Reconstruction of surfaces behind occlusions in range images

Freek Stulp, Fabio Dell'Acqua, Robert Fisher
Division of Informatics
University of Edinburgh
5 Forrest Hill, Edinburgh EH1 2QL
freeks@dai.ed.ac.uk

Abstract

Analysis and reconstruction of range images usually focuses on complex objects completely contained in the field of view; little attention has been devoted so far to the reconstruction of partially occluded simple-shaped wide areas like parts of a wall hidden behind furniture pieces in an indoor range image. The work in this paper is aimed at such reconstruction. First of all the range image is partitioned and surfaces are fitted to these partitions. A further step locates possibly occluded areas, while a final step determines which areas are actually occluded. The reconstruction of data occurs in this last step.

1 Introduction

Range images are used in a wide range of applications. So far they have been used extensively in object recognition [9, 12], reverse engineering [5], and other applications, nearly all focusing on small and rather complex objects and scenes. While extending the use of range images to reconstructing a whole environment rather than well-delimited objects (an important example of these applications is the CAMERA EU project [7]) new issues arose. Occlusion is a major cause of information loss: even in moderately complicated scenes it is virtually impossible or impractical to obtain complete range scans. Even for simple indoor scenes hundreds of scans might be needed to recover all of the small surfaces hidden by the occluding features in a realistic scene [11]. Even with today's scanners this is very impractical. Still, an exhaustive description of the observed object or environment is needed for some applications, like construction of a 3-D model [7]. An alternative way of filling in the gaps, at least partially, without performing extra scans, is to hypothesize the layout of objects in the occluded area by exploiting information from the surroundings. This procedure is termed reconstruction. Reconstruction of simple-shaped wide regions occluded by objects located closer to the sensor is an important point for this procedure.

This reconstruction is clearly cosmetic; however it is necessary for removing the many missing data artifacts in real scenes so as to produce visually acceptable reconstructions. Processes that do not require complete models would not need this process.

In this paper we propose a method to reconstruct surfaces behind occluding objects, such as a furniture piece in front of a wall. This is a new problem, on which little previous work [4] been done. The work presented here improves on the methods used in [4] by 1) improving the techniques used and 2) extending the method to cylindrical and spherical structures and groups of surfaces.

The key to reconstruction is to identify contiguous surface regions potentially connected behind closer occluding surfaces. Hypothetical surfaces can then be created to connect or fill in the contiguous surfaces behind the occluding surfaces.

In this paper, section 2 presents the data that was used. Section 3 discusses noise removal. The smoothed image is segmented to yield smooth continuous surfaces of the same shape, as described in section 4. Possible occlusions are detected (section 5) by locating contiguous surfaces, finding the areas between them, and hypothesising a possibly occluded surface. In the last section we demonstrate how the choice between reconstructing or not is made.

2 The data

Range information can be obtained by a variety of methods [1]. Here part of the data was sensed and registered, together with reflectance values, by a K2T 3-D laser scanner. The system, placed roughly in the middle of a room, performs a solid scan of 360 degrees azimuth, 63 degrees elevation (half above, half below the horizon). Precision of the range data is around one millimetre. Due to the huge dimensions (8000×1400 pixels) of the raw range data files (hundreds of Mb) of the K2T laser-scanner we experimented only on subsets of the original images containing

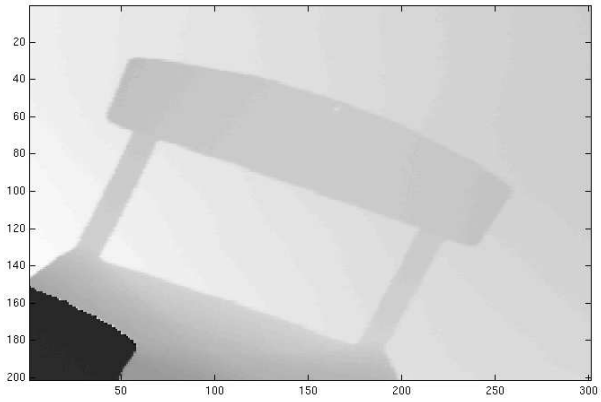

Figure 1. Range image from the K2T scanner

observed occlusions.

Another source of data was the REVERSA laser stripe ranger at the Machine Vision Laboratory at Edinburgh University. It uses structured light, and has an orthographic scanning method. Its accuracy is about 15 microns in the z-direction. The algorithm has been written to cope with both types of images.

3 Noise reduction

One of the problems with the range data acquired by the K2T laser-scanner was the high level of noise. On average, 4.62% of the pixels report a value higher than the highest surrounding one or lower than the lowest. It appears that the noise has a salt-and-pepper behaviour, but a closer inspection showed us that most of the noise occurs around depth discontinuities, and not on smooth surfaces. Although the noise in the xyz-data is non-uniform, we assume the noise measured in the range data to be Gaussian. In [8] five methods for removing Gaussian noise in three-dimensional images are compared. Nearest neighbourhood smoothing and maximum likelihood are found to perform the best. For reasons of efficiency we choose for nearest neighbourhood smoothing.

4 Image segmentation

The range data acquired by the K2T 3-D laser scanner is spherical. Unfortunately, almost all the segmentation algorithms available assume that data has been scanned orthographically [6], ignoring the geometrical distortions that are found in the r_{ij} form of a spherically scanned range image.

One algorithm for segmenting 3D datasets is [10]. We have used a slight simplification of that segmentation algorithm to do the segmentation in the xyz-domain (which is much less troubled by geometric distortions) as much as possible. Since we are only interested in reconstructing large surfaces, the segmentation algorithm can discard smaller surfaces, processing only the larger ones. We by no means claim this segmentation algorithm to be perfect, but it is sufficient for our purposes.

Transforming the 2.5D r_{ij} form into the 3D xyz form uses the standard formulas for transformation described in for instance [2]. The result of this transformation is a partly unordered 3D point cloud, shown in figure 2. Note that the very close object seen in lower left-hand corner of figure 1 has been clipped off this image. Of course it is not discarded during computation. Because the xyz-data is unordered we can no longer use local windows, as is customary in image analysis. Instead of using the window as an estimator of proximity we use the actual geometric distance as an exact measure, defining points that are within a certain geometric distance from each other as *neighbours*.

The neigbour-finding method takes one parameter: the desired mean number of neighbours per point. A distance threshold is computed such that the mean number of neighbours per point is as specified. For reasons of efficiency we only randomly sample 10% of the total number of points to determine this threshold. The number of neighbours depends on the number of points within this distance threshold. A typical mean number of neighbours per point would be 9.0. The actual number of neighbours then varies between 0 and 25. Every point keeps a list of its own neighbours, so computing them only has to be done once.

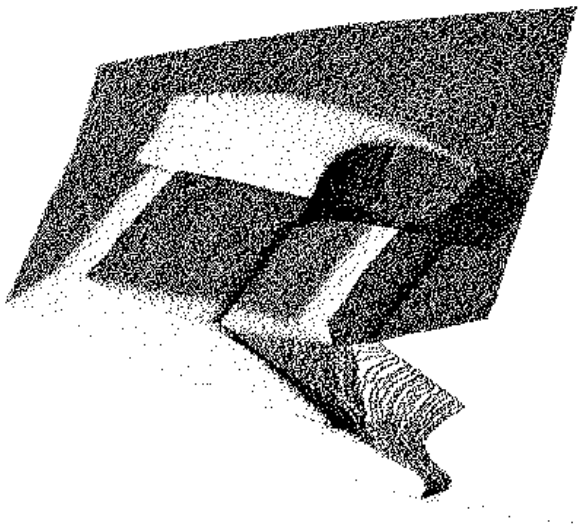

Figure 2. The 3D point cloud

Our segmentation algorithm aims at finding large areas of the same shape. Therefore we will do clustering of neighbouring 3D points within depth discontinuities that are sim-

ilar in normal and curvature, yielding continuous surfaces of roughly constant shape class.

4.1 First segmentation

Besides using the neighbour-relationship instead of windows, our segmentation proceeds in much the same fashion as previous segmentation algorithms such as described in [6]. We do a local surface estimation at all the points, enabling the computation of Gaussian and mean curvature, as well as the surface normal.

The local surface fitting is based on a local least squares surface model using discrete orthogonal polynomials. This method has been used successfully for curvature estimation in range image segmentation (*e.g.* [2]).

After these surface characteristics have been computed, we will do a first segmentation, using a simple region growing algorithm based on the local surface characteristics. Points are considered to belong to the same region if they meet the following requirements:

Must be neighbours: This clusters points that are close to one another. It will locate depth discontinuities. Discarding points with less than 2 neighbours (as defined above) also removes a lot of noise around the edges.

Must have same curvature sign: This allows us to separate regions that are of a different shape. This will give us an idea about what kind of surface we should fit to the region.

Angle between normals $<$ 5 degrees: The dot-product between the two normals must be smaller than 5 degrees. This locates fold-edges.

Furthermore, regions must be larger than a certain threshold. We have chosen a minimum number of 200 points. In these cases we feel that there is not enough information available from the surroundings to justify a reconstruction.

The segmentation of figure 2 results in the regions shown in figure 3. For reasons of clarity we have shown this result in the r_{ij}, although segmentation actually takes place in the xyz-domain.

4.2 Surface fitting

To acquire good comparative measures between the different clouds of xyz-points we have found in the segmentation, we will fit surfaces to them. The geometric parameters will be compared to yield continuity between surfaces. Inspection of numerous images showed that planes, cylinders and spheres are usually sufficient to describe most of the surfaces in the image, especially where buildings are concerned. In the following table the geometrical description of the surfaces is shown.

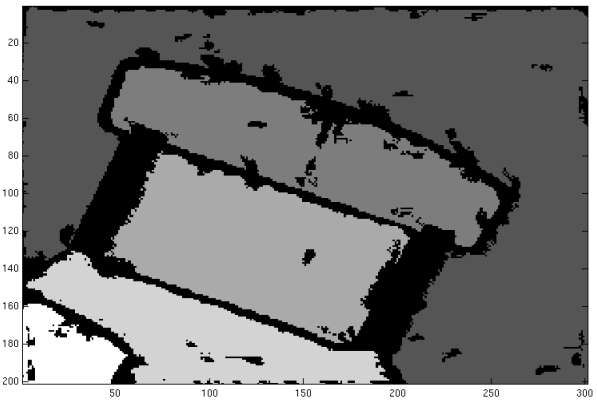

Figure 3. The first segmentation.

Surface type	Description
Planar	surface normal $surf\_nl$ displacement $disp$
Cylindrical (circular)	point on axis p unit vector of axis $axis\_uv$ radius r
Spherical	centre c radius r

5 Finding Possibly Occluded Areas

Now that we have segmented the image into large surfaces, we compute if any surfaces are occluding other surfaces. First contiguous, but unconnected, surfaces are matched and grouped. We then determine the areas lying between the contiguous surfaces. If the intermediate regions are closer to the sensor, then this indicates the possible existence of occluded surface that connects the contiguous surfaces. Assessment of this possibility is the main topic of this section.

5.1 Matching surfaces

For two regions to be deemed possibly contiguous, they must first be of the same type. We use the geometric properties of the surfaces described in the previous section for comparison. For planes we consider the angle between the surface normals and the displacement between the two planes. For cylinders we determine the vector pointing from the first axial point to the second one (called $p_1p_2\_uv$ in the table). This vector, as well as the other two describing the axis of the cylinder must all be approximately colinear to each other. Also, the radii are compared. For spheres the distance between two centres ($dist(c_1, c_2)$ in the table) must not be too large. Again the radii must not differ too much. The table below gives the parameters and thresholds

used for determining whether two surfaces match or not.

Surface type	Requirements for matching
Plane	$surf\_nl_1 \cdot surf\_nl_2 < 5\deg$ $\lvert disp_1 - disp_2 \rvert < 5cm$
Cylinder	$axis\_uv_1 \cdot axis\_uv_2 < 5\deg$ $axis\_uv_2 \cdot p_1p_2\_uv < 5\deg$ $p_1p_2\_uv \cdot axis\_uv_1 < 5\deg$ $(2 * \lvert r_1 - r_2 \rvert)/(r_1 + r_2) < 5\%$
Sphere	$(2 * dist(c_1, c_2))/(r_1 + r_2) < 10\%$ $(2 * \lvert r_1 - r_2 \rvert)/(r_1 + r_2) < 5\%$

Within an image all the possible surface pairs are compared and organised into groups. Groups are such that each surface within a group matches with each other surface within the same group, according to the definitions given above. Two non-matching surfaces can never be in the same group. Suppose surfaces A&B and B&C match, but A&C do not. In this case we will make two groups (A&B) and (B&C), and not one large group (A&B&C). This differs from the approach in [4].

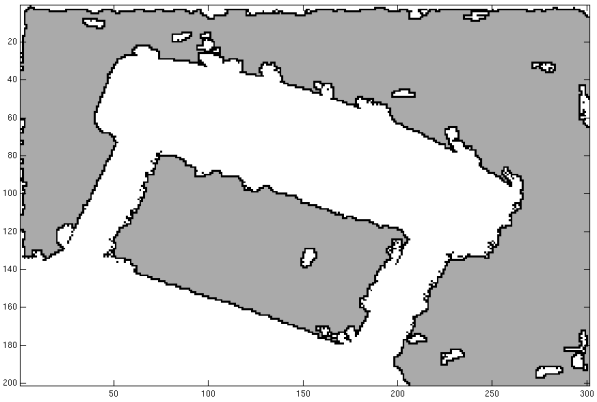

Figure 4. Matching areas. These belong to the same wall lying behind the chair. Areas are gray, perimeter points are shown in black.

5.2 Area between two surfaces

Up till now we have been working in the xyz domain, for reasons given in section 4. Because we only want to reconstruct points at positions that could actually have been sensed by the sensor at a certain element (i,j) in the range image, it is necessary to go back to the r_{ij} domain. Note that this means a significant change in the way that distance and proximity should be interpreted! Distance now means the two dimensional distance between two pixels in the r_{ij} image in pixels. If we want to refer to geometric distance we will do so explicitly.

Before reconstruction takes place, we have to determine at which pixels in the r_{ij} image reconstruction would be justified. A first guess is to take the areas lying between matching surfaces of the same group. As we shall see later this is not enough, but for now it will do.

To avoid reconstructing between surfaces that lie very far apart, we will also place a distance constraint. A certain point may only be reconstructed if it lies within a certain threshold distance from both matching surfaces. This threshold varies between surfaces and is implicitly encoded in the expansion size of the distance transform, which will be discussed later in this section. The main idea is that the larger the area of the surface is, and the better the fit of the surface to the patch of points it represents, the more certain we can be about reconstructing points in its neighbourhood. This implies that we can allow reconstruction of points that are further away from such a surface than a surface with a small area or a bad surface fit.

There are two straightforward ways of acquiring the pixels between two regions. The first is to take the convex hull containing the two regions. The pixels lying in this hull (minus the two regions themselves of course) will approximately yield the points lying between the two regions. Unfortunately, it yields no information about the distance of the pixels to both matching surfaces needed for applying the distance constraint. Another method [4] is to connect all the perimeter points of the matching regions with lines. Only lines that are shorter than the distance threshold and that do not cross the matching regions themselves are accepted. The Bresenham algorithm [3] can then be used to determine the area lying between the two regions. This algorithm is robust, but computationally very expensive.

We propose a method that combines the robustness and efficiency of the two. The greatest advantage is that information about the distance from every point to both regions is stored, allowing an easy computation of the hypothesised occluded surface, as we will see in section 6.1. The idea is to find perimeter points that are actually facing the other area, as the convex hull method does. After that we use the approach of drawing lines between the pixels facing each other with the Bresenham algorithm, as in [4]. Points facing the other area are defined as follows: A line between the point of interest and the closest point of the other area should not be longer that the distance threshold, and may not cross any other area besides the occluding surface.

The method relies on *distance transforms*: If the input is a binary image, in which the 1's represent a certain area, the pixels in a 2D distance transform will have a value that encodes the distance (in pixels) to the nearest region point. Points in the regions themselves thus have value 0. In our enhanced distance transform, every element also encodes at

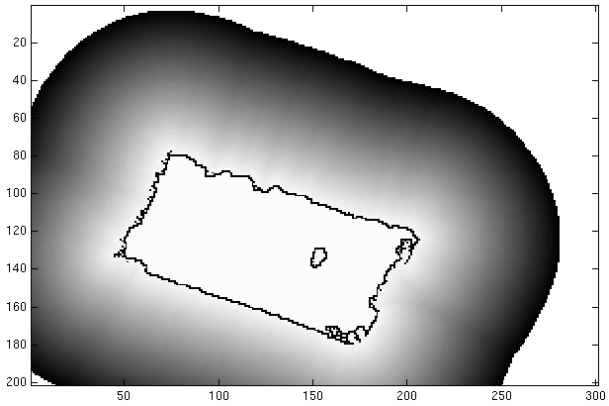

Figure 5. The distance transform for one of the areas shown in figure 4. Perimeter points have been added for clarity.

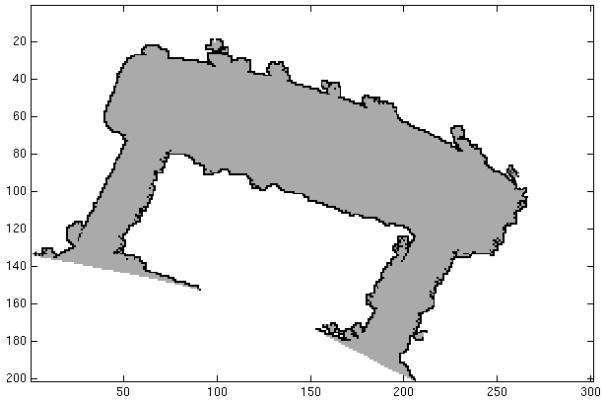

Figure 6. The potentially reconstructible area between the two surfaces shown in figure 4.

which location the nearest 1 can be found. Figure 5 shows the distance transform of one of the matching areas in figure 4.

Finding the area between two surfaces can now be done as follows:

1. Compute distance transforms for both surfaces. The number of points it represents determine how far the distance transform may *expand*. The expansion is twice the square root of the number of points. More points means that more information can be used to reconstruct surfaces, which means that we can reconstruct further away from the surface. An absolute maximum has been set to 150.0 pixels.

2. Find all the points whose difference in values in the distance transform is within a small threshold (≤ 5 pixels). These equidistant points have approximately the same distance to both surfaces.

3. For all the equidistant points, determine which perimeter points fall within the distance transform range at that equidistant point. This locates 5-10 perimeter points from both surfaces which face that equidistant point.

4. Connect all the perimeter points facing an equidistant point to the matched perimeter points in the other area, but only if the distance between them is smaller than the distance threshold. Fill in the points on all the connecting lines using the Bresenham algorithm.

The final result for the matching areas in figure 4 can be seen in figure 6. The black pixels are the perimeter points that are facing each other. Note that this requirement excludes a lot of the perimeter points shown in figure 4. The area that might be occluded is gray.

6 Actual reconstruction

The possibly occluded area has been determined. We still say possibly, because another requirement must be met before we can be certain occlusion is taking place. It may very well be that the actually observed area between the two matching surfaces is further away from the sensor than the surface to be reconstructed. This happens on many occasions when niches such as doors or windows are involved. The difference between a niche and an occlusion is shown in figure 7.

Before we can determine whether we are dealing with a niche or an occlusion we will have to compare the measured points with the points of the reconstructed surface along the corresponding line of sight. This means we will have to reconstruct the surface anyhow for comparison. If it lies behind the measured points (occlusion), we replace these points with the reconstructed ones. If it lies in front of the measured points (niche) we will discard the reconstructed points and leave the situation unaltered. Because the reconstruction requires hypothesising non-observable data, we will be very conservative, and only reconstruct areas that have a high likelihood of being correct.

6.1 Surface hypothesis: intersections

Given an occluding pixel and an occluded surface, a simple and intuitive way to perform reconstruction is to intersect the ray from the sensor to the occluding point with the

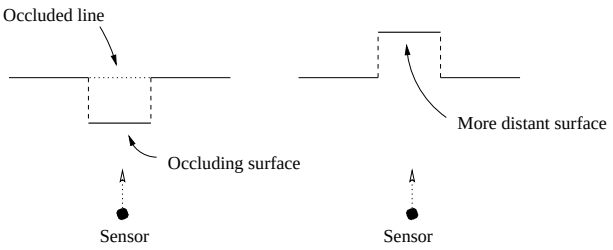

Figure 7. A *real* occlusion and a niche

occluded surface. As this ray overlaps the optical ray of the laser scanning beam, the reconstructed pixel is placed in a position that could actually have been sensed by the sensor.

In the unlikely event that no intersection is found, which happens if the line is parallel to the plane, or if the line passes the cylinder or sphere, no reconstruction takes place for that point.

For planes there is usually one intersection. There is the exception of a line parallel to the plane. Because this would require the sensor to lie in the plane as well, it is impossible since the sensor always has a certain volume.

For cylinders and spheres we usually find two intersections. There are certain exceptions such as a line tangent to the cylinder or sphere. Considering that we are working with floating point accuracy, the perfect alignment needed for the tangent to a cylinder or sphere is also unlikely to happen. Because we compute two intersections, it is necessary to choose between them. First of all, priority is always given to the point that would yield a positive range. A negative range would arise if the sensor is inside the cylinder, causing one intersection to lie behind the sensor. We clearly want the point that lies in the direction the sensor is facing. If they both lie in this direction, the shape of the surface is used. If it is the concave interior of a trough or bowl, choose the more distant intersection. Similarly choose the closer intersection on convex surfaces.

6.2 Interpolation between intersections

As expected due to noise, when the two (or more) surfaces are extrapolated into the possibly occluded area, they never perfectly match each other. Instead of choosing to use the intersection with only one of the surfaces, we will interpolate between them. The chosen solution is a weighted averaging between all the intersections with the extrapolated surfaces, with weights depending on the distances from the intersection to the closest perimeter point of each surface. In other words, given a pixel to be reconstructed, two or more intersections are computed as explained above, one per each candidate surface. The position of the reconstructed pixel is computed as follows:

$$\vec{x}_{inter} = \frac{\sum(f(d_s)\ \vec{x}_s)}{\sum f(d_s)} \tag{1}$$

$$f(d_s) = (d_{max} - d_s)^{\frac{3}{2}} \tag{2}$$

The summation in equation (1) takes place over all the surfaces involved in the reconstruction; that is, all the transitively matching surfaces combined in a group. $\vec{x}_s$ is the intersection with the s-th plane, d_s is the distance to the closest actually observed point in surface s, and $f(d_s)$ a weighting function. As discussed in section 5.2 the size of expansion of the distance transform depends on the number of points size in the surface. All the surfaces usually have different sizes, causing their maximum expansion sizes to differ. d_{max} is the maximum of the maximum expansion sizes of all the surfaces taking place in the summation. Note that d_s will never exceed d_{max}.

Computing the distance of all the points $\vec{x}_s$ to the closest point in all surfaces s would be computationally expensive problem, were it not that we have already computed the distance fields of all the surfaces involved. Because the distance field of a region specifies the distance of a point at (i,j) to the nearest point of that region, the problem is a simple table look-up!

Because points that are closer to a certain surface should be weighed heavier than points that are further away, the weighting function should be decreasing with the input distance. The weighting function we chose is shown in equation (2).

6.3 Incorporating perimeter points

Interpolating between the different surfaces has proven to be insufficient, because discontinuities around the edges arise when the surface is not a perfect fit at these edges. We solve this problem by again interpolating, but this time only between the closest perimeter point of the closest surface patch and the point found through interpolation of the intersections. Remember that every pixel in the extended distance transform not only contains information about the distance to the closest perimeter point, but also exactly which perimeter point this is. This allows us to find the perimeter point we should use for interpolation easily.

This interpolation is based on a logarithmic decay function. The influence of the perimeter point on the reconstruted point decays as the point to be reconstructed lies further away from the known data. The decay function we have chosen kan be seen in equation 4. At the transition from measured data to reconstructed data the perimeter point completely overrides the influence of the intersection, guaranteeing a smooth transition into the measured data. Given equations 3 and 4 the influence of the perimeter point

at distances 0, 5, 10 and 20 pixels is 100%,53%,29% and 8% respectively.

$$\vec{x}_{recon} = g(d_s)\ \vec{x}_{perim} + (1 - g(d_s))\ \vec{x}_{inter} \quad (3)$$

$$g(d_s) = e^{(-x/8)} \quad (4)$$

$\vec{x}_{recon}$ is the final hypothesised reconstructed point. $\vec{x}_{perim}$ is the closest perimeter point of the closest surface. $\vec{x}_{inter}$ is the point found in equation 1. $g(d_s)$ is the weighting function discussed in the previous paragraph.

This procedure is repeated for all the points lying between the regions of the matching group. In the end we have a reconstructed surface that can be compared with the original image.

6.4 Voting for reconstruction

We are now faced with the choice of whether to reconstruct or not. Reconstruction is a severe change to the image, so we want to be very careful in applying it.

We consider all the pixels in the area between the surfaces in a certain group of matching surfaces. For each pixel, we determine if it is further away from the sensor than the original range measured at that pixel. If this is the case it is worth reconstructing; if it is not the case, it belongs to a niche and should not be reconstructed. Considering reconstruction of pixels individually is of course not very robust. For this reason we introduce a voting system.

Basically, a pixel votes for or against reconstruction, depending on the issues described above. The area between the surfaces itself might contain several other surfaces, which do not belong to the group of surfaces taking part in its reconstruction. The votes of the pixels are polled per surface. If enough pixels are in favour of this reconstruction, it is reconstructed, otherwise it isn't. A high percentage of 90% was chosen to ensure that reconstruction was justified.

7 Results

Figure 8 shows some reconstructed images. They are shown in both r_{ij}- and xyz-form. Since the reconstruction is mainly a cosmetic improvement, and the reconstructed surfaces appeared unnaturally smooth, we added the same amount of Gaussian noise in the z-direction as found in the surfaces on which its reconstruction was based.

The left image is the image we have been processing throughout the article. The right one is an image acquired by the orthographical scanner. It shows the reconstruction of two cylinders occluded behind a plane.

The computation times for the C++ program running on a 440Mhz Sun workstation were 50 and 30 seconds for the segmentation of these two images. This is mainly due to the accurate surface fitting. The actual reconstruction algorithm took 29 and 17 seconds respectively.

8 Conclusion and future work

A method for analysing range images, locating occlusions of large homogeneous surfaces and reconstructing these surfaces behind occluding objects has been presented. Some results obtained from the research have also been shown.

Future work will be aimed at reconstructing the intensity texture on the reconstructed surface. In the present we have simply added noise, but the rendering would be visually enhanced if the 3D texture, as well as patterns in the intensity image, were modelled and projected on the reconstructed surface.

Also, using knowledge about the world instead of the low-level representation such as planes and cylinders might prove useful in reconstruction. The reliability in reconstructing a wall behind a chair would be much enhanced if it was actually known that it was indeed a wall. Unfortunately, we will have to await advances in other areas of research before it becomes viable in this context.

A problem is that the images used are subimages of larger scenes. The performance of this algorithm on large scenes needs to be tested more extensively, but it is to be expected that more reconstruction errors will be made in more complex scenes. On the other hand, this is research in progress, so the algorithm is improving as well!

One could also develop a reliability gauge for the reconstructed pixel, as not all the pixels can be reconstructed with the same confidence, and this should be taken into account when using reconstruction results for further processing, e.g. triangulation. The distance to the closest perimeter point has been used as an approximation of this measure, but we would like to implement a more statistically-founded measure.

One might argue that the solution to the occlusion problem is simply to acquire additional images. Recent results by Sanchiz [11] show that even simple scenes can require hundreds of images to obtain complete, high quality range data. Thus, it may be preferable to reconstruct small missing regions instead of attempting to observe them.

An interesting idea is to merge the two systems. Parts of the image that can be reconstructed with high certainty will not have to be scanned from another angle. When areas of the image cannot be reconstructed, or only with low certainty, Sanchiz's next-best-view algorithm can compute how to obtain information about this area, after which more reconstruction can take place. This *scan-reconstruct-scan* cycle may drastically reduce the number of scans needed to obtain a complete description of the scene.

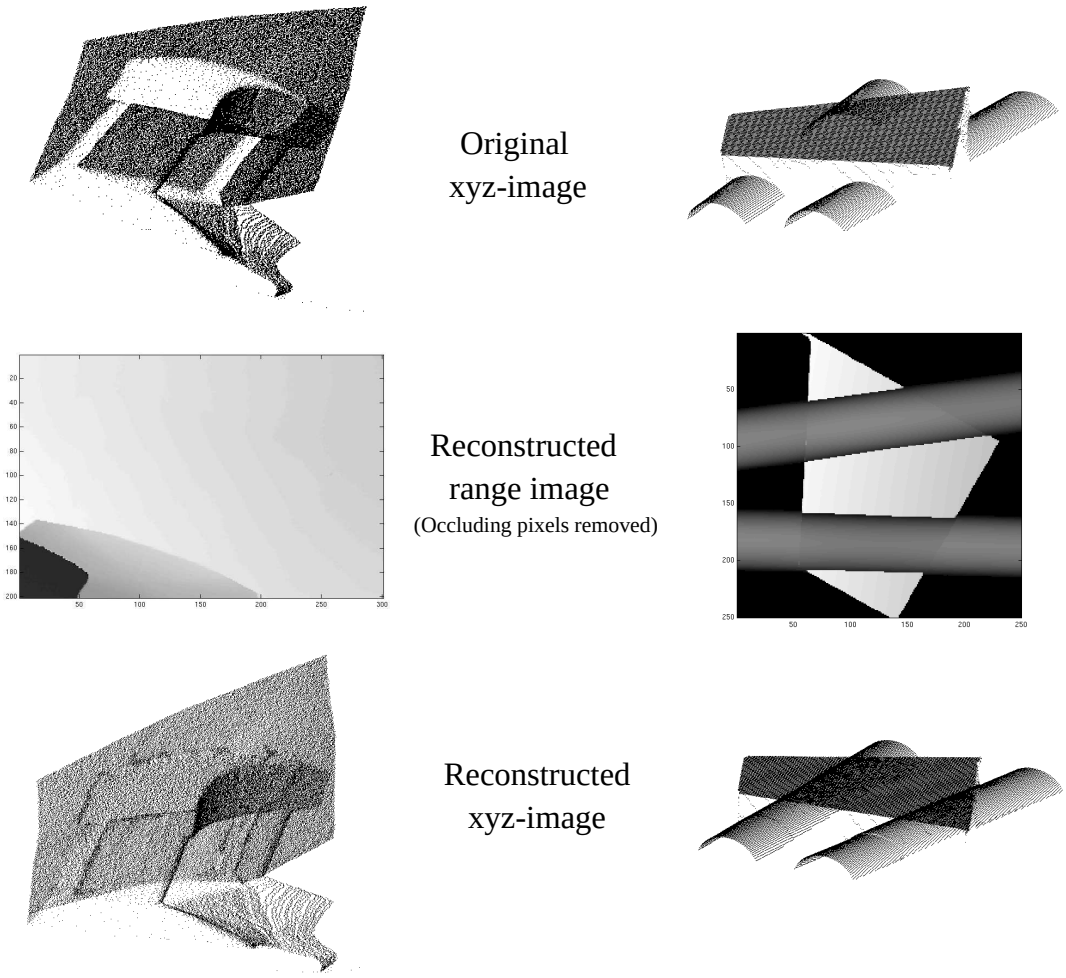

Figure 8. Results for an occluded wall (left) and two occluded cylinders (right)

Acknowledgements

This research was supported by the EC TMR network CAMERA (ERB FMRX-CT97-0127).

References

[1] P. J. Besl. Active, optical imaging sensors. *Machine Vision and Applications*, pages 127–152, 1988.

[2] P. J. Besl. *Surfaces in range image understanding*. Springer-Verlag, 1988.

[3] J. Bresenham. Incremental line compaction. *The Computer Journal*, 1(25):116–120, 1982.

[4] F. Dell'Acqua and R. Fisher. Reconstruction of planar surfaces behind occlusions in range images. University of Edinburgh, 2000.

[5] R. Fisher D.W. Eggert, A.W. Fitzgibbon. Simultaneous registration of multiple range views for use in reverse engineering of cad models. *Computer Vision and Image Understanding*, 69(3):253–272, 1998.

[6] A. Hoover, G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, R. Fisher An experimental comparison of range segmentation algorithms. *IEEE Trans. Pat. Anal. and Mach. Intel.*, 7(18):673–689, 1996.

[7] R. Fisher. http://www.dai.ed.ac.uk/daidb/people/homes/rbf/camera/camera.htm.

[8] S. Hurt and A. Rosenfeld. Noise reduction in three-dimensional digital images. *Pattern Recognition*, 17(4):407–421, 1984.

[9] P.G. Mulgahonkar, C.K. Cowan, J DeCurtins Understanding object configurations using range images. *IEEE Trans. Pattern Anal. and Mach. Intel.*, 14(2):303–307, 1992.

[10] R. B. Fisher, A. W. Fitzgibbon, D. Eggert Extracting surface patches from complete range descriptions. In *Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, pages pp 148–155, May 1997.

[11] J. Sanchiz and R. Fisher. Environment recovery by range scanning with a next-best-view algorithm. University of Edinburgh, to appear in Robotica, 2000.

[12] R.B. Fisher, A.W. Fitzgibbon, M. Waite, M. Orr, E. Trucco Recognition of complex 3-d objects from range data. In *Proc. CIAP93*, pages 509–606, 1993.