Efficient generation of 3-D models out of depth maps

F.E. Ernst, C.W.A.M. van Overveld, P. Wilinski

Philips Research
Prof. Holstlaan 4
5656 AA Eindhoven, The Netherlands
Email: fabian.ernst@philips.com

Abstract

Depth reconstruction or acquisition with a 3-D camera results in a video sequence where each pixel of a frame is annotated with a depth value. We propose an approach to combine this depth information into a 3-D model of the assumedly static scene. For processing of video, we have to satisfy strong requirements on computational and storage efficiency, incremental processing and handling of uncertainty in the depth maps. Our approach is based on an octree structure. Cells are split until the topology of the scene can be recovered uniquely. A signed-distance function to the nearest surface is computed in the octree vertices. The scene can then be reconstructed by extraction and triangulation of an implicit surface. The depth uncertainty results in boundary conditions for the implicit surface.

Keywords: 3-D reconstruction; signed distance; isosurface extraction

1 Introduction

When a scene is filmed, the resulting video sequence contains implicit information on the 3-D geometry of the scene. While for adequate human perception this implicit information suffices, for many applications the exact geometry of the 3-D scene is required. One category of these applications is when sophisticated data processing techniques are used, for instance in the generation of new views of the scene, or in the reconstruction of the 3-D geometry for industrial inspection applications. Annotating an image with depth data is not always sufficient to allow the required data processing techniques to be used.

In this paper, we discuss a new method to create a 3-D model of a scene. Currently, we assume that the scene is static, and may contain multiple rigid 3-D objects. We create the model by combining information from a series of depth maps, which associate with each pixel on the image plane a most likely depth value d_{ML}, and error bounds on the depth measurements. The end points of the depth intervals give information on the *topology*, whereas d_{ML} increases the *accuracy* of the position of the surfaces of the objects. These depth maps can be created from two images using structure-from-motion algorithms, through active acquisition techniques (e.g., structured light) or passive acquisition techniques (e.g., laser scanning). Furthermore, we assume that the position and orientation of the camera is known (i.e., we have calibrated cameras), or has been obtained by a camera calibration algorithm. We assume that no camera position is inside an object.

If video streams are to be processed, it is required that the method should be able to incrementally update the 3-D model with the arrival of new frames. Since the method should eventually run in real time, the computational complexity should be low, and the method should be fully automatic and not require interactive user input. Usually little a-priori information is available on the scenes, so it should be possible to represent any 3-D model (consisting of multiple, non-connected objects). Since the number of video frames can be large, the storage should be efficient. The representation should also be able to handle the uncertainty in the depth information in a consistent way.

2 Existing approaches

Generation of 3-D models out of depth data has generated a large amount of interest in the vision community. In *parametric* approaches, objects are represented by fitting parameters of some primitive function to the data [11, 14]. Although the amount of parameters is limited, their determination is usu-

ally highly nonlinear in the input data. Moreover, there is a large amount of freedom in the choice of the primitives. Arbitrary, complex, surfaces often require a high number of parameters. Another class of approaches creates a 3-D model by *merging triangular meshes* generated from depth maps (or from point clouds [7]), and merging them into a combined mesh. Merging takes place by stitching them together explicitly [15], combine them into a single co-ordinate system [16], or by first generating a function in 3-D space based on the meshes and then extracting an implicit surface [2, 6]. The concept of mesh merging methods is rather intuitive. However, since we are combining 2-D information in 3D, a lot of attention has to be paid to the accurate 'stitching together' of the meshes. This can make the methods quite complex on an implementation level. In *volumetric approaches* [4, 3, 5], a so-called 'universe' is divided into volume elements (voxels). Subsequent depth maps are used to decide which parts of the universe are 'empty space', and which parts consists of 'objects'. The volume elements can be cubes [3, 5] or tetrahedra [4]. The size of the voxels is decreased recursively until an acceptable resolution is reached, and can be stored in a tree-based structure. The disadvantage of this type of method is that, especially for scenes with a lot of curved surfaces, a very large amount of volume elements is needed to obtain the required accuracy, making storage expensive.

Recently, Kunii [8] proposed to partially overcome these limitations by defining the *essential information* in the scenes as the location of the *singularities* and storing those in an *octree* (the 3-D equivalent of a binary tree; in an octree, each cell can be split into 8 children cells). The singularities are the vertices, edges and bounding surfaces of the objects in the scene. Each object is bounded by *surfaces*. The surfaces are bounded by *edges*. These in turn have as end points *vertices*. In this way each object can be built from a hierarchy of singularities, with vertices at the lowest level, then edges, then surfaces and finally the object itself. (Note, however, that the hierarchy does not have to start at the vertex level, e.g. in the case of a ball.) The main strength of the Kunii method is that the storage is extremely efficient through use of the octree and terminating the subdivision stage at an early level: As soon as the structure within the cell is simple enough (i.e., if a cell contains only one singularity of the lowest order), and not only when a cell is completely inside or outside an object.

A major obstacle in applying the Kunii method for this application is the extraction of the singularities (essential features) from the depth maps. First of all, accurate localization of vertices and edges from images or depth maps has already generated a vast amount of literature on, e.g., corner detectors, edge detectors and segmentation algorithms, but no suitable general purpose algorithm exists yet. Even if an adequate detector of singularities were available in 2D, these singularities might be just apparent singularities and not real ones. All locations on a curved surface which we see under an angle of 90 degrees seem to be singularities in the image. The extraction of singularities can therefore not be done just from a single image. For a (near) real-time application, identification of singularities by a human operator is no viable solution.

In this paper, we propose a volume-based 3-D reconstruction method which adapts the approach by Kunii to make it suitable for automatic, robust and incremental processing.

3 Octree generation

3.1 Alternative splitting criterion

The essence of the Kunii approach is that the subdivision of the octree is already halted at an early stage: as soon as the description of the object within a cell can be uniquely specified. To circumvent the problem of singularity extraction, we propose to replace the single-singularity criterion by: *A cell should not be split if the topology of the surface within the cell can be derived uniquely from the information at the cell vertices.*

To illustrate this criterion, we give an example for the most simple case. Assume that for each cell, we know for each of its 8 vertices whether they are inside or outside an object. We can show that for the configurations where the topology of the surface can be uniquely reconstructed, the set of "inside" vertices and the set of "outside" vertices both form a connected set. E.g., if vertices 0, 2, 4 and 6 are inside, and 1, 3, 5 and 7 outside, the surface crosses the cell more or less vertically. If, on the other hand, vertices 0, 3, 4 and 7 are inside, and 1, 2, 5 and 6 are outside there are two possible configurations (see figure 1).

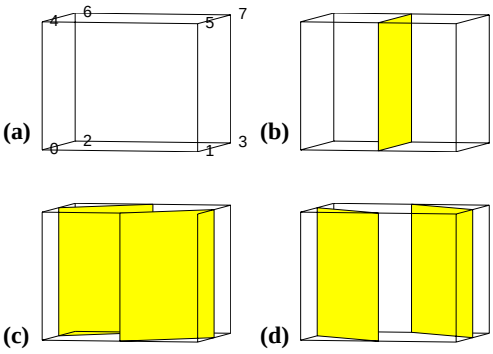

Figure 1: The uniqueness criterion. **(a)** Indexing used for the cell vertices. **(b)** This configuration specifies the topology of the surface in a unique way. **(c)**, **(d)** Although the configuration of inside (0,3,4,7) and outside (1,2,5,6) cell vertices is exactly the same, there are two possible ways how the surfaces can intersect the cell.

A consequence of the uniqueness assumptions is that each face and each edge of the cell may not be crossed by the surface more than once. If neighbouring cells (sharing either a face or an edge) in the octree have unequal sizes, we know for the larger cell not only whether its vertices are inside or outside, but have this information also at some other locations on the edges or faces. The information of these extra points might lead to the conclusion that the single-singularity criterion is no longer satisfied (see figure 2). If such a situation is encountered, the larger cell has to be split.

Also each object should be contained in at least two cells (this avoids cells completely containing an object: from the values at the cell vertices, we would expect the cell to be completely outside, whereas the cell does contain an object). The connectivity of the subsets, augmented with the checking of the above assumptions, can therefore be used as the criterion to decide whether a cell should be subdivided or not. With this alternative criterion, the advantage of the Kunii approach – efficient storage by subdividing cells no further than needed – then remains, while avoiding the problem of singularity extraction. This criterion can be completely resolved knowing whether cell vertices are inside or outside.

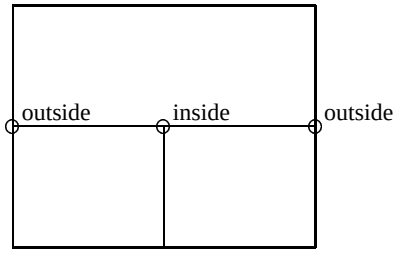

Figure 2: The upper cell should be split because its lower edge intersects the object surface twice.

3.2 Improving the accuracy

If we store in each cell vertex whether it is inside or outside objects, the topology of the surface can be recovered uniquely. However, its exact location within the cell is only determined with an accuracy of the cell size. We therefore extend the information in the cell vertex with quantitative information.

We propose to compute a *signed-distance function* u from the generated depth maps, where $u(\mathbf{x}) = 0$ at the boundary of an object; $u(\mathbf{x}) > 0$ inside objects and $u(\mathbf{x}) < 0$ outside objects. The absolute value $|u|$ denotes the distance to the nearest point of an object boundary, which may lie in any direction. Signed-distance function are closely related to level sets [12] and have become popular in computer vision applications as well [7, 2, 6, 5]. The boundaries of an object can be completely reconstructed from the signed-distance function by computing the isosurface $u = 0$ (see also [5]). This results in a gain in accuracy of the order of the cell size compared to just binary labeling (inside/outside).

The signed-distance function can be computed incrementally from subsequent images. We define $u(\mathbf{x}|\theta)$ as the signed distance at cell vertex $\mathbf{x}$ if we look in direction θ. If we are given the depth map of a single camera with the eye at $\mathbf{e}$, we look in direction $\mathbf{x} - \mathbf{e}$, and we have:

$$u(\mathbf{x}|\mathbf{x}-\mathbf{e}) = \| \mathbf{x}-\mathbf{e} \| - d_{ML}(\xi,\nu)\frac{\| \mathbf{x}-\mathbf{e} \|}{\mathbf{k}\cdot(\mathbf{x}-\mathbf{e})}, \quad (1)$$

where ξ and ν are the image-plane co-ordinates of the projection of $\mathbf{x}$ on the image plane, and $\mathbf{k}$ is the viewing direction. Note that u is only defined if (ξ, ν) lies within the image plane. This approximation of the signed-distance function is related to the

first object boundary seen from the camera eye $\mathbf{e}$ in direction $\mathbf{x} - \mathbf{e}$.

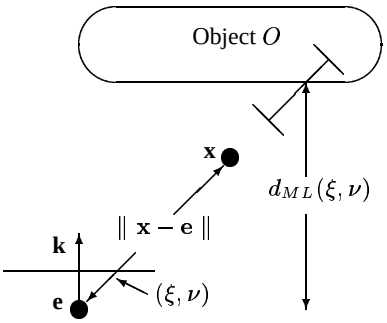

Figure 3: Quantities used in the computation of the signed distance function.

To combine the information from multiple depth maps, we have to define how to merge the information for $u(\mathbf{x}|\theta)$ into a single value for $u(\mathbf{x})$. A weighted averaging procedure has been proposed in [2]. Here, we take as the new best approximation u given the current approximation of the signed-distance function u_k and a new candidate v_k:

$$\begin{aligned} |u| &= \min\left(|u_k|, |v_k|\right), \\ \operatorname{sgn}(u) &= \begin{cases} 1, & u_k > 0, v_k > 0 \\ -1, & \text{else} \end{cases} \end{aligned} \quad (2)$$

This approximation is based on the following two observations:

- The signed-distance function is defined as the distance to the closest surface in any direction. Hence, $|u(\mathbf{x})| = \min_\theta |u(\mathbf{x}|\theta)|$. (See also figure 4.)
- If a cell vertex $\mathbf{x}$ is invisible from a certain camera position, it gets (with equation (1)) a positive value for the signed distance. However, we do not know whether $\mathbf{x}$ is inside, or behind the object. On the other hand, if $u < 0$, we know for certain that $\mathbf{x}$ is outside all objects: we are able to see through it. Therefore a negative value of the signed-distance function prevails over a positive one. (This is analogous to space-carving algorithms [9].)

3.3 Handling depth uncertainty

So far, we have been discussing deterministic values of depth and signed-distance functions. In reality,

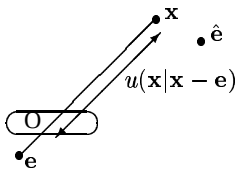

Figure 4: The minimum absolute value criterion: If $\mathbf{x}$ is seen from $\hat{\mathbf{e}}$, the next boundary in direction $\mathbf{x} - \hat{\mathbf{e}}$ might be very far away. However, from the camera at $\mathbf{e}$ we know that the closest object boundary (i.e., of O) is at most $|u(\mathbf{x}|\mathbf{x} - \mathbf{e})|$ away.

however, our depth maps have a stochastic nature in the sense that upper and lower bounds of the depth are given, together with the best guess. The depth uncertainty information allows us to mitigate the effects of errors and outliers in the depth information.

For each depth measurement, we can define three regions: A region which is definitely outside, a region containing an object boundary (the so-called 'thick wall' region), and a region which is behind the object boundary when seen from this view point (see figure 5). Note that we can not say that it is definitely inside, since this region might not even contain points which are inside objects (see figure 5b); basically we do not have information on this region since we can not see it.

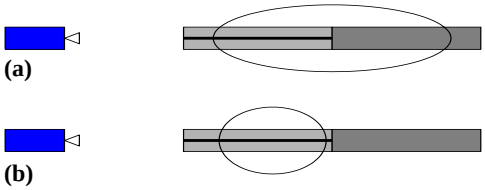

Figure 5: **(a)** Illustration of the regions: the thick line is the measurement; the light gray part is the thick wall region and the dark gray part is the inside region. Note that the inside region extends beyond the object (the ellipse). **(b)** The inside region does not have to contain any points inside the object: due to the large error bound the complete object is already contained in the thick wall region.

We incorporate uncertainty by defining a trinary 'region value' (outside, thick wall and inside) and associating it with each cell vertex. This region value can be found in a similar way to the sign of

the signed distance function. As an ordering relation, we introduce

$$\text{outside} \prec \text{thick wall} \prec \text{inside} \qquad (3)$$

If a new depth map arrives, the region value is update by taking the minimum region value of the current value and the measurement, where the minimum is defined according to ordering relation (3). The reasoning underlying this ordering relation is the following: If a point is seen from anywhere as being outside any object (free space), we have seen through it and it can not be anything else than free space. Since we do not have any information on the inside region, it is overruled by thick wall information, since that means that there is an object boundary in that region. If the depth uncertainty is zero, this reduces to the signed-distance ordering relation. See figure 6.

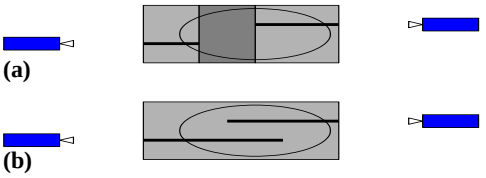

Figure 6: **(a)** The updating procedure for two depth measurements (thick lines; here assumed to be on the same line but plotted slightly offset for clarity); the light gray part is the thick wall region and the dark gray part is the inside region. **(b)** If the depth error interval is large, the inside region may collapse.

3.4 Processing of depth maps

During initialization, we set the boundaries of the universe we work in (specified as a cube); this is the root of the octree. Initially, the signed-distance function at each cell vertex in the initial structure is set to infinity and its region value to 'inside'. For every depth map, the following processing sequence is then applied:

- Read new depth map d_i and corresponding camera parameters for image i.
- Update the values for the cell vertices in octree:
  - For each cell vertex $\mathbf{x}_k$ in the octree, compute $v_k = u(\mathbf{x}_k|\mathbf{x}_k - \mathbf{e}_i)$ according to equation (1).
  - Update u_k by finding the new best approximation from u_k and v_k using equation (2).
- Check for each cell whether it needs to be split according to the uniqueness criteria. If so, it is split and the cell vertex values are updated. This continues until no more cells need to be split.
- Finally, update the region values for all cell vertices (since this does not influence the octree structure, we can do it after all splitting has taken place).

4 Triangulation of the isosurface

Once we have processed all depth maps, the object boundaries are given by the isosurface $u(\mathbf{x}) = 0$. We have to approximate the isosurface on an octree-based structure. For computational reasons we opt for a triangulation. All triangles are constrained to lie in the "thick wall" region. This is guaranteed if no triangle intersects an octree edge of which the end points are either both inside points or both outside points.

The extraction of isosurfaces from tree-based structures is a well-known problem. A basic algorithm for isosurface generation through triangulation is the marching-cubes algorithm [10]. In its original form, the algorithm triangulates each cell of a regular grid separately into a (small) number of triangles. On each cell edge crossing the isosurface (we call this a *zero edge*) a point is taken as a triangle vertex. This (possibly non-planar) polygon is then triangulated. The C^1-discontinuities, which are unavoidable in triangulations, are to a large extent on the cell faces. This is undesired since the isosurface should not depend on the underlying data structure, and the alignment of many edges with the co-ordinate axes causes annoying visible artifacts. For octrees, where adjoining cells may have a widely different size, a per-cell triangulation requires some effort to enforce (C^0-)continuity of the isosurface over cell boundaries to avoid the so-called "crack" problem [1].

To avoid the crack problem and the visual artifacts of marching cubes, we base our approach on the zero edges. Each edge has either three or four adjoining cells. We select in each cell one point, which will be called the *zero point*. The locations for the zero points can for instance be found through

interpolation of u, or can be chosen such that an adequate error functional is minimized. For each edge, the polygon spanned by the zero points of the adjoining cells is triangulated. Note that the isosurface is continuous, and the location of C^1-discontinuities is at the zero points and triangle edges.

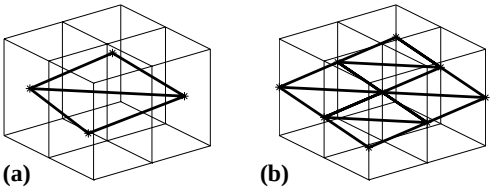

Figure 7: **(a)** Triangulation of (part of) the isosurface crossing the four cells using our algorithm. The triangle vertices are in the interior of the cells; the zero crossing on the edge where the four cells join gives rise to two triangles (thick lines). **(b)** Triangulation of the isosurface crossing the four cells using marching cubes. The triangle vertices are on the cell edges; in each cell two triangles (thick lines) are necessary.

Compared to marching cubes, the main differences are that the triangle vertices are in the inside of the cell instead of on its edges, and the triangles connect interior points of neighbouring cells instead of edge points of the same cell (see figure 7). Since each edge might have multiple neighbouring cells, we have to find the one for which this edge is also a zero edge (see figure 8).

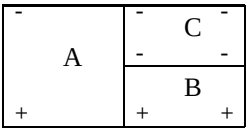

Figure 8: For cells with varying size, we need to find the relevant neighbor. Here, the relevant right neighbour of cell A is cell B, not cell C.

Since the locations of the zero points within the cells are in general not regularly distributed (such as always in the center of the cell), the triangle vertices will not be aligned with the co-ordinate directions. This avoids the visibly annoying artifacts of the marching-cubes algorithm.

5 Examples

To illustrate our approach, we have applied the method to a synthetic example (where ground truth is available), and to a real example. The synthetic example serves to illustrate the method and allows for some quantitative conclusions on the accuracy. The second example on a real video sequence of a statue illustrates the method for a realistic situation.

5.1 Synthetic data example

The scene to be reconstructed consists of two objects: a cube which is diagonally behind a ball. The edge length of the cube is 200 units, the radius of the ball 175 units. We reconstruct the objects from six depth maps with small uncertainty, of resolution 720×576 pixels, where the camera is positioned in front, to the left, to the right, behind, above and below the scene. In the depth maps, the objects partially occlude each other.

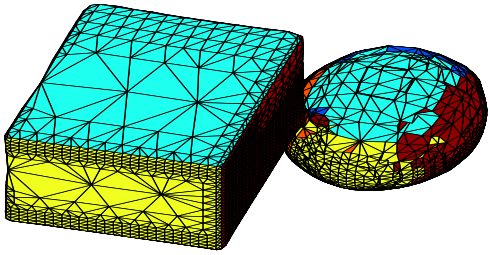

Figure 9: View of the 3-D reconstruction. The octree structure of the 3-D model can be inferred from the varying sizes of the triangles.

In figure 9 we show a view of the reconstruction. The universe in which the objects were reconstructed was the area which was visible from all six camera positions. The zero points in the cells have been found by a weighted interpolation of the signed-distance function. The octree structure can be inferred from the variation of the triangle size. The average distance to the surface of the triangle vertices of both the ball and the cube was around 2.9, resp. 1.5 units (1.5% of the radius, resp. 1% of the edge length). Most octree cells had an edge length of 16 or 32 units; the size of the largest cell was 32 times the size of the smallest cell.

5.2 Real data example

The second example is based on a video sequence of a statue of Dionysios, which has been acquired with a hand-held camera (see figure 10 for a representative frame). To prepare the data, we used an adapted version of the camera calibration described in [13], and the structure-from-motion algorithm from [17] to generate per-pixel depth maps with uncertainties (see also figure 11).

Figure 10: Original frame of the Dionysisos sequence.

Figure 11: Typical depth map for the Dionysisos sequence (same frame as in figure 10).

We have selected five depth maps which together span the viewing area of the sequence. These depth maps, together with their corresponding camera positions and orientations, form the input to our overtime integration algorithm. In figure 12 we show the resulting wireframe. In figure 13 a different view of the statue is shown, taken from a view point not present in the original sequence. Due to the absence of ground truth, a quantitative assessment of the reconstruction error in 3-D is not possible; hence a qualitative assessment is made by inspection.

Figure 12: The wireframe of the 3-D reconstruction of the Dionysios statue.

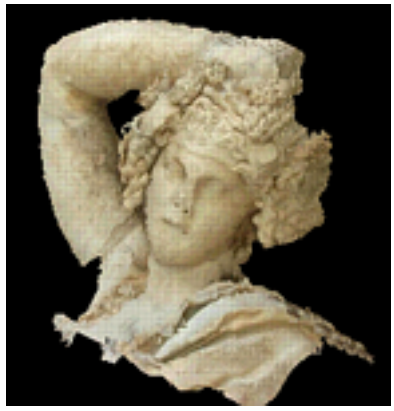

Figure 13: Reconstruction of the Dionysios statue, seen from a position from which it has not been filmed.

6 Conclusion and discussion

We have presented an approach to create 3-D models out of sequences of depth maps (range data) with associated uncertainties. The 3-D model is represented by an *octree*, where the cells have to satisfy the requirement that the behaviour of the object surfaces within the cell can be derived in a unique way from properties defined at the cell vertices. In this way, the amount of cells in the octree is minimal. Once all depth maps have been processed, we estimate an isosurface, which corresponds to the object boundaries. Uncertainty in the depth maps is explicitly taken into account as boundary conditions on the surface. The triangulation proposed here is similar to the traditional marching-cubes algorithm.

However, to avoid discontinuity problems on cell faces and annoying visible artifacts, we locate the triangle vertices in the interior of the cells.

The main strong points of our algorithm are its very efficient storage (the octree has the minimum number of cells which still allows an unambiguous reconstruction of the objects), while its computational complexity is low due to the incremental approximation of the signed-distance functions and the region values, and its regular structure. The amount of computations is linear in the volume of the input data. The main advantage over the approach proposed by Kunii is that there is no need to do the (ill-defined) extraction of singularities from the images and depth maps. Checking whether a cell should be split is solely based on the information at the cell vertices.

Since the reconstructed 3-D model is consistent with the separate depth maps and their associated uncertainties, it enables us to reconstruct (the geometry of) the original images, where noise effects have been mitigated. The 3-D model might also be used as a priori information for depth reconstruction of other images in the video sequence, thus enabling *improvements in the efficiency and accuracy.* The 3-D model can be used for *industrial inspection*, e.g., by computing certain geometrical quantities such as volumes and comparing them with the specifications. It can also be used for *new view generation* by rendering the scene based on a new view point (see figure 13). Optimally merging information on the appearance is a subject of current research.

Acknowledgement

The research presented here was partly sponsored by the ITEA BEYOND project. The Dionysios sequence has been provided by Marc Pollefeys of KU Leuven.

References

[1] J. Bloomenthal. Polygonization of implicit surfaces. *Computer-Aided Geometric Design*, 5:341–355, 1988.

[2] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH'96*, pages 303–312, 1996.

[3] C. Dorai, G. Wang, A.K. Jain, and C. Mercer. Registration and integration of multiple object views for 3D model construction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:83–89, 1998.

[4] Olivier Faugeras. *Three-dimensional computer vision. A geometric viewpoint.* MIT Press, Cambridge, 1993.

[5] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Computer Graphics Proceedings SIGGRAPH 2000*, pages 249–254, 2000.

[6] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Implicit surface-based geometric fusion. *Computer Vision and Image Understanding*, 69:273–291, 1998.

[7] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, J. Hin, H. McDonald, J. Schweitzer, and W. Stuetzle. Surface reconstruction from unorganised points. *Computer Graphics*, 26:71–77, 1992.

[8] T.L. Kunii, Y. Saito, and M. Shiine. A graphics compiler for a 3-dimensional captured image database and captured image reusability. In *Proceedings of IFIP workshop on Modelling and Motion Capture Techniques for Virtual Environments (CAPTECH98)*, Heidelberg, 1998. Springer.

[9] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:188–195, 1995.

[10] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface reconstruction algorithm. *Computer Graphics*, 21:163–169, 1987.

[11] S. Muraki. Volumetric shape description of range data using "Blobby Model". *Computer Graphics*, 25:227–235, 1992.

[12] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jaobi formulation. *J. Comp. Phys.*, 79:12–49, 1988.

[13] M. Pollefeys, R. Koch, and L. van Gool. Self-calibration and metric reconstruction inspite of varying and unknown internal camera parameters. *Intl. J. Computer Vision*, 32:7–25, 1999.

[14] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:131–147, 1990.

[15] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Computer Graphics Proceedings SIGGRAPH 1994*, pages 311–318, 1994.

[16] B.C. Vemuri and J.K. Aggarwal. 3D model construction from mulitple views using range and intensity data. In *Proceedings of IEEE Conference on CVPR*, pages 435–438, 1986.

[17] Piotr Wilinski and Kees van Overveld. Depth from motion using confidence based block matching. In *Proceedings of Image and Multidimensional Signal Processing Workshop*, pages 159–162, Alpbach, Austria, 1998.