulting rift

Fig. 2. Rift generation. Starting with a high weight edge (bold red edge in (a)), we recursively follow high weight edges connected to the neighbours of its end vertices. Explored edges are indicated with arrows. The detected rift is shown in (b).

cloud ((Fig. 1(c)), and is used to create the graph structure (Fig. 1(d)). The graph edges have been coloured according to weight - low weight edges are green, while high weight edges are red.

V. Edge Rifts

We now tackle the problem of identifying areas of the scene which should be marked for exploration. Apparent in Fig. 1 is the fact that groups of edges with high weights and in close spatial proximity are indicative of discontinuities on the surface explored so far. We call these groups of edges *rifts*, and our method for identification of rifts in a graph is as follows.

Algorithm 1: Rift identification in a graph

Input: Seed edge e_s
Edge weight threshold δ
Output: Rift R

$ToVisit = [e_s]$
while $ToVisit \neq \emptyset$ **do**
  $e = pop(ToVisit)$
  foreach $n \in Neighbours([e_u, e_v])$ **do**
    foreach $d \in Edges(n)$ **do**
      if $d_w > \delta$ **then**
        $ToVisit = ToVisit \cup d$
      end
    end
  end
end

Starting with a high weight edge, e, we recursively follow high weight edges connected to the neighbours of its end vertices, as described in Algorithm (1). Every high weight edge encountered is added to the rift R, along with its end vertices. Finally a rift vertex is created in $\mathcal{G}$ at the geometric center of the vertices comprising R.

The process is shown graphically in Fig. 2 on a simple graph containing one rift. The emboldened graph edge in Fig. 2(a) is the edge with the highest weight and is the starting point for rift generation. Explored edges are indicated by arrows, with the detected rift shown in Fig. 2(b). The rift vertex can be seen in the geometric center, with blue edges connecting it to $\mathcal{G}$. We apply Algorithm (1) to every edge in $\mathcal{G}$ which has a sufficiently large weight resulting in a set of rifts $\mathbf{R} = [R_0, R_1, ..., R_n]$

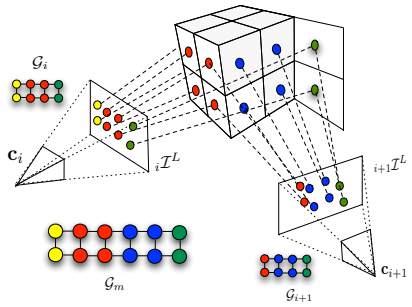


Fig. 3. Merging of two graph structures. Each vertex v, in $\mathcal{G}_{i+1}$ is projected into ${}_i\mathcal{I}^L$. If the pixel it projects onto is valid ($[r,c]^T \in \mathcal{D}_{valid}$) then the outgoing edges of v are added to the vertex in $\mathcal{G}_i$ corresponding to ${}_i\mathcal{D}_{r,c}$, and v is discarded. Otherwise v is added to $\mathcal{G}_i$ as a new vertex.

VI. Merging Graphs

Given two graphs, $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$, along with associated camera poses $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$, we wish to merge these into a single graph $\mathcal{G}_m$. Any given rift in $\mathcal{G}_i$ may not be visible from $\mathcal{G}_{i+1}$. To ensure complete surface coverage all existing rifts must be inspected at some point, and so we do not want to discard the old graph structure once after a change in pose. The merging process aims to combine vertices from $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ which are representations of the same point in $\mathbb{R}^3$.

Algorithm 2: Graph merge operation

Input: $\mathcal{G}_i = [\mathbf{V}_i, \mathbf{E}_i]$
$\mathcal{G}_{i+1} = [\mathbf{V}_{i+1}, \mathbf{E}_{i+1}]$
Minimum distance threshold γ
Output: Merged graph: $\mathcal{G}_m = [\mathbf{V}_m, \mathbf{E}_m]$

$\mathcal{G}_m = \mathcal{G}_i$
foreach $v \in \mathbf{V}_{i+1}$ **do**
  $[r,c]^T = \pi^{-1}(v_\mathbf{x})$
  $ValidPixel = [r,c]^T \in {}_i\mathcal{D}_{valid}$
  $Distance = ||\pi_{i+1}(r,c) - v_\mathbf{x}||_2$
  if *ValidPixel && Distance* $< \gamma$ **then**
    foreach $[u,v] \in Edges(v)$ **do**
      $\mathbf{E}_m = \mathbf{E}_m \cup Edge(u, {}_i\mathcal{D}_{r,c})$
    end
  end
  else
    $\mathbf{V}_m = \mathbf{V}_m \cup v$
  end
end

First we define a projection function from a point $\mathbf{p}$ to image coordinates $[r, c]^T$:

$$\{\pi^{-1}(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}^2\} = \left[\frac{f.p_x}{p_z}, \frac{f.p_y}{p_z}\right]^T \quad (4)$$

Each vertex $v \in \mathcal{G}_{i+1}$ is projected into the disparity image ${}_i\mathcal{D}$. If v is reprojected onto a valid pixel, (*i.e.* $[r, c]^T \in {}_i\mathcal{D}_{valid}$) then we calculate the distance in between v and $\pi_{i+1}(r, c)$. If this is below some threshold, typically of the order of $1cm$, we add the outgoing edges of v in $\mathcal{G}_{i+1}$ to the vertex corresponding to ${}_i\mathcal{D}_{r,c}$ in $\mathcal{G}_i$, and discard v. Otherwise v is added to $\mathcal{G}$.

The pseudocode for this operation is given in Alg. 1. The process is shown graphically in Fig. 3 for a simple synthetic scene.

VII. Edge Visibility

The problem which now arises is that of determining when an edge rift has been resolved, *i.e.* the unknown area of the workspace represented by the rift has been explored. Consider an edge e seen from pose $\mathbf{c}_i$ and a new pose $\mathbf{c}_{i+1}$. This is shown in Fig. 4.

If there exists a vertex in $\mathcal{G}_{i+1}$ which is collinear with a point on e and the camera center $\mathbf{c}_{i+1}$ then we perform a visibility test. The projection of edge point $\mathbf{p}_e$ in ${}_{i+1}\mathcal{I}^L$ is

$$[r, c]^T = \pi^{-1}_{i+1}(\mathbf{p}_e) \quad (5)$$

If $[r, c]^T \in \mathcal{D}_{valid}$, we can retrieve the $\mathbb{R}^3$ position of the corresponding vertex v:

$$\mathbf{p}_{r,c} = v_{\mathbf{x}} \quad (6)$$

The two points $\mathbf{p}_e$ and $\mathbf{p}_{r,c}$ are collinear with the camera center $\mathbf{c}_{i+1}$ as they both project onto the same pixel. The visibility test is a comparison of the distances from the camera center to both points:

$$d_{\mathbf{p}_e} = ||\mathbf{p}_e - \mathbf{c}_{i+1}||_2 \quad (7)$$

$$d_{\mathbf{p}_{r,c}} = ||\mathbf{p}_{r,c} - \mathbf{c}_{i+1}||_2 \quad (8)$$

If $d_{\mathbf{p}_e} < d_{\mathbf{p}_{r,c}}$ then point $\mathbf{p}_{r,c}$ is visible behind $\mathbf{p}_e$ and passes the test. The end points of e are projected into ${}_{i+1}\mathcal{D}$ and the set of pixels to test is chosen by the Bresenham line drawing algorithm. Edge e is deleted from $\mathcal{G}$ if the fraction of points passing the visibility test is greater than a threshold, typically 75% of the pixels.

Fig. 4 shows a graphical example. The red edges e and e' were created at pose $\mathbf{c}_i$. The camera has now moved to $\mathbf{c}_{i+1}$ and we use the visibility test on e and e' to determine whether they survive. Edge e will be deleted as the camera can now see pixels $\mathbf{p}_{r,c}$ behind every point on e. Edge e' will survive the visibility test because all points $\mathbf{p'}_{e'}$ are occluded by pixels $\mathbf{p'}_{r,c}$.

VIII. Planning Next View

We want to plan a continous camera trajectory through space from the current pose $\mathbf{c}_i$ to a new pose $\mathbf{c}_{i+1}$. The view

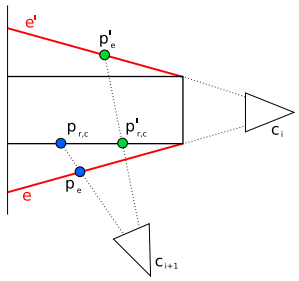


Fig. 4. Visibility testing. Two edges from e and e' seen from $\mathbf{c}_i$. From the new pose $\mathbf{c}_{i+1}$ pixel $\mathbf{p}_{r,c}$ is visible behind $\mathbf{p}_e$. Pixel $\mathbf{p'}_{r,c}$ is blocking the view of $\mathbf{p'}_e$.

from $\mathbf{c}_{i+1}$ should enable the visibility based deletion of a rift, and thus increase coverage of the environment surface.

A prerequisite is a method for surface normal estimation - the camera poses comprising the path will be oriented to look at the surface as the camera moves, and more importantly a rift vertex must have a viewing direction associated with it.

A. Surface Normal Estimation

For a given vertex, v, we calculate the local surface normal by considering its immediate neighbours in the graph and fitting a plane to this point set. Inherent in this method is a tradeoff between data smoothing and sensitivity to noise - a large neighbourhood leads to noise insensitivity but poor resolution. From experience we have found that using vertices within one graph edge of v gives acceptable results.

A typical vertex will have four immediate neighbours, but after graph merging (Section VI) may have more. Including the source vertex, we get a set of k points:

$$\mathbf{Q} = \begin{bmatrix} v_0 & \dots & v_k \end{bmatrix} = \begin{bmatrix} v_{0x} & \dots & v_{kx} \\ v_{0y} & \dots & v_{ky} \\ v_{0z} & \dots & v_{kz} \end{bmatrix} \quad (9)$$

A number of surface normal estimation methods for point clouds are described and evaluated in [10]. They conclude that for reliability, quality, and speed, a generally good choice is the Singular Value Decomposition (SVD)[11].

The plane solution is obtained by mean-centering the matrix $\mathbf{Q}$ and solving

$$\min_{\mathbf{n}} \left|\left|\mathbf{Q}^+\mathbf{n}\right|\right|_2 \quad (10)$$

where $\mathbf{n}$ is the plane normal $[a, b, c]^T$ and $\mathbf{Q}^+$ is the mean-centered matrix:

$$\mathbf{Q}^+ = \left[\mathbf{Q} - \frac{1}{k}\sum_{j=1}^{k}\mathbf{Q}_j\right]^T \quad (11)$$

This is solved by taking the SVD ($\mathbf{U}\Sigma\mathbf{V}^T$) of $\mathbf{Q}^+$. The vector in $\mathbf{V}$ corresponding to the smallest singular value in Σ is the normal of the plane.

B. Rift selection

When faced with a set of rifts $\mathbf{R}$, we order them according to severity and distance from $\mathbf{c}_i$ to the rift center:

$$cost(R_i) = \alpha\Delta + \beta\mathcal{S} \tag{12}$$

Δ is the distance through $\mathcal{G}$ from $\mathbf{c}_i$ to R_i. Severity, $\mathcal{S}$, is a measure of rift size and is taken to be the number of edges in the rift. α and β are weightings which are chosen depending on the application and the graph scale. The rift with the highest cost is chosen as the *dominant* rift, R_d. Pose $\mathbf{c}_{i+1}$ is oriented to look along the normal of R_d.

The camera trajectory is found by planning a path over the graph from $\mathbf{c}_i$ to a $\mathbf{c}_{i+1}$. The A^*graph search algorithm [12] [13] is used to find this path, and returns an ordered set of graph vertices, $\mathbf{v} = [v_s ... v_g]$, and correspondinge edges, which constitute a path from a source vertex v_s to a goal vertex v_g. Using the admissible heuristic of Euclidean distance in $\mathbb{R}^3$ between vertices, $A^*$is guaranteed to return the shortest path.

Each vertex on this path has a normal $v_\mathbf{n}$, and so the final camera trajectory consists of a ordered set of poses $\mathbf{c}_{path} = [\mathbf{c}_s, ..., \mathbf{c}_g]$ each oriented such that they view the workspace surface along the normal at a fixed height (typically $1m$). The normal of a rift is taken to be the mean of the normals of the vertices comprising the rift.

IX. Results

A. Experimental setup

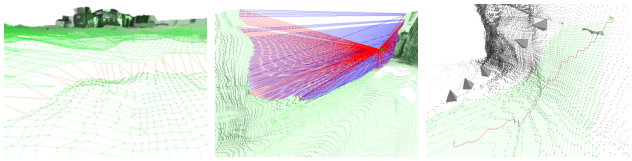
A handheld PointGrey Bumblebee stereo camera was used to capture mono images $\mathcal{I}^L$ and $\mathcal{I}^R$ at a resolution of 512x384 pixels. The camera poses were accurately estimated with a robust visual odometry system. This is described in [1], and is shown to have positional errors of the order of $1cm$ over a $1km$ loop.

B. Stereo

1) Preprocessing: The images are undistorted and rectified ensuring that the epipolar lines correspond to image rows - this reduces the stereo matching problem to a 1D search. They are processed with a fast approximation to the Laplacian of Gaussian filter - the separable Difference of Gaussians. This reduces image noise, removes photometric variation between images, and enhances edges.

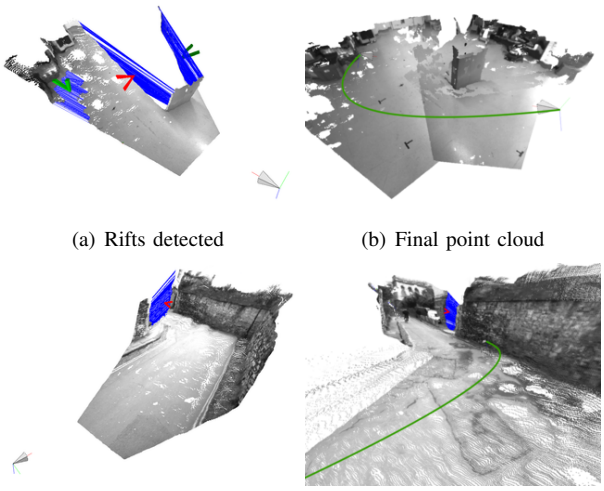
2) Local window-based stereo: Our local window-based stereo correspondence algorithm employs a number of disparity refinement and error detection stages. Using $\mathcal{I}^L$ as the reference image, we calculate correlation scores using sum of absolute differences over a square correlation window (typically 11x11 pixels^2). The disparity scores are refined using a multiple supporting window technique described by Hirschmüller et al. [14], helping to eliminate errors at depth discontinuities. For each pixel in $\mathcal{I}^L$ a search of the corresponding discrete correlation curve is performed, looking for the minimum (i.e. best) correlation score. A left/right consistency check, as proposed by Fua [15], ensures that this minimum is agreed on in both images. Only sharply defined minima are accepted as valid disparity scores, as a flat curve is indicative of low texture and poor confidence in the match. Subpixel interpolation is performed by fitting a parabola to the correlation minimum and neighbouring values, with the curve's critical point being taken as the subpixel disparity score. Finally, we filter isolated pixels by considering the local neighbourhood of each refined disparity and discarding pixels which do not share similar disparity values with their neighbours.

Fig. 1(b) shows a typical result of this algorithm.



(a) Merged graph (b) Rift (c) Path planning

Fig. 5. (a) shows detail of the graph merging operation - orange edges are new and join two graphs. (b) shows a rift with edges connecting the rift vertex to the graph. (c) shows detail of a path planned across a graph with A^*.



(a) Rifts detected (b) Final point cloud

(c) Rifts detected (d) Final point cloud

Fig. 6. (a) and (c) show the initial rifts detected in the point cloud with the dominant rift, R_d, in red. (b) and (d) show the point cloud after moving the camera to inspect R_d and performing merging and visibility checks. The camera trajectories are shown in green.

C. Surface coverage

1) Graph operations: Fig. 5 shows results from various stages of our method. Successful graph merging (Section VI) is shown in Fig. 5(a). Orange edges are new edges introduced to connect the two graphs. Fig. 5(b) shows a rift in the graph, identified using the method of Section V. The red edges connect the rift vertex to the graph, and the arrow indicates the normal direction of the rift. Finally Fig. 5(c) shows a path planned over the graph using A^*(Section VIII), and the resulting camera poses - note that the camera is always oriented to point towards the surface.

Fig.	6(a)	6(b)	6(c)	6(d)
Raw points	140789	548978	15124	605831
Graph vertices	140789	422060	151214	447503
Graph edges	280130	902108	300032	884977
Surface area	$13m^2$	$52m^2$	$177m^2$	$452m^2$

TABLE I
DATA FOR FIG. 6.

2) Real data: Fig. 6 shows the results of applying our method to real data. Figs. 6(a) and 6(b) are generated from a sequence of images from a lab environment, a photo of which is seen in Fig. 1(a). Figs. 6(c) and 6(d) are generated using images from the New College data set[16].

Fig. 6(a) shows the graph generated from a single camera pose. Rifts have been identified and highlighted with green arrows, the dominant rift, R_d, with red. Fig. 6(b) is the result of a change in camera pose to point along the normal of R_d.

Fig. 6(c) is a graph generated from an image of a street. Note the high walls ahead and to the right meaning that the obvious place to explore is the corner to the left. The dominant rift is indeed oriented such that the new camera pose will be pointing into this unknown area. Fig. 6(d) shows a view of the previously unseen workspace. The original dominant rift has been deleted due to the visibility check of Section VII, and a new dominant rift has been chosen.

The camera paths resulting in both Fig. 6(b) and 6(d) consisted of 4 intermediate poses. The graphs were merged and visibility checks were done at each pose.

Numerical data for these figures is given in Table (1). **Raw points** is the number of points without any graph merging - concatenated point clouds. **Graph vertices** is the reduced point set after merging (this is the same as raw points for single poses). **Graph edges** is the number of edges after merging. **Surface area** is a rough estimate of the world surface covered by the graph.

Fig. 6 and Table (1) show that chasing down the dominant rift has improved the completeness of the surface of the workspace. This is shown both in terms of surface area covered and from views of the resulting point clouds.

X. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this paper we have presented a method for workspace surface discovery using a stereo camera. A non-planar graph structure is constructed with observed pixels in stereo disparity images as vertices. We have shown that by careful consideration of the properties of this graph we can merge graphs from multiple camera poses, and eliminate spurious edges through a visibility check. Areas of discontinuity in the graph known as rifts are identified and ranked, and new camera poses are chosen such that these rifts are explored.

This method is shown to improve and extend the workspace surface map in a small scale lab environment and an outdoor scene. The result is a graph covering the surface of the world, which can be lifted to 3D yielding a full scene reconstruction.

B. Future Work

We currently assume that the camera poses generated by the graph search algorithm are reachable by our camera. An obvious next step is to take the physical constraints of a robot platform into account.

Robustness of this approach to poorly textured surfaces, which pose problems for stereo processing, will be explored. We plan to investigate combining stereo with a time-of-flight sensor as described in [17] to help alleviate this problem.

XI. ACKNOWLEDGEMENTS

This work was supported by an EPSRC Industrial CASE studentship with OC Robotics.

REFERENCES

[1] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schröter, L. Murphy, W. Churchill, D. Cole, and I. Reid, "Navigating, recognising and describing urban spaces with vision and laser," *The International Journal of Robotics Research*, 2009.

[2] B. Yamauchi, A. Schultz, and W. Adams, "Mobile robot exploration and map-building with continuous localization," in *ICRA*, vol. 4, May 1998, pp. 3715–3720 vol.4.

[3] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006, pp. 2276–2282.

[4] R. Sim and J. J. Little, "Autonomous vision-based robotic exploration and mapping using hybrid maps and particle filters," *Image and Vision Computing*, vol. 27, no. 1-2, pp. 167 – 177, 2009.

[5] B. Tovar, L. Guilamo, and S. M. LaValle, "Gap navigation trees: Minimal representation for visibility-based tasks," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2004.

[6] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[7] Y. Landa, D. Galkowski, Y. R. Huang, A. Joshi, C. Lee, K. K. Leung, G. Malla, J. Treanor, V. Voroninski, A. L. Bertozzi, and Y. R. Tsai, "Robotic path planning and visibility with limited sensor data." *American Control Conference*, 2007.

[8] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, and M. Pollefeys, "Real-time visibility-based fusion of depth maps," *ICCV*, 2007.

[9] B. Morisset, R. Rusu, A. Sundaresan, K. Hauser, M. Agrawal, J. Latombe, and M. Beetz, "Leaving flatland: Toward real-time 3d navigation," *ICRA*, 2009.

[10] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *ICRA*, 2009.

[11] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[13] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[14] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 229–246, 2002.

[15] P. Fua, "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, 1993.

[16] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595 – 599, May 2009.

[17] Y. Chan, P. Delmas, G. Gimel'farb, and R. Valkenburg, "On fusion of active range data and passive stereo data for 3d scene modelling," *Image and Vision Computing New Zealand*, 2008.