Exploiting Passthrough Information for Multi-view Object Reconstruction with Sparse and Noisy Laser Data

Martin Herrmann
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA - 15213

Siddhartha S. Srinivasa
Intel Labs Pittsburgh
4720 Forbes Avenue, Suite 410
Pittsburgh, PA - 15213

February 12, 2010

CMU-RI-TR-10-07

Abstract

We describe a probabilistic model for utilizing passthrough information for producing 3D geometry models from a rotating laser scanner. Our method is fast, performs particularly well with relatively sparse data, is robust to noise of the depth data and naturally handles grazing points. We demonstrate our results on the HERB platform where the robot automatically builds a watertight 3D model of an object it is handed.

This work was partially supported by the National Science Foundation under Grant No. EEC-0540865.

1 Introduction

Household robots such as [12] have to operate in unknown environments. For such a task, perception is of vital importance. In order to grasp objects, the robot needs to build geometry models of both the environment and the objects.

In addition to cameras, a common sensor for inspecting the environment is a spinning or tilting laser scanner. A planar laser scanner measures the distance to the closest surface in discrete directions in a single plane; by rotating the scanning plane, we get the distance to the closest surface in three dimensions.

The depth data from the laser scanner is inherently noisy. In addition to the noise of the measurement itself, the reflectivity of the surface as well as other surface properties affect the measured depth. High quality laser scanners are available, but they are expensive. For a household robot, we are interested in small and inexpensive laser scanners. These can have a scanning inaccuracy of as much as 50 mm[3]. Our method is robust to large amounts of arbitrarily distributed noise.

We implemented a scenario where HERB is either handed a novel object to model (Fig. 1) or encounters the object on a table. HERB examines the object from multiple view that are registered into a common reference frame. In the case where HERB is holding the object, we use the arm's joint encoders and forward kinematics for registration. In the case where HERB encounters the object on a table, we use markers on the table for registration. Once views are registered, our algorithm builds watertight 3D models using the pipeline shown in Fig. 3. Exploiting the geometry of the laser by using passthrough information enables us to produce models that are far more accurate than those produced by merely treating the output of the laser as a point cloud, as seen in Fig. 2.

2 Related Work

The reconstruction of watertight 3D models from perceptual data has been extensively studied. Reconstruction from multiple images has been achieved using visual[8] and photo hulls[7]. Numerous methods have also been developed to produce very high quality surface- and volume-based models from laser data ([13, 4, 10, 6]).

Reconstruction of 3D geometry from visual sensors is a common task in computer vision. Laurentini[8] introduced the concept of *visual hulls* which reconstruct objects from image silhouettes. Kutulakos and Seitz[7] create *photo hulls* from multiple images using color consistency.

From a laser scanner, we get a point cloud describing an object. Numerous methods have been developed to create surface or volume based models from point cloud data. Luo and Tseng[13] reconstructs surfaces from LiDAR point clouds. Hoppe[4] produces smooth reconstructions from unorganized point clouds. Mederos et al.[10] and Kolluri et al.[6] reconstruct surfaces from noisy point clouds.

Some of these methods produce very high quality reconstructions, but many of them require dense point clouds and long computation times. An example given in [6] uses two million points and takes 437 minutes to compute. In an interactive application for a household robot, we can tolerate a slightly lower quality model, but we want the process to take less than one minute, including acquisition of the data.

Our method has the following advantages:

- it is fast to compute
- it is robust against noise of the depth measurement

Figure 1: HERB[12] examining a novel object with its spinning laser and camera

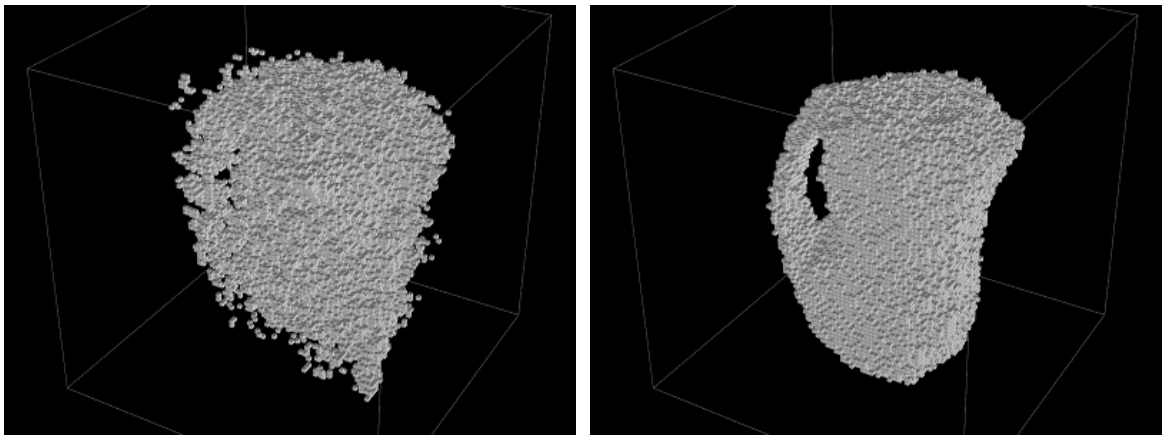
- it performs well with relatively sparse point clouds, thus allowing rapid data acquisition
- it handles grazing points naturally, that is, without explicitly having to deal with them

3 Using Passthrough Information

A laser scanner measures the distances $\tilde{d}_i = \tilde{d}(\theta_i, \varphi_i)$ to the closest surface in discrete directions (θ_i, φ_i). With the pose of the laser scanner, we can compute the coordinates of points $\underline{p}_i$ in space that lie on the surface of an object. A set of such points $\{\underline{p}_i\}, i \in \mathbb{N}$ is called a *point cloud*.

However, in doing so, we are discarding the information that the surface is the closest surface to the laser scanner - i. e., that the space between the laser scanner and the surface is unoccupied. This passthrough data is actually more useful than the hit data because the latter only provides information about one single point in space, whereas the former provides information about all points on a line segment between the laser scanner and the hit point.

The difference between hit information and passthrough information is illustrated in figure 4. Figure 4(a) shows a one-dimensional voxel grid with a single hit under low noise conditions. Nothing is known about the grey voxels. The second voxel grid has the same hit information, but with a high



(a) Using only hit information

(b) Our method, using passthrough information

Figure 2: Our method improves the reconstruction of voxel grids from noisy laser data. This example shows the traditional method (a) and our method (b) under moderate noise conditions ($\sigma = 5$ cm).

amount of noise. The graph shows quantitatively the confidence that the voxel at a given position is occupied. With noise, the confidence declines because the probability for the actual location of the surface is dispersed over more voxels.

Figure 4(b) shows the passthroughs instead of the hits for the same voxel grid. In the case of low noise, all of the red voxels' state is known. Again, nothing is known about the grey voxels (which are behind the surface). With noise, the confidence about voxels close to the hit declines, as with the hit information, but the information about voxels close to the laser remains useful.

In order to utilize this information, we need additional knowledge about the scan geometry. In particular, for each point $\underline{p}_i$, we need to know the location of a point in space where the corresponding ray came from. We call this point the *origin* $\underline{o}_i$ of the ray[1]. A set of pairs of surface points and corresponding origins, $\{(\underline{p}_i, \underline{o}_i)\}, i \in \mathbb{N}$, is called a *ray cloud*.

3.1 Grazing points

Sometimes, a ray – which has a non-zero thickness – partially hits a surface. One part of the ray is reflected right away and the other part continues until it hits the next surface. The distance measured by this ray will typically assume some value between the distances of the two surfaces that reflected the parts of the ray. A point derived from such a measurement is an outlier called a *grazing point*. A grazing point typically does not lie on any surface.

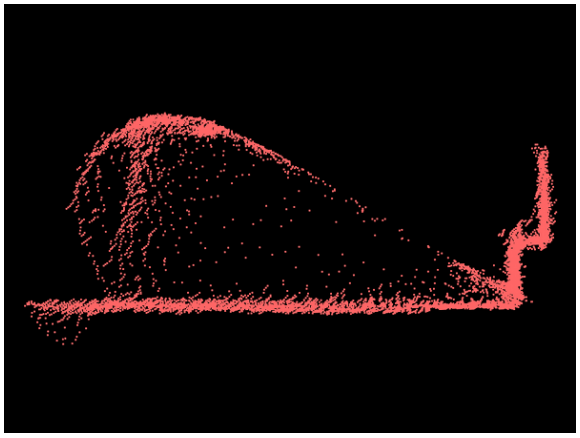
Grazing points can not always be easily identified, particularly in the following cases:

- the grazing points lie near the center of the scanned area, where the point cloud is very dense
- the scene contains structures whose size is in the order of magnitude of the noise

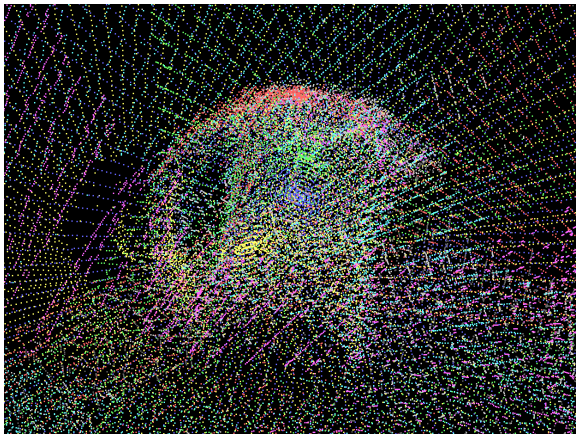
An example of the first case is shown in Fig. 3(b). Note that the grazing points along the top side of the object are denser than the actual surface points on the side of the object.

[1] note that the origin of a ray is not necessarily the point where the ray was emitted, nor is the origin uniquely defined

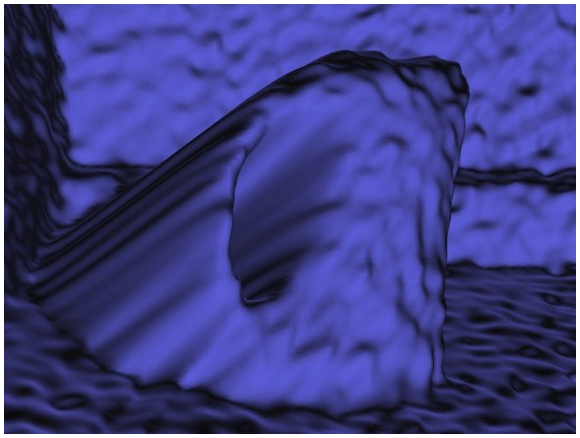
(a) The object (kettle) standing on a table



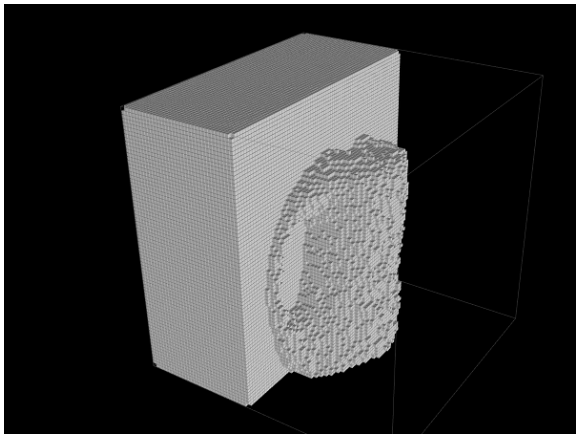
(b) The image shows the kettle standing on a table (with the handle to the left). The location of the laser scanner is to the upper left.



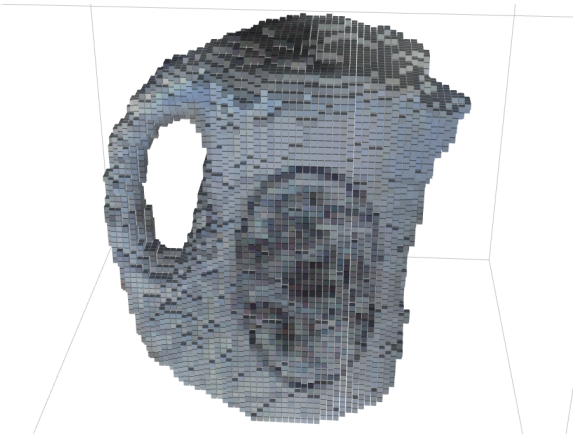
(c) The raw point clouds from seven viewpoints



(d) A depth image. The location of the laser scanner is to the upper right, behind the viewer;

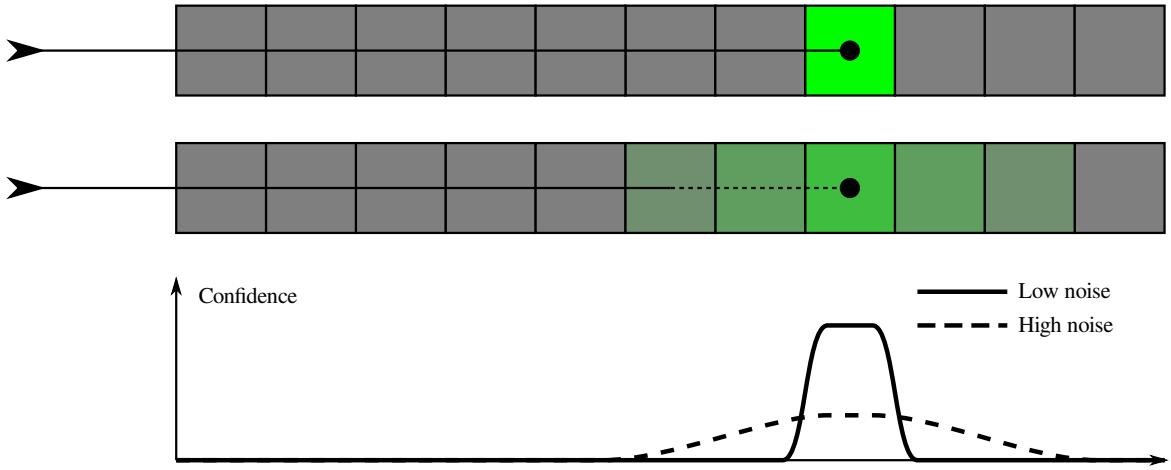


(e) An intermediate state during voxel carving

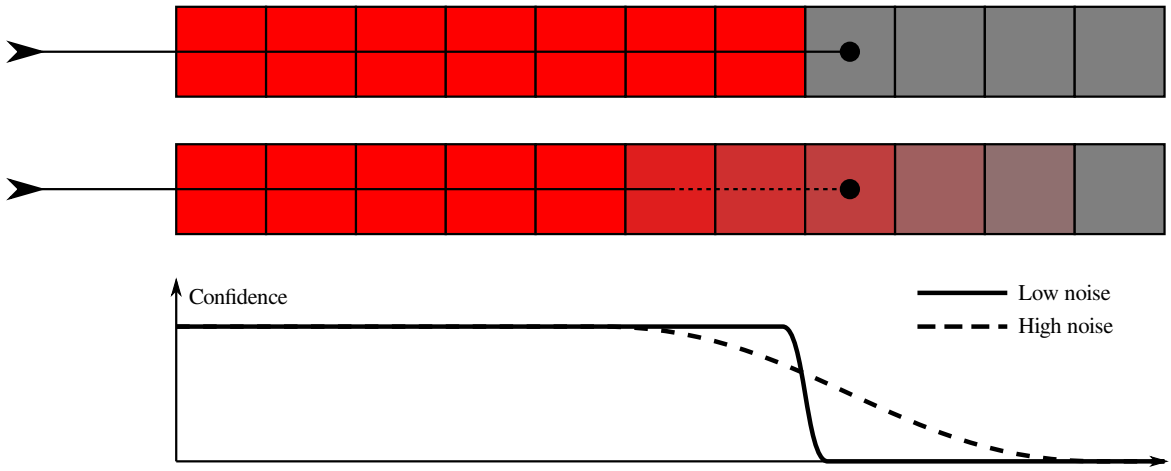


(f) The voxel grid generated from seven depth images

Figure 3: An overview of our method



(a) The usefulness of the hit information declines with noise



(b) Most of the passthrough information remains useful with noise

Figure 4: The relative confidence of hit and passthrough information under low and high noise conditions

3.2 Passthrough information in the presence of noise

The measurements from the laser scanner are noisy. The measured depth $\tilde{d_i}$ is given by

$$\tilde{d_i} = \left|\left| \underline{p}_i - \underline{o}_i \right|\right| = d_i + n_i$$

where d_i is the true depth and the random variable n_i denotes the noise. With increasing noise, the hit information becomes meaningless, while a large part of the passthrough information remains useful.

The probability density of the measured depth is

$$f_{\tilde{d_i}}(x) = P(\tilde{d_i} = x) = f_{n_i}(x - d_i)$$

where f_{n_i} is the probability density of the noise.

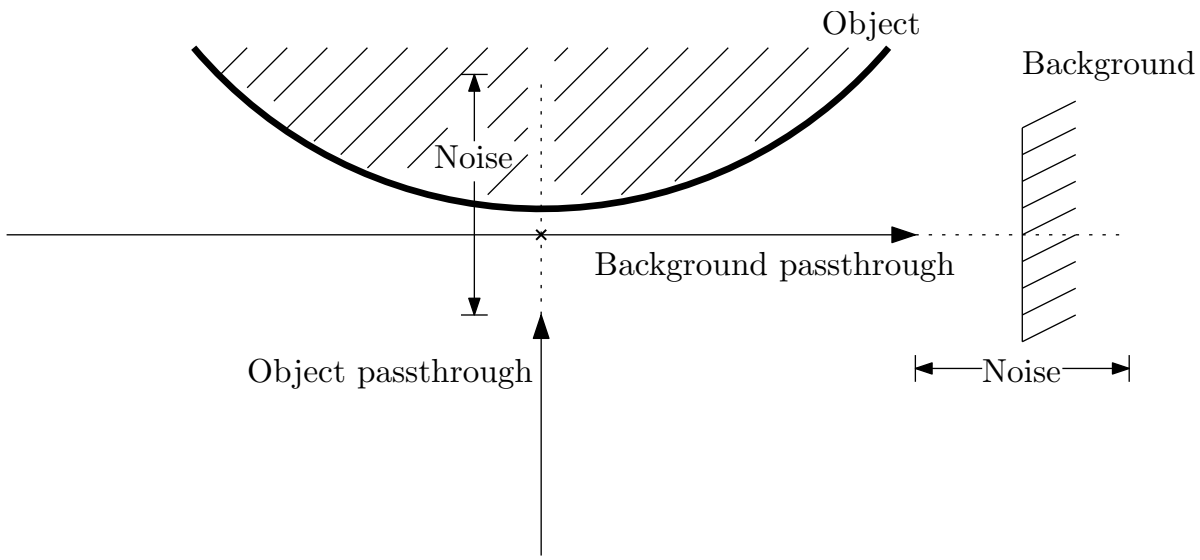


Figure 5: With noise, the hit information becomes meaningless for evaluating the occupancy of the marked point. The passthrough information from the ray passing by the object remains useful.

When evaluating points on a ray $(\underline{p}_i, \underline{o}_i)$, we assign to each point $\underline{p}$ one of the labels "free" or "unknown", based on the distance of the point to the origin, $\left|\left|\underline{p}-\underline{o}_i\right|\right|$:

$$\text{label}(\underline{p}) = \begin{cases} \text{free} & \text{if } \left|\left|\underline{p}-\underline{o}_i\right|\right| < t_i \\ \text{unknown} & \text{else} \end{cases} \tag{1}$$

with some threshold t_i. All points on the ray which are closer to the origin than the threshold are classified as "free". We choose the threshold by subtracting a *margin* Δd from the measured depth:

$$t_i = \tilde{d}_i - \Delta d$$

We generally want to use a low value for Δd because otherwise, cavities will look flatter than they actually are. Cavities with a depth less than Δd cannot be reconstructed at all.

Using a noise model of the laser scanner, we can determine a suitable value for the margin by considering the probability that there are points which are misclassified as "free", that is

$$\begin{aligned} \exists \underline{p} \in \left[\underline{o}_i, \underline{p}_i\right) : (\left|\left|\underline{p}-\underline{o}_i\right|\right| > d_i) \wedge (\left|\left|\underline{p}-\underline{o}_i\right|\right| < t_i) \\ \Leftrightarrow d_i < \tilde{d}_i - \Delta d \end{aligned}$$

The probability for a misclassification is given by

$$\begin{aligned} P_{\text{mis}} = P(d_i < \tilde{d}_i - \Delta d) &= P(n_i > \Delta d) \\ &= 1 - \int_{-\infty}^{\Delta d} f_n(x)\mathrm{d}x \\ &= 1 - F_n(\Delta d) \end{aligned}$$

where F_n is the probability distribution of the noise. Given a maximum acceptable misclassification probability $P_{\text{mis,max}}$, we can determine the smallest value of Δd that results in $P_{\text{mis}} \leq P_{\text{mis,max}}$.

For example, in the case of no noise ($f_n(x) = \delta(x)$), choosing $\Delta d = 0$ results in a misclassification probability of $P_{\text{mis}} = 0$. For gaussian noise with mean μ and standard deviation σ ($f_n(x) = \frac{1}{\sigma}\varphi(\frac{x-\mu}{\sigma})$), we get

$$\begin{aligned} P_{\text{mis}} &= 1 - \int_{-\infty}^{\Delta d} \frac{1}{\sigma}\varphi\left(\frac{x-\mu}{\sigma}\right) \mathrm{d}x \\ &= 1 - \Phi\left(\frac{\Delta d - \mu}{\sigma}\right) \\ \implies \Delta d &= \Phi^{-1}(1 - P_{\text{mis,max}}) \cdot \sigma + \mu \end{aligned} \tag{2}$$

Thus, for bigger variance of the noise, we need a bigger margin in order to achieve the same probability of misclassification. Note that we don't require gaussian noise; any noise that can be expressed in terms of a probability distribution can be handled.

3.3 Voxel carving with passthrough information

We use a volume based approach for geometry reconstruction. We divide the space into a uniform grid of cube shaped *voxels*. For each ray cloud, we rasterize all of the rays using an algorithm such as [2] and mark voxels whose center $\underline{c}_v$ was classified as free according to 1, for at least one ray of the ray cloud.

Voxels near the surface of the object can be determined as free in two different ways, as shown in Fig. 7: either by a ray directly hitting the surface near the voxel (*object passthroughs*), or by a ray passing by the object and hitting the background (*background passthroughs*). With noise, the usefulness of object passthroughs diminishes, while the background passthroughs remain useful.

With enough scans, an unoccupied voxel will generally have several passthroughs. It is useful to require a minimum number of passthroughs in order for a voxel to be carved. This allows us to choose a lower margin because we can tolerate misclassification of a voxel in a certain number of ray clouds without the voxel being carved. A lower margin leads to better reconstruction of cavities.

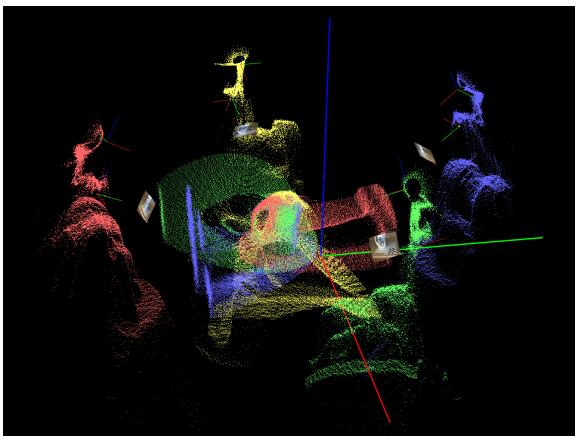
4 Using Depth Images

Rasterizing the rays works well for dense ray clouds, that is, ray clouds where the minimum distance between adjacent points is smaller than the voxel size. However, for sparse ray clouds (or, equivalently, high voxel resolutions), this method does not work because there will be unoccupied voxels which have no passthroughs and therefore will not be carved. We solve this problem by introducing the concept of *depth images*.

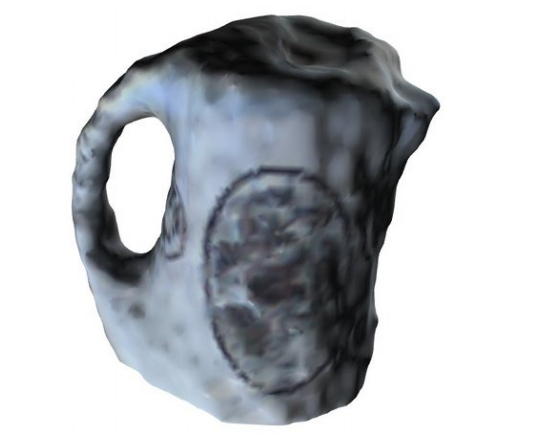
For this method, we require that all rays of the ray cloud pass through a single point in space, that is, we can assume a common origin for all rays:

$$\underline{c} = \underline{o}_i \quad \forall i$$

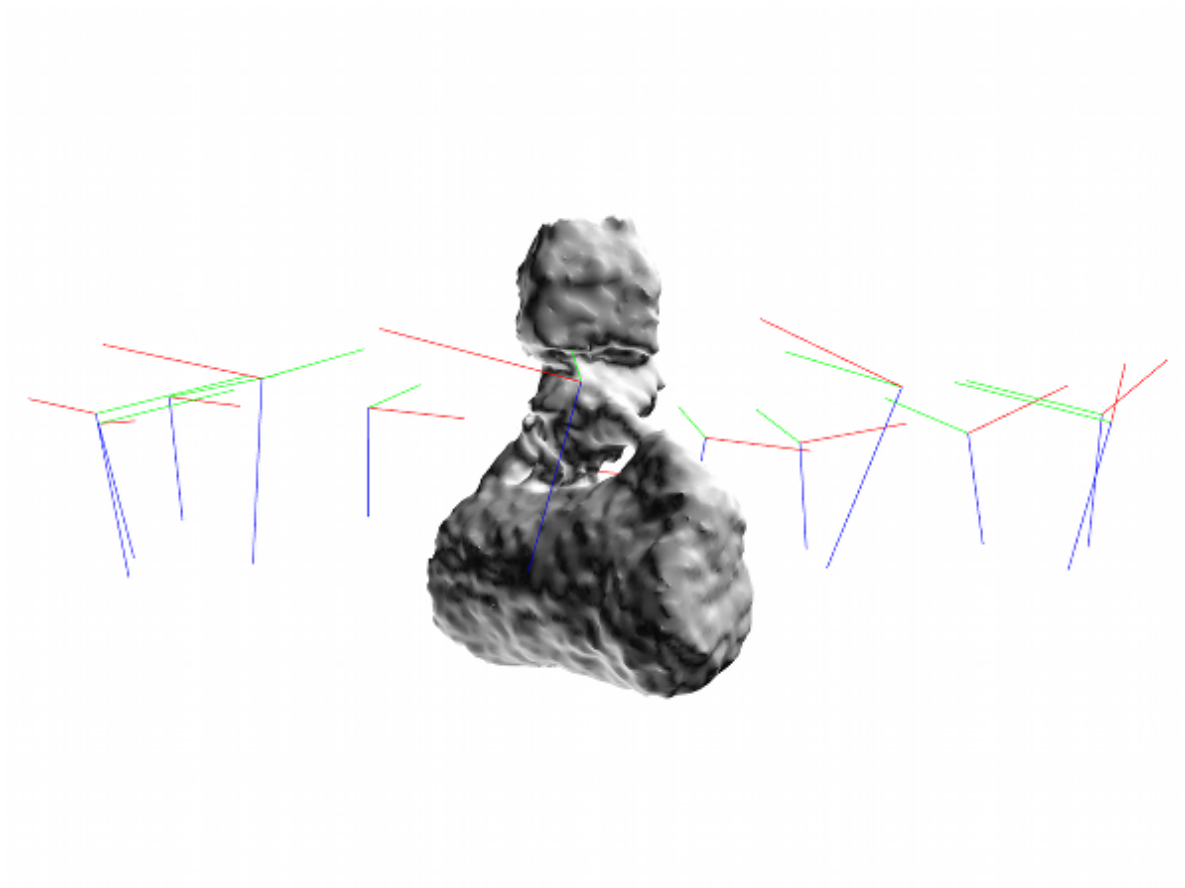
$\underline{c}$ is called the *center* of the ray cloud.



(a) The captured dataset (images and point clouds) in our visualizer software (showing four of seven point clouds)



(b) The smoothed and colorized mesh of the kettle on the table



(c) The smoothed mesh of the kettle in the hand of the robot, grasping the kettle from above, with the poses of the laser scaner

Figure 6: Results from the experiments

4.1 Storing and retrieving depth data with depth images

A depth image is a data structure that allows us to determine an approximation to the depth (that is, the distance of the closest surface from the center of the ray cloud), in an arbitrary direction within a predefined range. This is done by storing the depth $d(u,v),\ u,v \in \mathbb{N}$ for a regular grid of directions[2] and interpolating between neighboring values for grid cells for which no depth information is available.

In order to create a depth image for a point cloud, we first define a region of interest as well as a parametrization $(u(\underline{p}), v(\underline{p}))$ of rays starting at the center $\underline{c}$ of the point cloud. For each ray $(\underline{p}_i, \underline{c})$ of the ray cloud whose direction falls into the region of interest, we project the point $\underline{p}_i$ to the depth image by determining $(u_i, v_i) = (u(\underline{p}_i), v(\underline{p}_i))$ and store the depth of the ray, $\left|\left|\underline{p}_i - \underline{c}\right|\right|$ in the corresponding grid cell, $d(u_i, v_i)$.

In general, there will not be a one-to-one mapping between rays and grid cells. If multiple rays project to the same grid cell, we use the minimum of their depths:

$$d(u,v) = \min_{(u(\underline{p}_i), v(\underline{p}_i)) = (u,v)} \left|\left|\underline{p}_i - \underline{c}\right|\right|$$

The minimum is used in order to avoid misclassification of points as "free". For grid cells that no rays project to, we use the minimum depth of the neighboring cells which have a valid depth. We repeat this step until all cells have a value associated with them.

The approximation to the depth for a given direction, defined by the ray cloud center and a point p, is then given by $d(u(\underline{p}), v(\underline{p}))$.

This method is both

- fast, because the depth for any given direction can be directly looked up without searching multiple rays, and
- memory efficient, because we only store single depth values rather than points in 3D space and because for dense regions, multiple rays are combined in a single cell of the depth image

One particularly useful parametrization is given by

$$\begin{aligned} u(\underline{p}) &= \left\lfloor \frac{(\underline{p} - \underline{c}) \cdot \underline{e}_2}{(\underline{p} - \underline{c}) \cdot \underline{e}_1} \right\rfloor \\ v(\underline{p}) &= \left\lfloor \frac{(\underline{p} - \underline{c}) \cdot \underline{e}_3}{(\underline{p} - \underline{c}) \cdot \underline{e}_1} \right\rfloor \\ &\text{for } (\underline{p} - \underline{c}) \cdot \underline{e}_1 > 0 \end{aligned} \tag{3}$$

where $(\underline{e}_1, \underline{e}_2, \underline{e}_3)$ forms an arbitrary orthonormal basis. This is equivalent to a pinhole camera model centered in c, so it can easily be integrated with other data, like intensity or color information. It is also fast, because no trigonometry is involved in the projection, as opposed to using spherical coordinates.

The viewing direction of the pinhole camera model is $\underline{e}_1$. It is only suitable for a field of view that is significantly smaller than 180°. For typical applications, the region of interest will not be larger than ±90°, where the model is suitable.

[2] in 2D space, the direction will only have one parameter

4.2 Voxel carving with depth images

As before, we can use multiple depth images to create a voxel grid reconstruction of an object. We examine each voxel separately, projecting its center $\underline{c}_v$ to all of the depth images and counting the depth images where the voxel center $\underline{c}_v$ appears as free, that is,

$$||\underline{c}_v - \underline{c}|| < d(u(\underline{p}), v(\underline{p})) - \Delta d$$

A voxel is carved if the number of "free" classifications exceeds a certain minimum. The margin Δd can be determined according to 2 as before.

5 Experiments

We validated our method in two setups: the first one is an object standing on a table as can be seen in Fig. 3(a), and the second one is an object in the robot's hand. When recording the ray clouds, we have determine the pose of the laser scanner. Pose registration is beyond the scope of this work. For our purposes, we found that determining the pose of checkerboards on the wrist of the robot or on the table, respectively, with a camera mounted on the robot, is sufficient. More sophisticated methods, like [11], could be used as well. The camera can also be used to colorize the resulting model, as seen in Fig. 6(b).

For the object on the table, we recorded seven ray clouds at a low resolution, taking about 1.5 s per cloud. Each of the clouds contains about 64000 points. After filtering out points lying outside of the region of interest, about 10000 points per point cloud remain. The points clouds after filtering are shown in Fig. 3(c).

From each of the point clouds, we create a depth image. One of the depth images is seen in Fig. 3(d). Fig. 3(f) shows the resulting voxel grid. We then use the Marching Cubes algorithm[9] and Laplacian smoothing[1] to create a mesh from the voxel reconstruction. Finally, we use the camera images to colorize the mesh. The result is shown in Fig. 6(c). More sophisticated methods for creating a mesh out of a voxel grid are available, but this is beyond the scope of this work.

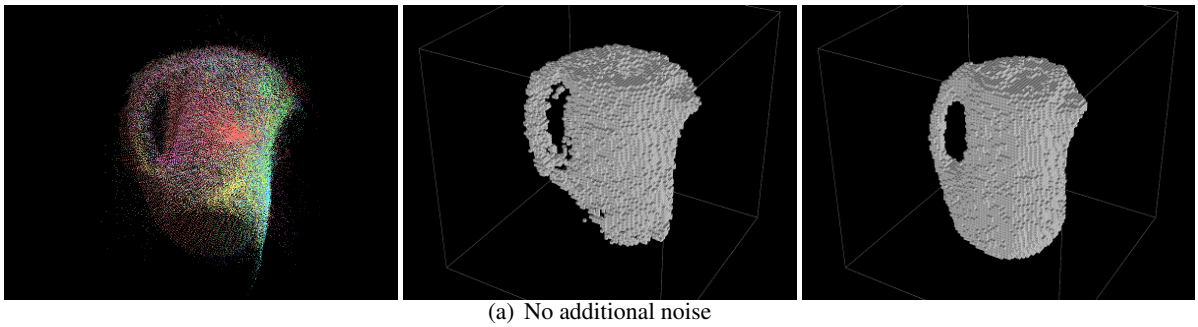
Creation of the depth images takes less than half a second. Reconstruction of the voxel grid takes about 14.7 s for a grid of $456 \cdot 10^3$ voxels. The time for generating and colorizing the mesh is less than one second.

For the object in the hand, we recorded 13 ray clouds. The result is shown in Fig. 6(c). Note that this mesh could not be colorized due to the narrow field of view of the camera.

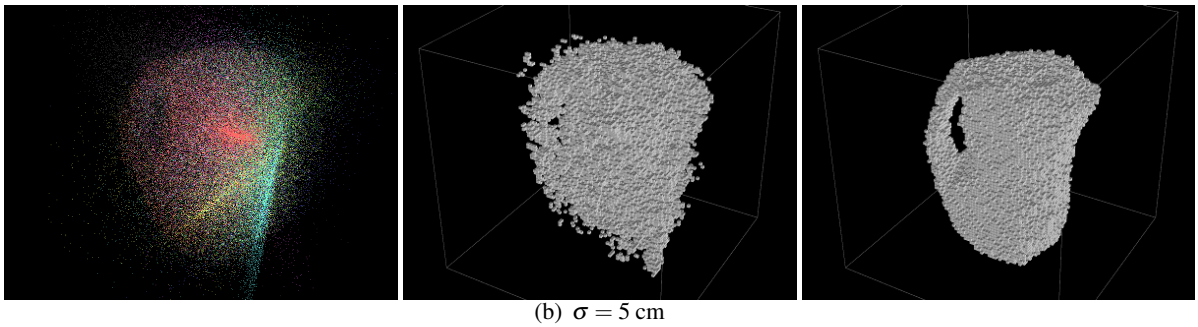
5.1 Robustness to noise

For evaluating the robustness of our method to noise, we used the acquired data and added zero-mean gaussian noise to the depth. Note that neither of these properties is required for our method - the probability distribution can be arbitrary and have a non-zero mean. We chose a maximum acceptable misclassification probability of $P_{\text{mis,max}} = 0.2$, resulting in a margin of $\Delta d \approx 1 \cdot \sigma$. The results of the reconstruction can be seen in Fig. 7.

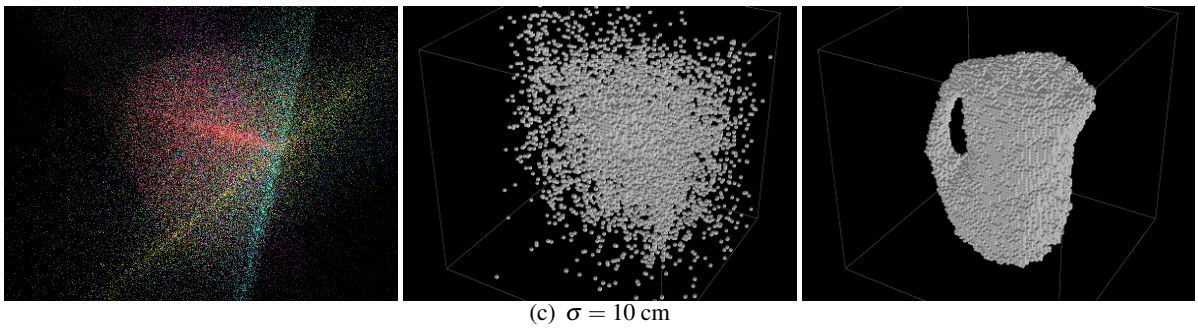
Fig. 7(a) shows the original point cloud with no additional noise. Both the conventional method, using hit information, and our method, using depth images, approximate the shape well, but with the traditional method, part of the surface to the lower left is missing due to the low density of points. In Fig. 7(b), noise with a standard deviation of $\sigma = 5$ cm was added. The conventional method results in a very jagged surface.



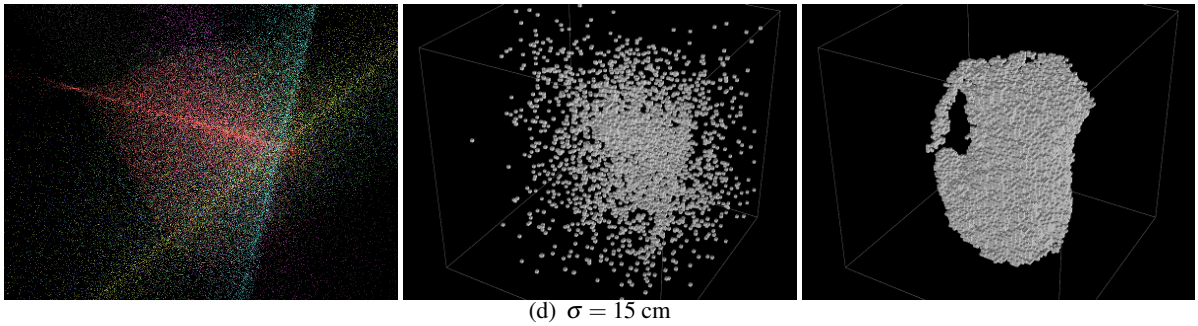
(a) No additional noise



(b) $\sigma = 5$ cm



(c) $\sigma = 10$ cm



(d) $\sigma = 15$ cm

Figure 7: The effect of noisy data: the top row shows the raw point clouds. The middle row shows the result of the reconstruction using hit data only. The bottom row shows the result of our method. Our method is robust against noise, even in the order of magnitude of the object size.

With more noise (Fig. 7(c), $\sigma = 10$ cm, and Fig. 7(d), $\sigma = 15$ cm), the object can barely be recognized in the point cloud any more. The conventional method does not produce any useful results, while our method still performs well. Note that in Fig. 7(d), the standard deviation of the noise is more

than half the size of the object.

6 Discussion

6.1 Recovering partially occluded objects

If the robot is holding an object in its hand, part of the object is occluded by the hand and thus cannot be correctly reconstructed. This problem can be solved by creating two sets of scans of the object using different grasps; for example, for one set of scans, the object could be grasped from below, as in Fig. 1, and for the other one from above or through the handle.

The poses of the scans from the separate sets have to be registered with respect to each other (which is beyond the scope of this work). After pose registration, our method can be applied without change. As we require less than half of the ray clouds to agree on a point being free in order to carve the corresponding voxel, the voxels that were only occupied in half of the scans (that is, the voxels making up the hand) will automatically be removed.

6.2 Incorporating priors

Our method does not require any priors about the object shape, as long as enough scans are available. However, in many application, certain priors, such as smoothness constraints, will be available. Such information can easily be integrated with our method, for example by treating the voxel grid as a markov random field[5] and using the passthrough information as a local propery.

6.3 Behavior under high noise conditions

If the depth information is very noisy, we have to use a large margin Δd. If the margin is significantly larger than the largest cavity of the object, the information from object passthroughs (see Fig. 3.2) becomes irrelevant because any voxel that has object passthroughs will also have background passthroughs.

For a ray cloud with a center, the information provided by background passthroughs is equivalent to a silhouette created from a picture taken by a camera located in the center of the raycloud. Thus, for high noise conditions, our method is identical to the visual hull[8] algorithm, but without the need to explicitly create the silhouettes. As the noise gets lower, additional information is integrated.

6.4 Coverage of the object with noisy data

When inspecting an object, we want to answer the question of which position to record the next scan from. In the presence of noise, rays do not provide reliable passthrough information close to the point where they hit a surface. This means that for any given voxel, of all the rays that have a passthrough through this voxel, there should be at least one ray for which the surface it hits is further away from the voxel than Δd.

For example, Fig. 8 shows an object standing on a table. We consider the marked voxel in front of the object. Rays (a) and (b) hit a surface close to the voxel. The margin Δd, as determined from 2, is denoted by a dotted line segment. Thus, for rays (a) and (b), the voxel will be labeled as "unknown" and not carved. Only from ray (c) can we get the information to label the voxel as "free".

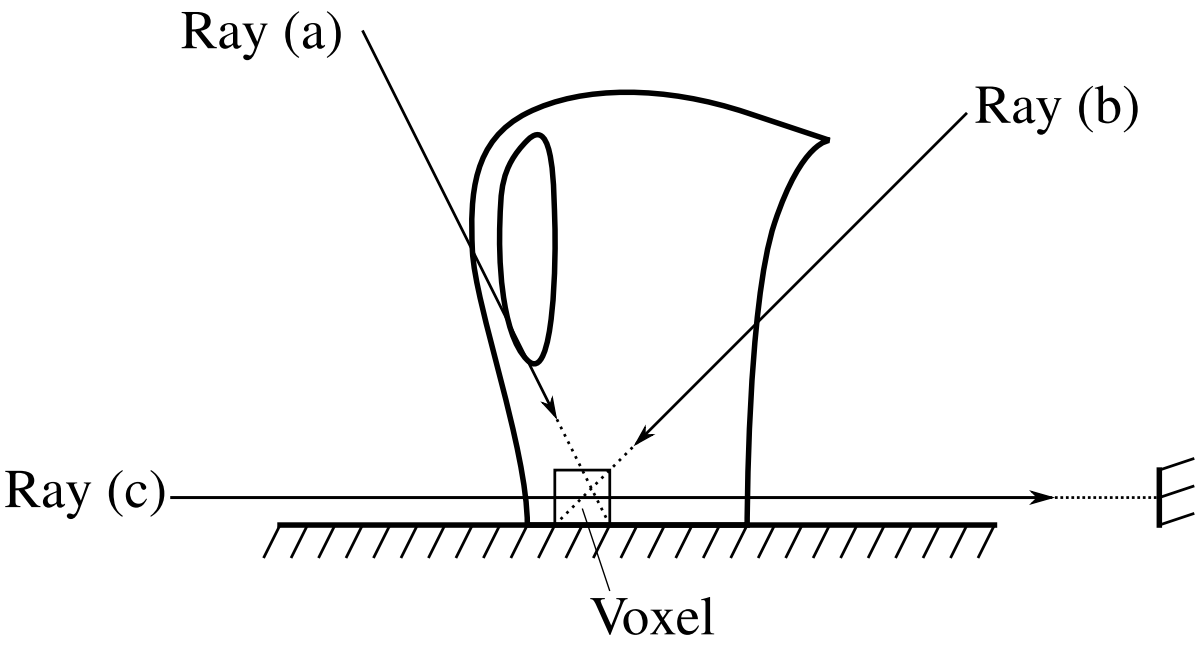


Figure 8: On Object coverage: rays (a) and (b) do not provide information about the marked voxel in the presence of noise but ray (c) does

7 Conclusions

In this paper, we showed that passthrough data carries more information than hit data. Passthrough data can be used to reconstruct objects using a volume based method, even in the presence of large amounts of noise.

We presented a memory efficient method for reconstructing objects from sparse data that is not affected by grazing points. Our method is fast in both data acquisition (because it can use sparse data, which can be rapidly recorded), and computation. It is thus useful for a personal robot operating in unknown environments. We validated our method in two different setupts using real-world data.

8 Acknowledgements

Martin Herrmann is supported by a Baden-Württemberg Stipendium. This material is based upon work partially supported by the National Science Foundation under Grant No. EEC-0540865. We are grateful to the members of the Personal Robotics project at Intel Labs Pittsburgh for useful discussions and help with HERB.

References

[1] Alexander Belyaev. Mesh Smoothing and Enhancing. Curvature Estimation. http://www.mpi-inf.mpg.de/~ag4-gm/handouts/06gm_surf3.pdf.

[2] C. Ericson. *Real-time collision detection*. Morgan Kaufmann, 2005.

[3] Hokuyo. http://www.hokuyo-aut.jp/02sensor/07scanner/uxm_30ln.htm.

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics(ACM)*, 26(2):71–78, 1992.

[5] R. Kindermann and J.L. Snell. *Markov random fields and their applications*. American Mathematical Society, 1980.

[6] R. Kolluri, J.R. Shewchuk, and J.F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21. ACM New York, NY, USA, 2004.

[7] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *The International Journal of Computer Vision*, 38(3):199–218, 2000.

[8] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2), February 1994.

[9] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4), July 1987.

[10] B. Mederos, N. Amenta, L. Velho, and L.H. de Figueiredo. Surface reconstruction from noisy point clouds. In *Proceedings of the third Eurographics symposium on Geometry processing*, page 53. Eurographics Association, 2005.

[11] A. Nuchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6D SLAM with an application in autonomous mine mapping. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, volume 2, 2004.

[12] Siddhartha Srinivasa, David Ferguson, Casey Helfrich, Dmitry Berenson, Alvaro Collet Romea, Rosen Diankov, Garratt Gallagher, Geoffrey Hollinger, James Kuffner, and J Michael Vandeweghe. HERB: a home exploring robotic butler. *Auton. Robots*, 28(1):5–20, January 2010.

[13] Y. Tseng, K. Tang, and F. Chou. Surface Reconstruction from LiDAR Data with Extended Snake Theory. *Lecture Notes in Computer Science*, 4679:479, 2007.