Robot control via region-based 3d reconstruction

Lilia Markham, Stephanie Cord Melton, and Zachary Dodds
Harvey Mudd College
301 Platt Blvd. Claremont, CA 91711
USA
lmarkham@hmc.edu, scmelton@cs.hmc.edu, dodds@cs.hmc.edu

ABSTRACT
Reconstructive visual control of mobile robots currently focuses on sparse image features, e.g., points and lines. This work extends the reach of 3d reasoning to inexpensive – and increasingly ubiquitous – platforms whose visual input consists of *segmented regions*, rather than raw pixels. We propose and evaluate a pair of algorithms that enable two fundamental facets of robot control: obstacle avoidance and landmark reacquisition. Our results suggest that space-carving outperforms triangulation when using image regions to reason about one's 3d environment.

KEY WORDS
Vision-based control, autonomous robotics, 3d reconstruction, modeling and visualization

1. Introduction

Vision offers perhaps the highest bandwidth-to-cost ratio to robotic systems. In theory at least, one camera is a complete sensor suite. This potential, countered by pixels' close coupling of lighting, scene, and optics, has spurred decades of reconstructive research - reaching as far back as Shakey and the Stanford Cart. The progress in 3d reasoning from image streams has accelerated over the past five years: today's algorithms untangle even monocular data into consistent and accurate environmental representations [1-4]. Results such as [5] are, quite frankly, inspiring.

Yet these remarkable systems exhibit a relatively narrow focus in their design: they rely on sparse, accurately reconstructed image features. Stepping back, one might consider a spectrum of reconstructive approaches in which features' precision trades off against their density, as sketched in Figure 1.

Work to date emphasizes the bottom-up paradigm: SIFT, SURF, KLT, et al. provide locally distinct feature points, triangulate them into 3d via robust statistics, and hang textures onto the result [6,7]. An alternative path runs from left to right in Figure 1: segmentation algorithms [8,9] produce regions whose interframe correspondences yield a rough set of 3d surfaces. From those surfaces, texture mapping and further image processing could sculpt more accurate world representations -- but only as required or desired.

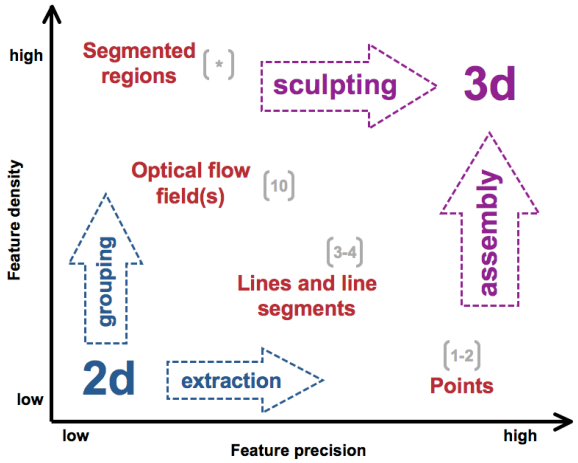

Figure 1. A taxonomy of visual reconstruction approaches. Reference numbers place algorithms according to the precision and density of the underlying 2d features they use and 3d primitives they create. Whereas most algorithms first extract accurately localized features, usually points, and then assemble them into a 3d cloud, this work [∗] transforms image regions into coarser estimates of world structure.

Indeed, with this work we argue that despite the successes of environmental assembly, environmental sculpting offers a complementary approach useful in situations where precise feature matching can fail:

(1) with densely or dynamically textured natural surfaces – such as bark, foliage, or water
(2) with featureless surfaces, e.g., indoor office environments
(3) for tasks, such as obstacle avoidance, in which the density of the world's representation supersedes its absolute accuracy - or
(4) when only segmented regions are available, not the images themselves.

This fourth use case has become increasingly common as low-cost platforms for research and education proliferate. For example, the Mach 5 [11], KIPR's XBC [12], and Lego's NXT [13] offer access only to presegmented data through Cognachrome [14], CMUcam [15], or Mindsensors [16] interfaces. Other widely accessible platforms offer "blob" tracking as an option for a

pedagogically accessible introduction to vision [17,18]. We contend that blobs, whether optional or required, do not preclude 3d reasoning for such platforms.

1.1 Context and contributions

Figure 1's counterpoint between world-assembly and world-sculpting is precisely the computer vision community's dichotomy between structural reconstruction and space carving. This work simply leverages the fact that an incomplete and/or coarse space-carved approximation can offer a great deal of utility to a robotic agent. Perhaps because of its computational cost and pose-accuracy requirements, pure space carving has not seen substantial use for robotic environmental modeling. For instance, [19] eases the burden with human-segmented and matched image patches; [20] space-carves automatically, but through a 1d camera retina.

In contrast to many reconstructive algorithms which use direct range sensing along with vision [21,22], we follow [1] by using only monocular data and approximate odometry, such as that available on the low-cost platforms cited above. Our work concentrates on cameras whose optical axis remains parallel to the ground plane; thus, it allows fewer camera freedoms than [1]. This ground-plane restriction is not inherent; rather it reflects a natural starting point for this work. This paper thus sets a foundation for robotic task-performance through environmental sculpting with the following three contributions:

(1) Three algorithmic variations for coarse reconstruction via segmented image sequences (Sec. 2)
(2) Metrics and results of these algorithms' perfomance on two fundamental robot control tasks: obstacle-avoidance and landmark reaquisition (Sec. 3)
(3) Data sets and source code openly available from [23].

2. Algorithms for reconstruction via regions

2.1 System inputs: 2d image segments

We begin by segmenting all input images using the Edison segmentation system [25]. Edison only segments; our system then proceeds to compute region correspondences across the image series. Regions are matched between consecutive images based on color, size, and shape characteristics. The use of region locations within each image can greatly improve the quality of correspondences, but forces us to assume small camera displacements between images. This process, illustrated in Figure 2, results in some number of objects to be reconstructed. These 'objects' are simply collections of corresponded image segments, representing the same physical object viewed from different positions. Rather than saving each segment as an image, however, we store only the dimensions and location (in pixels) of its

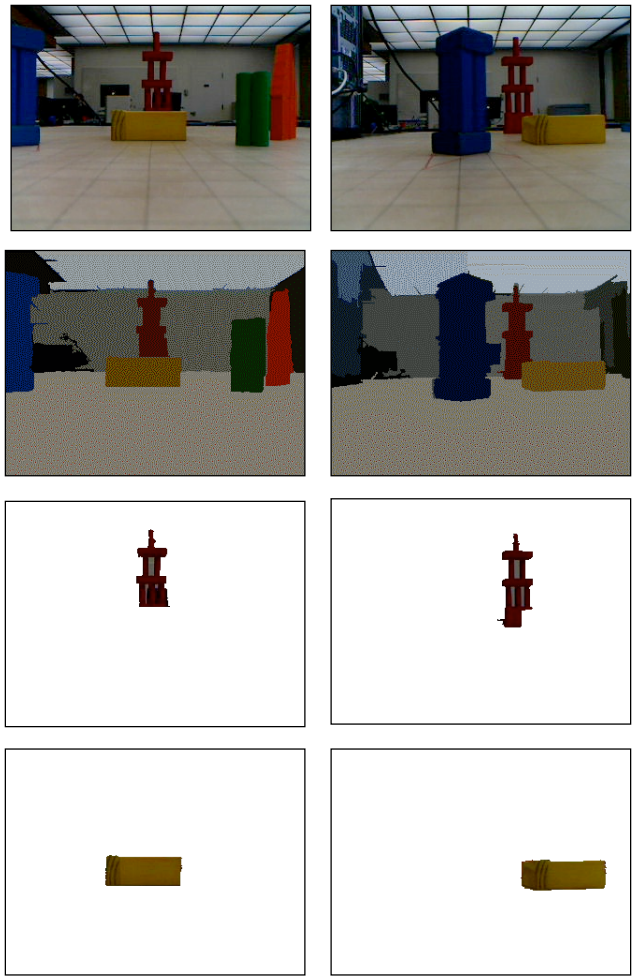

Figure 2. The top row shows two raw images; the second row depicts Edison's segmentations of those images, and the final two rows illustrate two corresponding segments of two objects.

bounding box as well as a pointer to the global coordinates and orientation (in the horizontal plane) of the camera at the time that image was taken.

Formally, let $O = \{O_0 \ldots O_N\}$ be a set of objects, where $O_i = \{S_{i,0} \ldots S_{i,M}\}$ is a collection of 2d image segments corresponding to the same 3d object in the real world. The index i differs for different objects; j differs for different images. Suppressing these indices among the components for simplicity, each segment $S_{i,j}$ comprises the four elements of $\{C,P,r,h\}$, where C describes the camera position and orientation associated with the image from which $S_{i,j}$ was taken; P is the two-dimensional center point of $S_{i,j}$ in image j; r is the segment's radius; and h is its height.

In the next two subsections we detail the reconstructive algorithms themselves; we follow these with results and a comparative evaluation of the approaches.

2.2 Cylindrical reconstruction via triangulation

Step 1: For each segment, $S_{i,j}$, in object O_i, we assume that the image plane lies a constant distance, *f*, in front of the camera, and convert the 2d vector, $P_{i,j}$, into a 3d vector, $P_{i,j}'$, which points from camera position $C_{i,j}$ to the center of segment $S_{i,j}$. This involves scaling the components of $P_{i,j}$ from pixels to spatial dimensions and rotating them into the global coordinate system's basis, since they are initially measured relative to the norm of the image plane:

```
Pi,j* = ( c*getX(Pi,j), c*getY(Pi,j), f )
Pi,j' = rotate(Pi,j*, getTheta(Ci,j) )
```

The constant, *c*, used in this conversion, is the ratio of spatial dimensions to image pixels at a distance *f* from the camera. The value of *c* may be determined for each camera and image resolution by capturing an image of a flat surface, positioned at distance *f* from the camera and oriented perpendicular to the norm of the image plane. Now, letting *p* be the length of the surface in the image, measured in pixels, and letting *l* be its length in space, measured in the same units used to measure *f*, it follows that $c = l/p$. The name *f* invokes the idea of focal length, but the actual value used for f is arbitrary, and need not be related to the actual focal length of the camera.

Step 2: Perform a triangulation on the center points of each pair of segments and average all the results. Psuedocode for this step is given below.

```
reconstructCenterPoint(Oi) {
    good_results = empty list
    for each pair si,j, si,k in Oi {
       Pi,j' = (c*Pi,j[0], c*Pi,j[1], f)
       Pi,k' = (c*Pi,k[0], c*Pi,k[1], f)
       (x,y,z,fail) =
             triangulate((Ci,j,Pi,j'),(Ci,k,Pi,k'))
       if fail != 0
          append (x,y,z) to good_results
    }
    if good_results is not empty
       (xavg,yavg,zavg) = average of good_results
    else
       (xavg,yavg,zavg) = null
    return (xavg,yavg,zavg)
}
```

The `triangulate()` method approximates the position of an object, given two different views, in much the same manner as a binocular vision system. It takes as input two camera positions ($C_{i,j}$ and $C_{i,k}$) which we may think of as two corners of a triangle, and two corresponding 3d vectors, ($P_{i,j}'$ and $P_{i,k}'$) which point along the sides of the triangle toward the third corner, which is assumed to be the center of the object being reconstructed. Our implementation converts these inputs into a system of two linear equations and uses singular-value decomposition to solve for the intersection (or the least squares approximation, if no intersection exists) of the two sides of the triangle.

Due to errors in the measurement of camera coordinates, especially when the distance between the pair of positions is small, some combinations of segments will cause the center of the segment to reconstruct behind the camera. If this occurs, `triangulate()` sets the `fail` flag to 1, and the result is not averaged into the final estimate of object location.

Step 3: Take the average radius and height, r_{avg} and h_{avg}, resp. over all the segments in the object.

Done: The resulting reconstruction may be represented as a cylinder centered at $(x_{avg}, y_{avg}, z_{avg})$ and having a radius and height of r_{avg} and h_{avg}, respectively. Figure 4 shows an example of these reconstructed cylinders.

Alternatively, the reconstruction may be represented as a sheaf of intersecting planes, all centered at $(x_{avg}, y_{avg}, z_{avg})$ where each plane corresponds to a segment in the object, and as such is oriented parallel to that segment's image plane and keeps that segment's radius and height as its own.

2.3 Polygonal reconstruction via space carving

As with triangulation, for each object O_i, consider each image segment for that object, $S_{i,j}$. Figure 3 summarizes the overall approach.

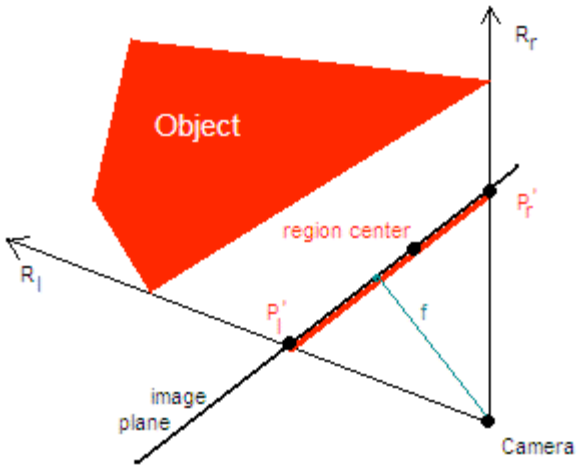

Figure 3: A top-down cross section of the space-carving approach to object reconstruction. The red line represents an image segment. R_l and R_r are the 3d rays delimiting the object; they intersect the image plane at P_l and P_r, respectively. With additional images, more space is "carved away."

The following steps detail the space-carving algorithm sketched in Figure 3.

Step 1: Locate the x-z coordinates of the right and left edge of the segment $S_{i,j}$. Using the distance of the camera

to the image plane *f*, the 2d coordinates and measures of the segment, and 3d coordinates of the camera that captured $S_{i,j}$, we can locate the 3d coordinates of the right and left edges of the region in 3d. However, since we are only concerned with the x-z footprint of the object, we will disregard the y coordinates and only concern ourselves with the x-z coordinates of the edges of the bounding box.

From the center and width of the image plane, locate the left and right edges P_l and P_r within the image plane. Find the distances between the image plane center and P_l and P_r and call them D_l and D_r, respectively. Scale those distances appropriately for objects at distance *f* from the camera to find the distances in x-z space, D_l' and D_r'.

Locate the center of the image plane P_c' in the x-z plane by taking the point *f* distance directly in front of the camera's line of sight. Find the points P_l' and P_r' that are, respectively, D_l' and D_r' away from P_c' along the line perpendicular to the camera's line of sight.

Step 2: Find the rays R_l and R_r going from P_c' to P_l' and P_r', respectively. Whatever of the object that can be seen by that camera location and orientation must be between those rays. Using this knowledge, we partition our world into space that might contain our object, and space that definitely does not. We choose to artificially constrain the area that the object could be in by assuming that there is a maximum distance, called *max_visibility*, away from the camera that visible objects can reasonably be.

We proceed to build a triangle that bounds the object: take the points P_l'' and P_r'' that are *max_visibility* away from P_c' along R_l and R_r. Construct a bounding triangle T_i whose vertices are at C, P_l', and P_r'.

Step 3: Each segment $S_{i,j}$ gives us a different bounding triangle for our object O_i. Thus, the intersection of bounding triangles from all segments gives us the closest estimate of the object's location and shape.

Done: To finish the reconstruction, take the final space-carving polygon as the base of the reconstructed 3d object. Use the height and y-coordinate from triangulation to center the polyhedron and define its volume.

3. Evaluation and results

Figure 4 shows examples of these reconstructions for a two-object world. Following these examples we describe metrics for evaluating such maps in two important robot control applications.

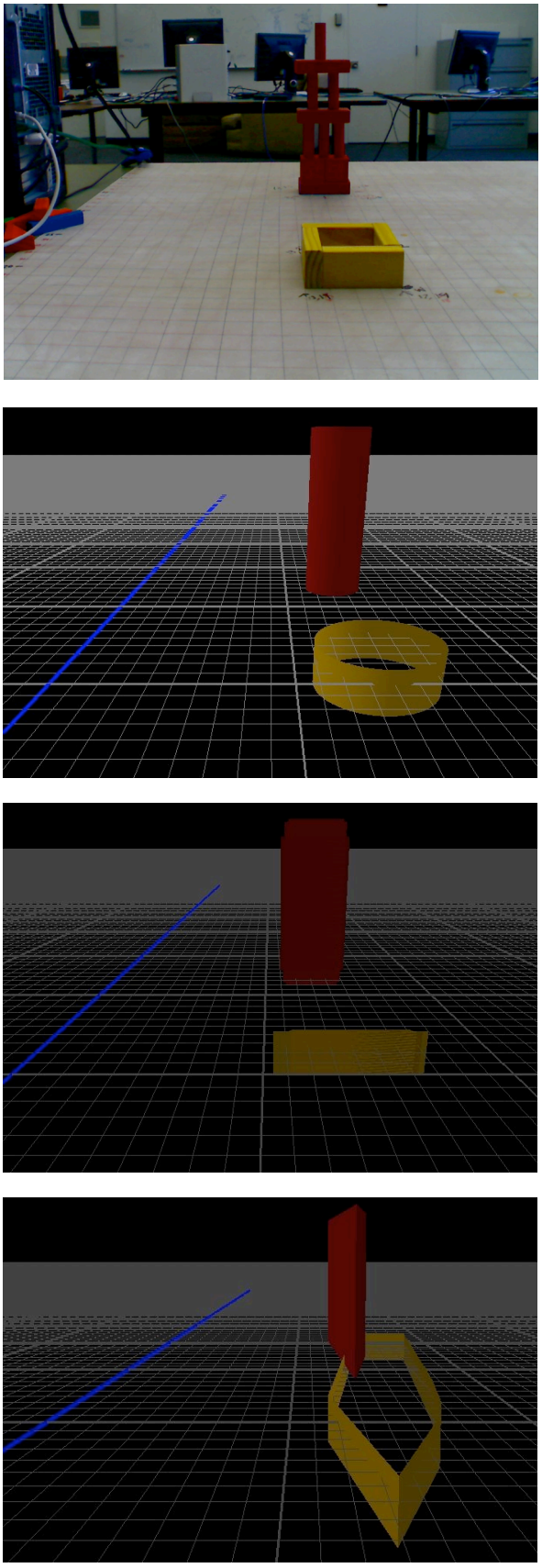

Figure 4. (Top) Raw data including two objects (Middle rows) triangulated cylinders and plane-sheaves (Bottom) Space-carved results, all for the same blocks-world input data.

3.1 Metrics for assessing reconstructive control

One potential application for 3d mapping from vision is the navigation of vision-enabled robotic agents. In such a case, we would like our robot to successfully avoid obstacles, but still move freely through the world. In order to asses our map's aptitude for these types of task, we define the following two metrics. *Recklessness* represents the fraction of actual obstacle area not enclosed by the corresponding reconstructed object. *Paranoia* is the fraction of reconstructed object area not enclosed by the actual obstacle.

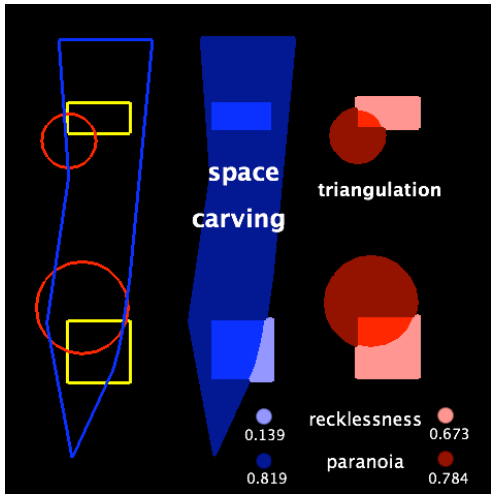

Figure 5. Example of paranoia and recklessness metrics using relatively few images. The yellow rectangles represent actual objects; blue indicates the space-carving approach, and red regions are triangulated.

To assess our ability to reacquire landmarks using our map, we use the following algorithm: Choose a novel viewpoint and move the camera to it in both the real and rendered worlds. Capture an image from each world, segment these two images (as shown in Figure 6), and compute region correspondences between them. Then define the distance between the two images as the sum of the difference measurements of the corresponded regions. The smaller this number is, the better our measure of ability to recognize landmarks is.

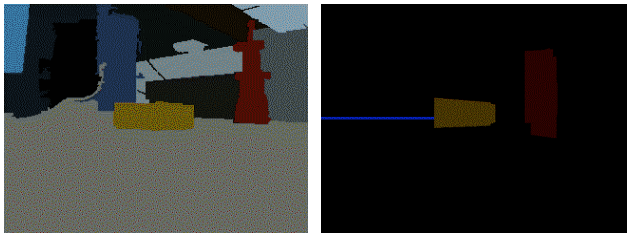

Figure 6: Two segmented images captured from the same position in the real and rendered worlds, resp.

3.2 Contrasting triangulation with space-carving

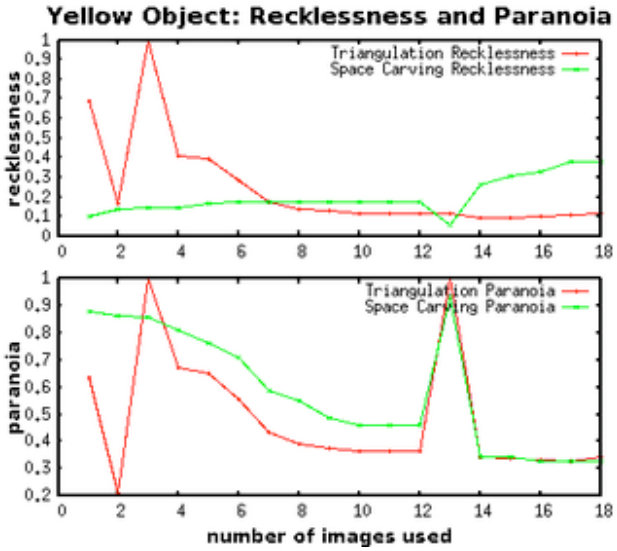

Figure 6. Comparison of recklessness and paranoia for triangulation and space carving.

As Figure 6 illustrates, triangulation outperforms space carving in both metrics for large numbers of images, though for small numbers, triangulation is clearly superior in paranoia and space carving is superior in recklessness.

More viewpoints increase the accuracy of triangulation because they increase the accuracy of the centroid estimate. The radius of the reconstructed cylinder tends to remain relatively constant as the number of images increase.

Space carving, in contrast, performed poorly as the number of viewpoints increased. Given ideal segmentation, space carving would continue to improve in paranoia while never sacrificing perfect recklessness. Correct segmentation would guarantee that any space inside the original object would be seen by the camera as such and would never be carved away. However, as we have imperfect segmentation, sometimes pieces of the true segment are lost. This means that space inside the actual object is carved away. As the number of images increases, inevitably so does the number of incorrect segments, and so does the recklessness of the space carving reconstruction.

4. Perspective

To be sure, our blocks-world image sequences highlight the strengths of region-based reconstruction. This in no way detracts from feature-based, bottom-up approaches to world-building. Rather, it provides a modular basis onto which alternative segmentation algorithms easily integrate, e.g., for complex natural scenes or feature-sparse indoor worlds.

It would be surprising if the single strategy of interpolating from a small subset of accurately-localized image data sufficed for all visual conditions. The dual approach -- coarsely placing and/or sculpting objects from the visible environment and refining as needed -- opens up complementary capabilities. We look forward to the robust algorithms that will emerge from combining these approaches into hybrid reasoning that smoothly incorporates both visually distinct features and more diffusely delimited data. Success in human-robot interaction will ultimately depend on robots' ability to reason about the 3d world in which humans perceive themselves. We will strive to help robotic systems realize this ability through the inexpensive, powerful, and richly varied channel of monocular vision.

Acknowledgements

The authors gratefully acknowledge the support of NSF DUE REU #0451293, CCLI #0536173, and Harvey Mudd College.

References

[1] A. J. Davison, I. Reid, N. Molton and O. Stasse "MonoSLAM: Real-Time Single Camera SLAM" *IEEE Trans. PAMI* (accepted for publication), 2007.

[2] D. L. Cardon, W. S. Fife, J. K. Archibald, and D. J. Lee. "Fast 3D reconstruction for small autonomous robots" (IECON_Fast3D_112005.pdf)

[3] P. Smith, I. Reid, and A. J. Davison. "Real-Time Monocular SLAM with Straight Lines" *Proceedings, British Machine Vision Conference*, 2006.

[4] Y.-Q. Cheng, X.G. Wang, R.T. Collins, E.M. Riseman, and A.R. Hanson. "Three-Dimensional Reconstruction of Points and Lines with Unknown Correspondence across Images" International Journal of Computer Vision 45(2) pp. 129-156, 2001.

[5] P. Mordohai, J.-M. Frahm, A. Akbarzadeh, B. Clipp, C. Engels, D. Gallup, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, H. Towles, G. Welch, R. Yang, M. Pollefeys and D. Nistér, "Real-Time Video-Based Reconstruction of Urban Environments", *Proceedings of 3DARCH: 3D Virtual Reconstruction and Visualization of Complex Architectures*, Zurich, Switzerland, July, 2007

[6] D. Nistér. "Automatic dense reconstruction from uncalibrated video sequences" PhD Thesis, Royal Institute of Technology KTH, Stockholm, Sweden, ISBN 91-7283-053-0, March 2001.

[7] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch, "Visual modeling with a hand-held camera" *International Journal of Computer Vision* 59(3), 207-232, 2004.

[8] D. Comanicu and P. Meer: "Mean shift: A robust approach toward feature space analysis." *PAMI*, 24, 603-619, May 2002.

[9] P. Felzenszwalb and D. Huttenlocher. "Efficient Graph-Based Image Segmentation" *IJCV*, 59(2) September 2004.

[10] Weber, J and Malik, J. "Rigid Body Segmentation and Shape Description from Dense Optical Flow under weak perspective" PAMI, 139-143, Feb 1997.

[11] Newton Labs' Mach 5 platform (accessed 7/15/07) `www.newtonlabs.com/mach5.htm`

[12] R. LeGrand, K. Machulis, D. P. Miller, R. Sargent, and A. Wright. "The XBC: a modern low-cost mobile robot controller" Edmonton, Canada, IROS 2005.

[13] M. McNally, F. Klassner, and C. Continanza. "Exploiting MindStorms NXT: Mapping and Localization Projects for the AI Course" FLAIRS-20 Key West, FL May 7-9, 2007.

[14] R. Sargent, B. Bailey, C. Witty, and A. Wright. "Dynamic object capture using fast vision tracking" *AI Magazine*, Spring 1997, Volume 18, No. 1

[15] A. Rowe, C. Rosenberg, and I. Nourbakhsh. "A low-cost embedded color vision system" IROS 2002, vol1. pp. 208-213.

[16] Mindsensors NXT camera, `mindsensors.com`

[17] J. Bruce, T. Balch, and M. Veloso. "Fast and inexpensive color segmentation for interactive robots" IROS 2000 volume 3, pp. 2061 – 2066.

[18] B. Gerkey, R. T. Vaughan and A. Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". ICAR 2003, Coimbra, Portugal, pp. 317-323

[19] R. Ziegler, W. Matusik, H. Pfister, and L. McMillan. "3D reconstruction using labeled image regions" 2003 Eurographics Symp. on Geometry Processing, pp.1-12, 2003.

[20] A. Eppendahl and A. Ojamaa, "Seeing Empty Space in an Environment without Silhouettes" *Proceedings, AMiRE* 2005.

[21] D. Hähnel, W. Burgard, and S. Thrun. "Learning compact 3D models of indoor and outdoor environments with a mobile robot" *Robotics and Autonomous Systems*, 44:15-17, 2003.

[22] Ohno, K.; Tadokoro, S. Dense 3D map building based on LRF data and color image fusion" *Proceedings, IEEE Int. Conf. On Intelligent Robots and Systems* (IROS), pp. 2792-2797, Aug. 2005.

[23] `www.cs.hmc.edu/twiki/bin/view/Robotics/ThreeDFromTexture`, this work's web site.

[24] **CGAL**, Computational Geometry Algorithms Library, `http://www.cgal.org`

[25] Edge Detection and Image Segmentation (`EDISON`) System, Robust Image Understanding Lab, Center for Advanced Information Processing, Rutgers University, 2002.