A Review of Image-Based Modelling Techniques

Pier Luigi Dragotti and Mike Brookes
Electrical and Electronic Engineering Department, Imperial College London, Exhibition Road, London, SW7 2BT

Abstract

This paper presents a categorisation of image-based modelling (IBM) algorithms based on their approach to the IBM problem. This categorisation groups the algorithms into three classes, surface reduction, depth map and sparse point algorithms. We then cover a further four key considerations or aspects related to choosing and designing an IBM algorithm, scene type, prior information, type of features and similarity measures. Finally the paper presents an implementation of an IBM algorithm based on the previous discussion and shows some simulation results using data from the Middlebury website www.middlebury.edu/stereo.

Keywords : Image-based modelling, Multi-view stereo

1. Introduction

Visual media is currently undergoing the biggest change in decades, the move from 2D to 3D scene representation. For instance broadening the user's experience to that of the real world with 3DTV or free viewpoint TV (where the user decides upon the viewpoint). Moving away from media entertainment, 3D scene representation, such as virtual viewpoint synthesis or 3D modelling, is also used in object tracking and recognition. These applications are a product of the expanding research into multi-view imaging, in particular the fields of Image-Based Rendering (IBR) and Image-Based Modelling (IBM). IBR and IBM form the two extremes of 3D scene representation [1].

In IBR arbitrary new virtual views of a scene are interpolated from a finite set of multi-view images without the need to generate a 3D model, hence giving the impression of a 3D scene. This is achieved by considering each image as capturing a set of light rays travelling from an object or scene to the camera [1]. Under this model the virtual view is simply determined by selecting the correct light rays from the image set. The interpolation is required as the image set is finite, hence it does not contain all possible light rays. As a result the intensity values of the arbitrary new views are found by interpolation of nearby light rays [2].

The light rays in question are described using the following seven dimensional function proposed by Adelson and Bergen [3], known as the *plenoptic function*

$$I = I_7(\theta, \phi, \lambda, t, v_x, v_y, v_z) \qquad (1)$$

where (v_x, v_y, v_z) is the viewing position, (θ, ϕ) is the viewing direction, λ is the wavelength and t is the time. In practice, as images are used, the viewing direction is parameterised in terms of (x, y) coordinates [3]. Figure 1 illustrates the plenoptic function parameters.

Bearing this in mind, IBR can be viewed in terms of sampling and reconstruction of the plenoptic function. The finite set of images with finite resolution, samples the continuous plenoptic function and the virtual view is the reconstruction of the samples [1].

At the other extreme, the goal of IBM is to reconstruct a 3D model of a scene or object given a set of 2D images from multiple viewpoints [4]. Therefore the 3D information (i.e. geometry) of the scene is recovered and stored from the multi-view images. This means that the scene or object is represented directly as a 3D model, although virtual views can be formed via projection of the 3D model into the image plane.

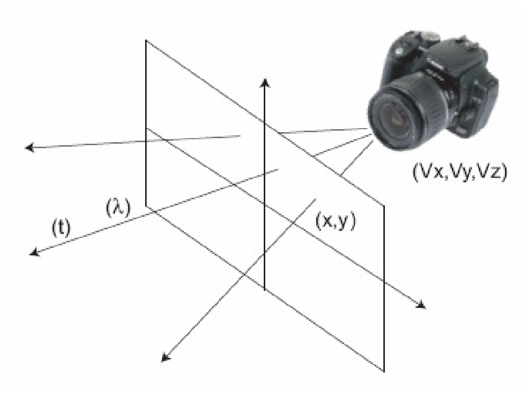

Figure 1: Diagram showing the seven parameters of the plenoptic function, where (v_x, v_y, v_z) is the viewing position, (x,y) is the viewing direction, λ is the wavelength and t is the time [2]

This process is a generalisation of the classic stereo reconstruction problem, hence also being known as multi-view stereo reconstruction. For an overview of stereo reconstruction algorithms see [5].

Consequently both approaches require a set of multi-view images of a scene; they differ, however, in the size of that set, the type of data that is stored and the type of 3D representation they provide.

Focusing on the size of the image set first, high quality view synthesis in IBR can only be achieved with a very large image set, whereas in IBM a much smaller set can be used to generate a 3D model. Consequently IBR requires that the large image set be stored, compared to a considerably smaller image set plus 3D model in the case of IBM. Lastly IBR, by definition, provides renderings from any angle of the scene, whereas IBM provides the 3D model directly. Hence IBR gives the impression of 3D compared to the actual 3D model generated by IBM.

Bearing this in mind, the remainder of this paper will focus on IBM algorithms. The paper is organised as follows, section 2 provides an overview of IBM algorithms, dividing them into three main classes based on their approach to the problem. In general no one algorithm works best for all scenes and certain aspects of an algorithm can be varied to achieve different results, such as feature points or similarity measures. As a result section 3 will examine key considerations when constructing an IBM algorithm. Lastly section 4 describes our implementation of an IBM algorithm based on the previous discussion with some simulations results and section 5 concludes the paper.

2. IBM Algorithms Classification

A recent survey of multi-view stereo reconstruction by Seitz et al. [6] classifies the algorithms according to six fundamental properties: scene representation, photo-consistency measure, visibility model, shape prior, reconstruction algorithm and initialisation requirements. However in this paper we consider a coarser and looser classification scheme, in which algorithms are classified based on their approach to the problem. As such there are three classifications, surface reduction approach, depth map based approach and sparse point approach. Note that details such as feature points and similarity measures are explained in section 3.

2.1 Surface Reduction Algorithms

Algorithms in this category use the assumption that the scene (or more precisely the object as discussed later) can be modelled by a surface. Hence they approach IBM from the end product point of view, i.e. the result of the algorithm should be a surface that represents the scene. Note that in general the surface is complete, so some algorithms consider the volume enclosed by the surface.

The high level approach is to start with a rough initial surface containing the scene and refine it until it is equivalent to the scene. This refinement process means that the initial surface is reduced to fit the scene, hence it can be considered as the minimisation of the surface given some constraints. For example common constraints would be to impose smoothness on the surface, use a photo-consistency measure, such as colour consistency or a similarity measure, or to impose geometric consistency [6]. Other than that, as [7] highlights, the competing algorithms mostly differ in the type of optimisation technique used, level set method, gradient descent or graph cut.

An example of this class of algorithm is the one proposed by Pons, Keriven and Faugeras [8]. The algorithm works by using the level set technique to minimise the surface defined by a prediction error. The prediction error is generated by measuring the similarity between a point in on image and the related point projected via the scene space into another image.

2.2 Depth Map Algorithms

This class of algorithm approaches IBM from the stereo reconstruction point of view, hence treating it as a number of stereo reconstruction problems linked together. The first step is to generate the depth map of each image; hence the depth of each pixel is computed. Once computed the depth maps are then merged to form a 3D representation of the scene, either as a 3D point cloud or a surface. Figure 2 shows the basic principle of computing the depth of a point given its position in two images.

Similar to the previous category, the method of computing the depth maps per image varies depending on the algorithm. For instance the depth map can be calculated using the image space [9] or by projection from the scene space [10,11], between two images [9] or over all the images [10,11].

The algorithm proposed in [9] is an example of calculating the depth maps between two images in the image space. In this algorithm, once the camera positions are determined, a dense stereo matching is performed on adjacent views. This process tries to match each pixel in one image to a pixel in another image by pair wise disparity estimation using a dynamic programming scheme, see [12]. The dynamic programming is performed between two corresponding epipolar lines. The computation is reduced by rectifying the images beforehand so that the epipolar lines coincide with the scan-lines of the images. The depth maps are then fused by a controlled correspondence linking. Each pixel in the reference image is transferred to another view and if the resultant depth estimate is within a confidence interval the link is considered good and it carries on to the next image. If this is not the case the linking is stopped.

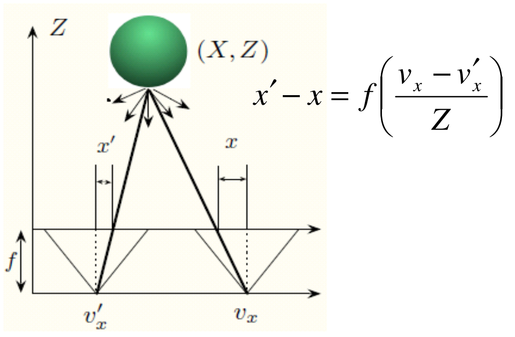

Figure 2: Diagram shows the basic principle of depth estimation of a point using frontal parallel images. Z is the depth, v′ x and vx the positions of the cameras, f is the focal length and (x′ –x) the disparity [2]

Whereas the algorithm in [10] calculates the depth maps for each image using all the other images and the scene space. The algorithm works by assuming that the scene is bounded within a volume and that for the reference image each pixel corresponds to a light ray passing through the volume. Then at each depth value along the ray the corresponding point is projected into the neighbouring images as possible matches. The match is evaluated using a similarity measure and the average value of the measure is taken across all the neighbouring images. Once this has been done for all the depth values along the ray, the one with the highest average similarity measure, above a certain threshold, is chosen, i.e. a depth is assigned to a pixel. This process is repeated for all the pixels in all the images. The thresholding imposed when choosing a depth value means that it is possible to have gaps in a depth map, however depth maps are calculated for all the images hence the gaps are likely to be filled.

2.3 Sparse Point Algorithms

The final class of algorithm approaches IBM from yet another view point, image matching and correspondence. In this category a set of feature points are extracted from all the images and then matched. The resultant sparse or patchy 3D point cloud is then turned into 3D model by fitting a surface to the points.

Although these algorithms may seem to be similar to the depth map algorithms there is a distinct difference in their aim. The aim of the depth map algorithms is to relate each pixel in the image to a depth value by determining correspondence between other view points. Hence each image should have a depth map that is as complete as possible allowing for a complete 3D reconstruction. Whereas the aim of the sparse point algorithms is to match and reconstruct key features.

An example of this class is proposed in [7]. The algorithm works by dividing each image into smaller sections and extracting a set number of features from each section. Two types of features are extracted from the image, corner features and blobs (explained in more detail in section 3). The features are then matched with features along the corresponding epipolar lines in other images. The matching process in this algorithm results in a patch being created, hence after the matching a sparse set of patches is returned. The second step is to expand these patches to nearby patches and then the final step is to filter the patches to remove outliers. The second and final steps are repeated three times to increase the patch density. The sparse patch cloud is then turned into a surface mesh.

3. Key IBM Considerations

Having defined three categories of algorithms this section looks at key considerations and aspects that are relevant to all algorithms. These are split into four groups, the types of scenes being modelled, the effect of prior information on the choice of algorithm, the choice of feature to be extracted and the type of similarity measures used for matching images. The first two groups are considerations when choosing an algorithm, whereas the last two cover interchangeable aspects related to all algorithms.

3.1 Types of Scenes

When choosing an algorithm it is important to consider the type of 3D scene that it should model. For instance some algorithms require the scene to be bounded in a volume [10] or to have images from all around the scene (surface reduction algorithms). Furukawa and Ponce, [7], define three types of scene datasets: objects, scenes and crowed scenes.

The first type of dataset consists of a single target object that is fully visible from all angles [7], hence the object is the sole focus of the image set. This means that there is almost no background information or occlusion to consider within the images. Algorithms using surface reduction are ideal for this type of dataset as the target object is bounded within a volume making an initial estimate of the surface easy, and the image set can cover a wide set of viewpoints [7].

The second dataset, scenes, consists of one or more targets that are embedded with other non-basic objects or clutter [7]. Consequently the targets could be partially occluded or the number of viewpoints constrained. An example of this dataset would be outdoor urban scenes. These more complex scenes are better suited to depth map or sparse algorithms as an estimate of the bounding volume is difficult and they deal with occlusion better. Note that if the scene is not too complex the target objects can be separated from the clutter or background using image segmentation.

The final dataset, crowded scene, is a more complex version of the second, as they contain moving objects along with the clutter. Hence they are dynamic not static scenes. In this case the best approach would be a robust sparse point algorithm as it would focus on key features only, however the reconstruction might be incomplete.

It is worth noting that a mixture of reconstruction strategies could be applied to complex scenes, hence different styles of algorithms could be used for different areas of the scene. This approach is used in [11], where the images are split into regions and different reconstruction strategies are applied depending on the characteristics of the region.

3.2 Prior Information

The most common prior information used in IBM is the assumption that the cameras have been calibrated and all the camera poses are known [7,8,10,11]. This enables the user to project the image points into the scene space and then re-project them into another image [8,10] or to calculate the fundamental matrix between two images allowing epipolar line searching [7]. A down side of this approach is that the camera parameters can not change and that the camera positions must be accurately known beforehand.

There are some algorithms that do not assume camera calibration or position; in fact they start by calibrating the cameras and determining the camera poses. Such algorithms use a self-calibration step [9,13]. This extra step increases the computation of the algorithm but does allow more freedom in the initial images used.

Other common prior information required, in particular for surface reduction algorithms, is an estimate of the bounding volume that contains the target object or objects.

3.3 Features

Extracting features from the image set is a common stage in nearly all IBM algorithms, in particular algorithms that aim to calibrate the cameras and calculate their position. The reason for this is that treating all the pixels in the image set [10] can lead to increased computation and possible indecision as some pixels are not unique.

Consequently, as indicated in [11], a good feature should be distinct, invariant to certain transformations and robust to noise, i.e. it should be possible to relate a good feature across the image set.

Bearing this in mind, [11] defines three types of features. The first type are interest points, these are single points within the image, which are determined by some criterion to be distinct or unique. For instance Harris corner points, used in [7,9], are determined by looking for the maximum gradient in both the x and y direction. Another type of interest point are those determined by Scale Invariant Feature Transform (SIFT) [14].

The second type of feature is an edge within the image. The most common edge detector is the Canny detector. The basic method is to smooth the image and then look for changes in gradients that signify edges. The downside to this type of feature is that it is not very affine invariant, hence large changes in viewpoint lead to the edge looking very different.

The final feature, defined by [11], is regions. These are regions within the image that are invariant under certain transformations [11]. The algorithm proposed in [7] uses Difference of Gaussian (DoG) operators along with Harris corners as key features, hence mixing regions and interest points. Likewise, [13] extracts affine invariant regions across the image set.

Ideally, like [7], extracting a mix of features seems the optimum strategy for matching and relating images, however there is a trade off with computation time.

3.4 Matching Similarity Measures

Once the feature points or regions are extracted from all the images in the set, the corresponding features occurring in each image need to be determined. This process involves matching the features to each other and comparing the result. In general this is done by using a similarity or dissimilarity measure calculated between the features followed by a threshold to decide if the match is good.

There are two types of similarity measures used to compare a feature p in image i to a feature q in image j; the first is a window based strategy. In this strategy a square (or rectangular) window is centred on feature p in image i and the same size window is positioned on feature q in image j. Then one of the following characteristics is calculated over the window, sum of the squared difference (SSD), sum of the absolute difference (SAD) or normalised cross-correlation (NCC) [5]. NCC is a popular measure as it normalises the result making it robust to varying illumination, it is used in [7,9,10,11].

The second type of similarity measure involves assigning a description vector to the feature. Consequently two features are compared by calculating the Euclidean or Mahalanobis distance between the vectors [11] (i.e. the shorter the distance the better the match). An example of this would be the SIFT feature which is assigned a vector of 128 components [14].

As with the decision on what type of feature to use, the decision on the similarity measure is a trade off between more accurate matching and complexity.

4. Implementation and Results

Based on the previous discussion we implemented an IBM algorithm in MATLAB using freely available code from the internet, including: Torr’s Structure and Motion Toolkit [15], Corke’s Machine Vision Toolbox [16], Ogale’s Stereo Matching Code [17] and Dey’s Cocone mesh generation software [18].

The algorithm is designed to be as broad as possible, in terms of the type of scene to be reconstructed, and to use limited prior information. As a result the algorithm is a mixture of the sparse point and depth map approaches. It extracts and matches a set of sparse points and uses them to determine the fundamental matrix relating the images. Once this is done it performs a dense stereo matching on the image pairs and fuses the result. Although the current algorithm uses calibrated images with known camera pose, a self-calibration step can be added to the algorithm in the future.

The following sections describe the different parts of the algorithm in greater detail. Note that the structure of the algorithm is shown in Figure 3.

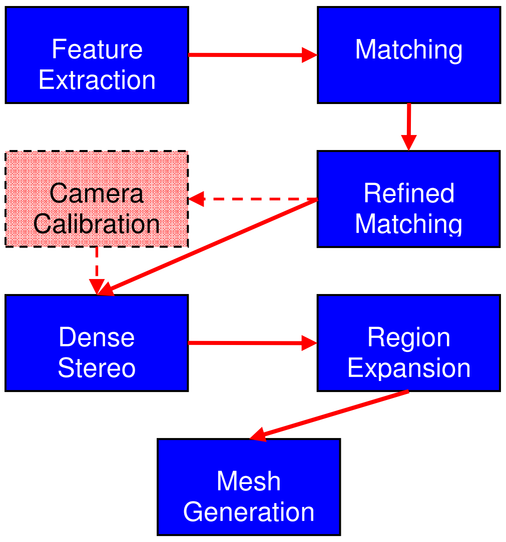

Figure 3: Diagram showing the structure of our IBM algorithm implementation. Currently the camera calibration step is not present, however it position in the algorithm is highlighted. As a result the algorithm relies on calibrated images with known camera pose

4.1 Feature Extraction and Matching

The first step in the algorithm is to extract features from the image set. It was decided that the features in question should be Harris corner points as they are not too computational expensive and commonly used in IBM algorithms.

The Harris detector code from [16] was used as it allowed greater control of the number of Harris points detected and their position.

Once all the features have been detected the next step is an initial feature matching, between adjacent viewpoints, using a NCC similarity measure with a window size of $\mu \times \mu \text{pixel}^2$ (where μ varies from 11 to 15). The threshold level for this matcher is set to 0.6. At this level a certain number of incorrect matches are expected, however they are dealt with in the next section.

4.2 Refined Matching

For each adjacent pair the following refined matching occurs, the initial matches are used to calculate a robust estimation of the fundamental matrix. The robust estimation uses a RANSAC method to determine a set of inliers and outliers, and then calculates the fundamental matrix from the inliers.

Once an estimate of the fundamental matrix is calculated a refined matching process can take place. This refined matching takes the outliers from one image and searches along the epipolar lines in the opposite image for matches (again using NCC), and then vice versa in the other image. In this search the threshold is raised to 0.9 to ensure that these matches are more reliable. The new epipolar line matches are then combined with the inliers and the fundamental matrix is re-estimated.

4.3 Dense Stereo Matching

The aim of the previous steps was to calculate an accurate estimate of the fundamental matrix between the adjacent viewpoints. This accurate estimate is now used to rectify the images so that the epipolar lines coincide with the image scan-lines.

Once this has been achieved a dense stereo matching algorithm [17] is used to generate a depth map for each image. The resultant depth maps are then fused and the matches projected into the scene space to give a 3D point cloud.

4.4 Region Expansion and Meshing

The 3D point cloud still contains some errors due to the dense stereo matching algorithm, for instance in areas where there is little or no texture. Consequently to remove these errors a region or plane expansion process is implemented. It currently assumes a frontal parallel plane, in the direction of the nearest image pair, however in the future the process will be expanded to allow any type of plane as long as the direction of the normal to the plane faces outwards. Therefore the region expansion fits frontal parallel planes to the points and removes points that are outliers, i.e. either no plane fits or it is very small.

The last step is to take the refined 3D point cloud and turn it into a mesh using the cocone algorithm [18].

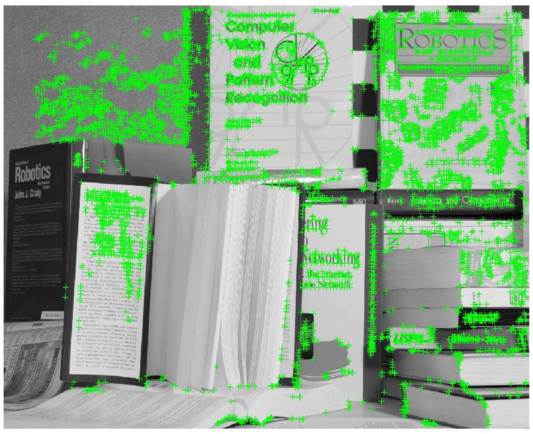

Figure 4a: Left image with the matches found after the refined matching stage. The number of matches was 6111

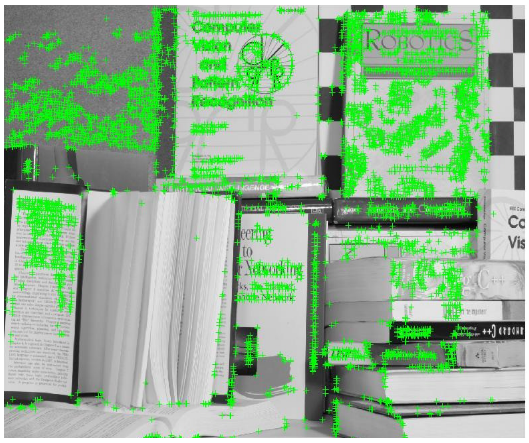

Figure 4b: Right image with the matches found after the refined matching stage. The number of matches was 6111

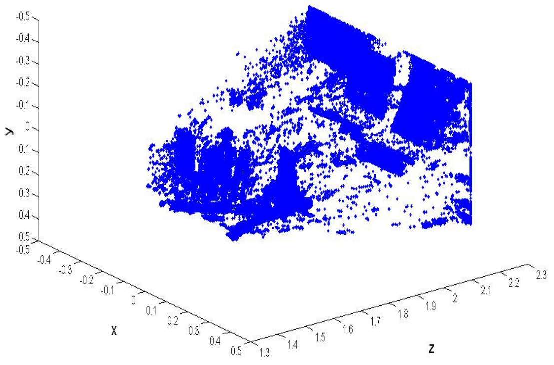

Figure 4c: 3D point cloud generated after the dense stereo matching stage of the algorithm. The noise appearance of the point cloud is due to incorrect matches occurring in the dense stereo matcher

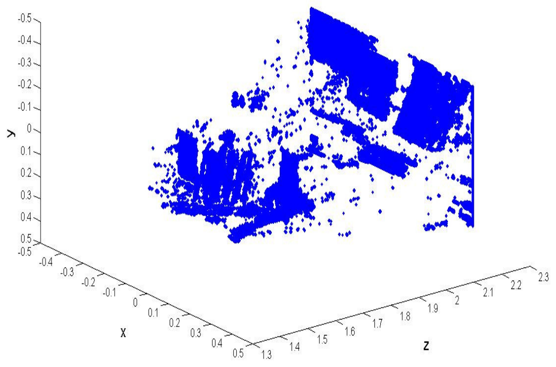

Figure 4d: 3D point cloud generated after the region expansion stage of the algorithm. The 'noise' present in figure 4c has been reduced

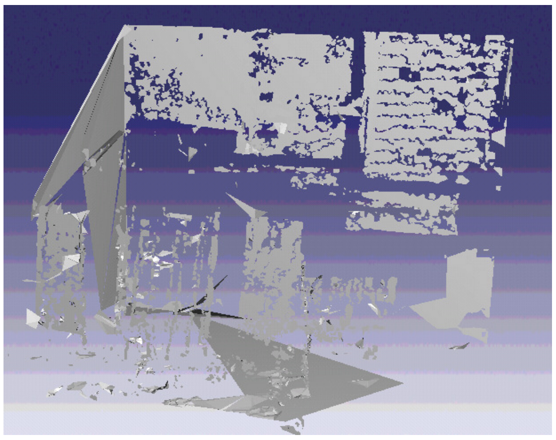

Figure 4e: Diagram showing the resultant mesh generated from the 3D point cloud shown in figure 4d. Note that the number of points in the cloud was reduced in order for the mesh generation software to work, hence the patch appearance

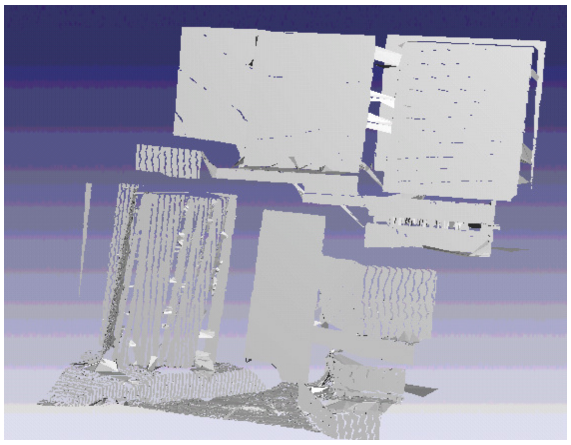

Figure 4f: Diagram showing the ground truth mesh generated from the ground truth 3D point cloud. Note, again, that the number of points in the cloud was reduced in order for the mesh generation software to work

4.5 Simulations

The algorithm was applied to a stereo pair of images from the Middlebury website, www.middlebury.edu/stereo. These simulation results are shown in Figure 4: part a and b) show the left and right image, respectively, with the refined matches, c) shows the 3D point cloud after the dense stereo matching, d) shows the refined 3D point cloud after the region expansion, e) shows the mesh generated from the refined 3D point cloud and f) shows the Ground true mesh. Note that the 3D point cloud given to the mesh generating software was down-sampled uniformly as the software can not handle very large point clouds. As a result the mesh in e) looks patchy compared to the point cloud in d).

5. Conclusion

IBM algorithms can be grouped into three categories based on their approach to the IBM problem. The first category refines an initial estimate of the surface until it matches the scene, hence a surface reduction approach. The next category generates a dense depth map for each image and fuses the results, hence the depth map approach. The final category reconstructs key features and fits a surface to them, hence the sparse point approach.

Along with this categorisation, there are two key considerations when choosing an algorithm. The first consideration is the type of scene that is being modelled. For instance a single object on its own, which is ideal for surface reduction algorithms, or several objects embedded in a complex scene, which suits depth map algorithms. The second consideration is the availability of prior information, such as a bounding volume for surface reduction algorithms or calibrated cameras and poses to avoid a self-calibration step.

Once the algorithm has been chosen there are another two aspects to consider. The first aspect is the type of features to be extracted for the purpose of image matching. This is a trade off between the complexity of extracting the feature and its uniqueness. There are three types of features, a single interest point, an edge or a region. The other aspect is the similarity measure used to find matched correspondence between two features. This splits into two groups, window based measures using a characteristic like NCC or comparing feature descriptor vectors by calculating a distance (e.g. Euclidean distance).

Bearing this in mind this paper presents an implementation of an IBM algorithm designed to cover any type of scene and use limited prior information. The algorithm initially uses a sparse point approach to the problem by extracting a set of interest points (Harris corners) and matching them across the image set. It then switches to a depth map approach by determining the fundamental matrix relating the images and performing a dense stereo matching on the image pairs. Lastly it fuses the depth maps and generates a mesh. The software used in the algorithm is all freely available on the internet; see [15,16,17,18].

The simulation results show that the presented algorithm is capable of reconstructing the scene. However they also highlight some limitations of the algorithm. The first is that the dense stereo matcher results in a noisy 3D point cloud, hence requiring an extra processing step (the region expander) to reduce this noise. The other major limitation is that the mesh generation software is limited to a certain size of 3D point cloud. Consequently the 3D point cloud is down-sampled in order for the mesh to be generated, leading to a patchy output.

Currently the algorithm is limited to generating a patchy 3D model (or mesh), therefore future expansion to the algorithm would be to improve the completeness of the output. One way to do this would be to improve the region expansion step to use planes with varying orientation, i.e. not just frontal parallel planes.

References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, *Multiview imaging and 3DTV*, IEEE Signal Processing Magazine, vol. 24, no. 6, pp. 10–21, 2007.

[2] J. Berent. *Coherent multi-dimensional segmentation of multiview images using a variational framework and applications to image based rendering*. PhD thesis, Imperial College London, 2008.

[3] E.H. Adelson and J.R. Bergen, *The plenoptic function and the elements of early vision*, in Computational Models of Visual Processing, pp. 3-20. MIT Press, Cambridge, MA, 1991.

[4] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer, *A survey of methods for volumetric scene reconstruction from photographs,* in International Workshop on Volume Graphics, 2001, pp. 81–100.

[5] D. Scharstein and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,* International Journal of Computer Vision, vol. 47, no. 1–3, pp. 7–42, 2002.

[6] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, *A comparison and evaluation of multi-view stereo reconstruction algorithms*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, vol. 1, pp. 519–528.

[7] Y. Furukawa and J. Ponce, *Accurate, dense, and robust multi-view stereopsis*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), 2007, pp. 1–8.

[8] J.-P. Pons, R. Keriven, and O. Faugeras, *Modelling dynamic scenes by registering multi-view image sequences*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, vol. 2, pp. 822–827.

[9] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, and J. Tops, *Visual modeling with a hand-held camera*, International Journal of Computer Vision, vol. 59, no. 3, pp. 207–232, 2004

[10] M. Goesele, B. Curless, and S.M. Seitz, *Multi-view stereo revisited*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, vol. 2, pp. 2402–2409.

[11] F. Remondino, S.F. El-Hakim, A. Gruen, and L. Zhang, *Turning images into 3-D models*, IEEE Signal Processing Magazine, vol. 25, no. 4, pp. 55–65, 2008.

[12] I.J. Cox, S.L. Hingorani, and S.B. Rao, *A maximum likelihood stereo algorithm,* Computer Vision and Image Understanding, vol. 63, no. 6, pp. 542–567, 1996.

[13] C. Strecha, T. Tuytelaars, and L. Van Gool, *Dense matching of multiple wide-baseline views*, in Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV'03), 2003, pp. 1194–1201.

[14] D. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, vol. 60, no.2, pp. 91-110, 2003.

[15] P. H. S. Torr, *A Structure and Motion Toolkit in Matlab*. Technical report, Microsoft Research, 2001. Website: http://cms.brookes.ac.uk/staff/PhilipTorr/

[16] P. I. Corke, *Machine Vision Toolbox*, IEEE Robotics and Automation Magazine, vol. 12 no. 4, pp 16-25, 2005. Website: http://petercorke.com/Machine%20Vision%20Toolbox.html

[17] A. S. Ogale, *Shape and the stereo correspondence problem*, International Journal of Computer Vision, vol. 65, no. 3, 147-162, 2005. Website: http://www.cs.umd.edu/~ogale/download/code.html

[18] T. K. Dey and J Giesen, *Detecting Undersampling in Surface Reconstruction*, in Proceedings 17th ACM Symposium Computer Geometry, 2001, pp. 257-263. Website: http://www.cse.ohio-state.edu/~tamaldey/cocone.html

Acknowledgements

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.