Cubelet Detection and Localization by Template Matching

Charles Chen
Electrical and Computer Engineering
University of Colorado at Boulder
Boulder, CO 80302

Charles Dietrich
Computer Science
University of Colorado at Boulder
Boulder, CO 80302

Chris Miller
Computer Science
University of Colorado at Boulder
Boulder, CO 80302

***Abstract*—We describe a method for detecting and localizing Modular Robotics Cubelets in a scene, for use as a subsystem in a system for autononmous assembly of Cubelets constructions using a robotic arm and grasper. We use a camera mounted over the work surface. Our method returns two-dimensional position and angle of rotation as well as the type of Cubelet. We obtain position and rotation using template matching on images preprocessed with a novel line detection algorithm. We obtain the type of Cubelet by using color information. We also describe a method to obtain depth information from a camera with an RGB-D sensor such as the Xbox Kinect.**

I. Introduction

The purpose of this research is to produce a vision subsystem to the system necessary to accomplish the overall goal of 'robots building robots'. The 'robots building robots' project involves the assembly of robots made up of Modular Robotics Cubelets [2]. Cubelets are a robotic toy consisting of a set of small cubes. Each cube, or Cubelet, is approximately 50mm on a side. Cubelets attach to one another by way of keyed magnetic faces. Each Cubelet contains a small computer. A Cubelet may be an actuator, a sensor, a thinking block or a battery block. Actuators have an actuation face and respond to incoming signals. Sensors have a sensor face and sense the environment and send out a signal. Thinking blocks change the signal received from neighbors in a predetermined way, e.g. a Inverse block inverts the signal received. The battery block provides power.

For the purposes of the overall goal of 'robots building robots', we are concerned with autonmously assembling the Cubelets into a predetermined structure that forms a robot. We assume that the Cubelets start scattered over the work surface, a flat, featureless surface. We intend to build the construction using the Clam arm [3], a small robotic arm intended for research, with a suitable grasper. We also intend to use ROS (Robot Operating System) [4] to bring together the entire system.

We assume that a vision subsystem is a necessary subsystem of the overall system. The vision subsystem will be used to direct the arm and grasper. The vision subsystem reifies visual and aligned depth camera data into an ontology that represents the arrangement of Cubelets on the work surface. This ontology can then be consumed by other parts of the system.

Our engineering solution provides a partial technology for providing a useful ontology. The ontology should contain position and rotation for each Cubelet as well as the type of Cubelet. Our technology aims to provide two dimensional position and rotation information and partial information on the type of Cubelet at a reasonable frame rate.

We hypothesize that template matching on images preprocessed with a novel line detection algorithm can provide position and orientation data. We hypothesize that color matching on matched areas can provide partial information on the type of Cubelet.

A. Background

The problem of providing the desired ontology is largely one of object detection, a well-known problem in computer vision. Object detection methods include feature detection methods such as SIFT [5] and SURF [6] as well as template matching, segmentation and classification of segments, and flexible template matching. Feature based-methods are generally faster than template matching. We present evidence that SURF does not work well with Cubelets due to their particular appearance.

II. Materials and Methods

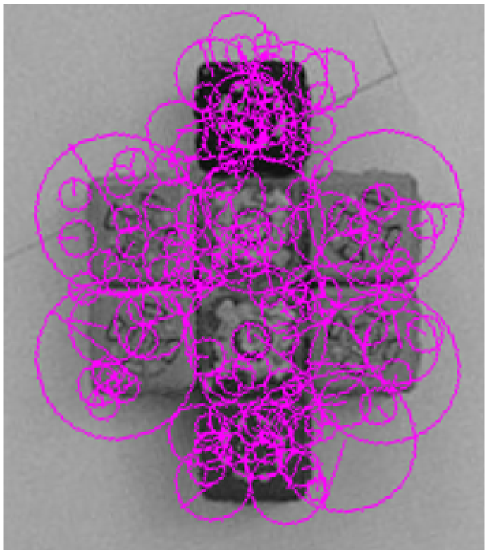
Our experimental setup consisted of a Xbox Kinect camera [7] mounted approximately 65cm above the work surface. This distance was far enough for the depth sensor to work even if the blocks are stacked, but close enough to make out features on the faces of the Cubelets. The work surface was covered in white paper. The lighting was not controlled but was normal office lighting.

We generated three sets of 10-14 images each. Within each set, the images were taken with the same setup and lighting conditions. Each image takes a picture of an arrangement of Cubelets. The Cubelet arrangements were randomly determined by us.

Our software is written in Python and uses ROS with OpenCV [8] and OpenNI [9].

A. *SURF Feature Detection*

One image from each set was used as ground truth. For this image, we hand-labeled the location of one Cubelet of each color with (x,y) information. For each set of images, we determined a template radius, corresponding roughly to the circular portion of the cube face.

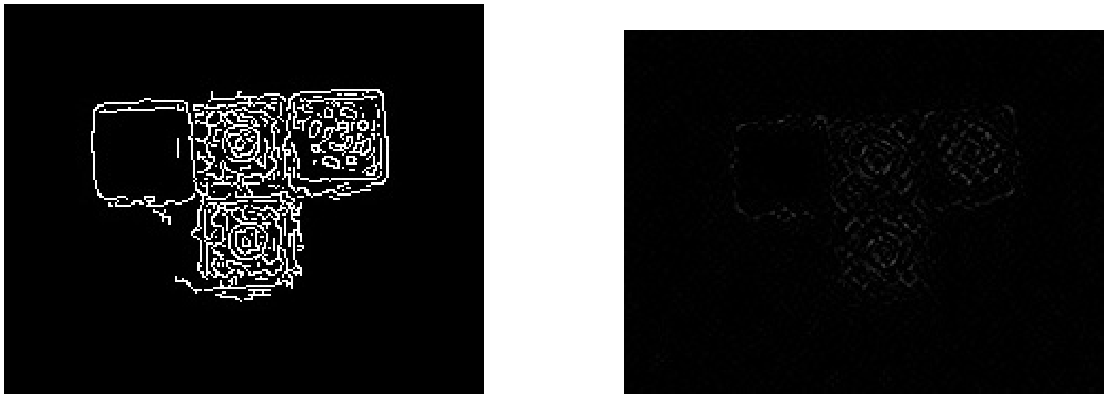


The first method we attempted was trying to match SURF features between a template image consisting of a single cube and test images of a scene containing one or more Cubelets. We were able to obtain many features for each Cubelet. The first issue we found was the 4-way rotational symmetry of the Cubelets' faces caused erroneous feature matches. In an effort to fix this problem, we selected only features on one quarter of the face. However, we found that even then the features were too easily confused with each other, i.e. the features were not stable.

We also attempted to use SURF features to detect the Cubelets through their rotational symmetry. Because they are rotationally symmetrical at 90 degrees, we assumed this would make the features appear more stable. This method has been effective for other scenarios [1], but the lack of stable features in Cubelets was still too much of a problem for it to work here. Additionally, this method finds symmetric matches other than the Cubelet faces.

B. *Template Matching*

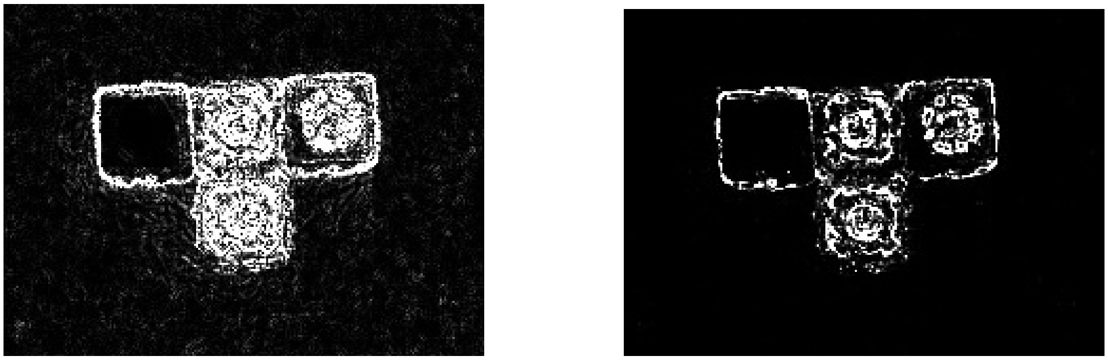
We then attempted to match faces to templates, first based on grayscale images and then based on RGB images. These methods were promising.



(a) Canny Edge Detection

(b) Sobel Edge Detection

Fig. 1. Edge Detectors

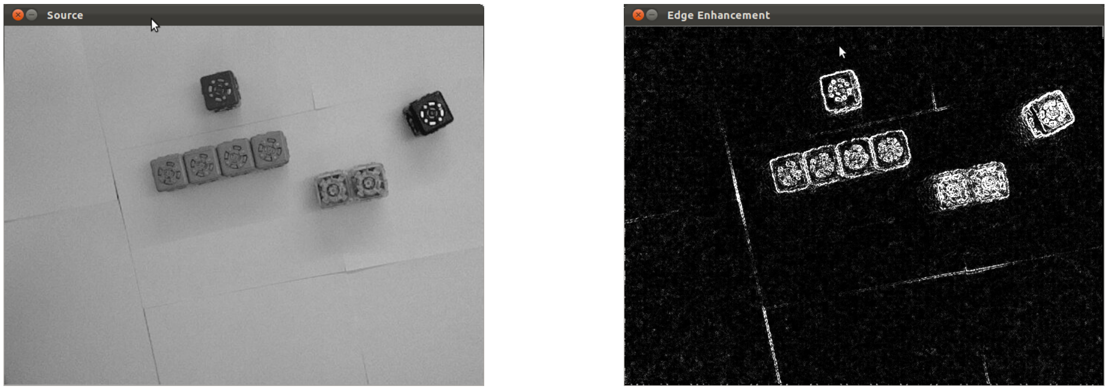


(a) Initial Edge Enhancement Filter

(b) Low Noise Edge Enhancement Filter

Fig. 2. Our Edge Enhancement Filters

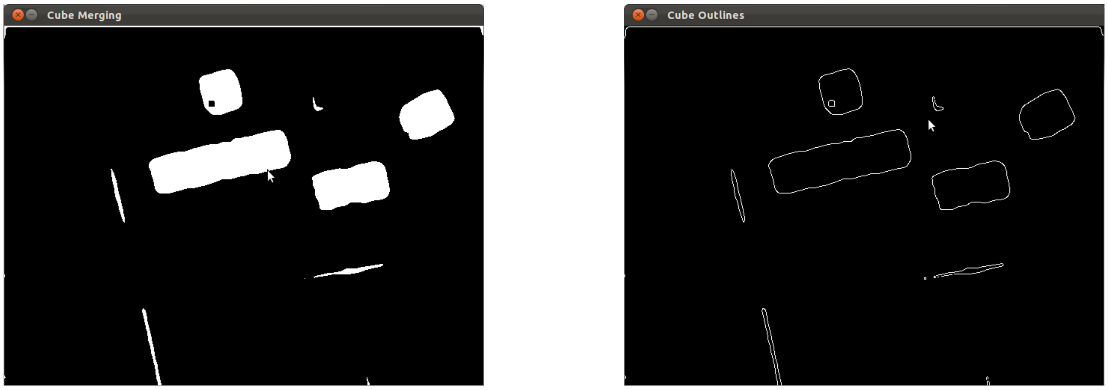
In order to better match the Cubelet faces, specifically the clear faces, we applied a novel form of edge detection/enhancement to the images. Without doing this, the features of the clear Cubelets were not very distinct. We found the Canny edge detector [10] to be too imprecise; a slight adjustment to the upper and lower edge merge thresholds would change the shape of fine detail edges. The Sobel edge detector [11] suffered from a different problem in that the edges that it did detect were very weakly enhanced. In order to improve the edge enhancement, we wrote our own novel filter which is very similar to how Sobel operates. Shown are examples of Sobel and Canny run on our test images, as well as the two versions of our edge enhancement filter.



(a) Original Image

(b) After Edge Enhancement Filter

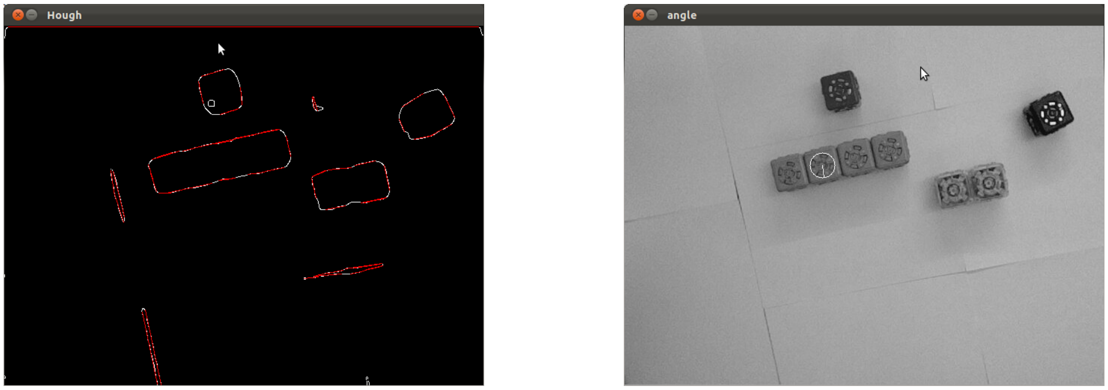
Fig. 3. Rotation Estimation - Initial Processing



(a) After Successive Blurs, Dilations, and Erosions

(b) After Canny Edge Detection

Fig. 4. Rotation Estimation - Cube Outlining



(a) Result of Hough Line Detection

(b) Final Estimation

Fig. 5. Rotation Estimation - Line Detection and Estimation

The first variant of our filter behaves as follows: for each pixel, the value of at pixel becomes the product of the difference between the pixels above and below and the difference between the pixel to the left and the right.

$$f(x,y) = |i(x+1,y) - i(x-1,y)| \times |i(x,y+1) - i(x,y-1)|$$

This filter enhances edge boundaries very strongly, but still suffers from a bit of noise. The second filter reduces the noise seen, but as a side effect it reduces some of the edge enhancement strength.

$$f(x,y) = \frac{1}{c}|i(x+1,y) - i(x,y)| \times |i(x-1,y) - i(x,y)| \times |i(x,y+1) - i(x,y)| \times |i(x,y-1) - i(x,y)|$$

Before performing template matching, we apply these filters to the template image and the target image. This template matching worked well, but did not accurately find matches of the correct color.

In order to handle the possibility of missing a match due to a rotation, we took each template image, rotated it by 15 degrees, and then template matched that rotated version to the target image.

To accomplish the actual template matching, we used the template matching function built into OpenCV, cv.MatchTemplate. The particular match value calculation equation we used, CV_TM_SQDIFF_NORMED, is as follows:

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

Here, $T(x,y)$ refers to a pixel value within the template image at (x,y), and $I(x,y)$ refers to a pixel value in the target image. The output of this function is a grayscale image with pixel values equal to difference metric $R(x,y)$ between the local window and the template. To find the matches in the image, we simply look for the minima within the resulting image.

Once a match was found in the match image, we excluded a circular area around the match from further matches.

C. Rotation Estimation

In order to determine the rotation of each Cubelet, we used the relatively distinct edges of the Cubelet. We started by applying our edge enhancement filter to the test image. We then apply blur, dilate, and erode operations to the resulting image in order to merge the edges of each Cubelet into a single connect component. This supresses the internal features of the Cubelet, such as the metal connectors. We then applied Canny edge detection to the image, resulting in only the outlines of the cubes.

After Canny edge detection has been applied, we used the Hough line detector [12] find the line segments present. The detector returns lists of line segment endpoints, from which we calculate the rotation of each line segment modulo $\frac{\pi}{2}$. For each Cubelet detected in template matching, we determined all the line segments within some radius of the Cubelet center, then averaged the calculated angles for those line segments to obtain the rotation angle for the cube.

As this rotation estimation method relies solely on the ability to detect the outlines of the cube, anything that hinders that would affect the accuracy of this estimation method. Since our environment does not contain clutter, this was not an issue.

D. Color Determination

Once we had a list of prospective Cubelet locations, we found the average color around the center of each match, and compared that to the color averages of each template image, selecting the smallest difference as the matching color. This step is highly dependant upon how well the center of the the Cubelet has been determined; a large enough offset will result in the background or a nearby Cubelet's pixels to be picked up as part of the average color evaluation. We hypothesize that this problem could be alleviated by using a set of color filters to filter out the table and the magnetic tabs.

E. Depth Information

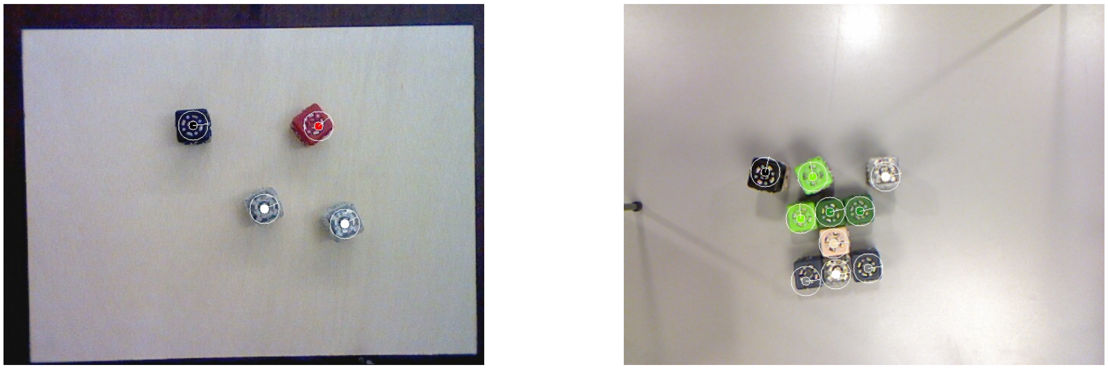
Depth data is collected using the XBox Kinect sensor. Since the depth information matrix is in the same shape the RGB color image information, we need only the (x,y) coordinate of a pixel in the image to determine its corresponding depth value in the 16-bit coupled depth image. Due to noise in the image and time constraints, we did not complete depth estimation.

III. Experiment

To evaluate the quality of our match results, we manually labeled each test image with the following information for each Cubelet: position, color and rotation. Rotation estimation was done rotating the image in an image editing tool until a particular cube's edge was parallel to the x-axis, then marking down the rotation angle required.

IV. Results

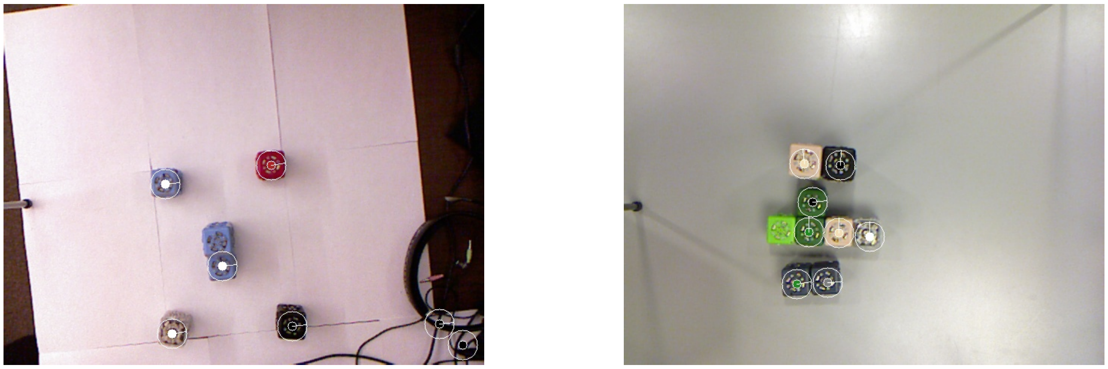
The algorithm was run on three different data sets, each generated at different portions of the development process. The first data set was taken with a a 640x480 webcam, with a limited set of Cubelet types on a wood panel background. The second data set was again taken with the webcam, but with a white-papered background. This test set also used blue-spray-painted Cubelets, which were not used in other tests. The final data set was taken with the Kinect sensor's camera, over a gray table background with a fairly bright light source and with the a set of 9 different colored Cubelets.



(a) From Test Set 1

(b) From Test Set 3

Fig. 6. Successful Detections and Estimations



(a) From Test Set 2

(b) From Test Set 3

Fig. 7. Unsuccessful Detections and Estimations

TABLE I
CUBE LOCALIZATION AND POSE ESTIMATION

	Pixel Distance from Actual	Standard Deviation
Set 1 (14 images)	3.9	5.6
Set 2 (10 images)	4.3	3.6
Set 3 (10 images)	3.1	2.7

(a) Average Offset from Actual Center and Standard Deviation

	Pixel Distance from Actual	Standard Deviation
Set 1 (14 images)	5.0°	6.8°
Set 2 (10 images)	10.3°	12.9°
Set 3 (10 images)	3.3°	3.8°

(b) Average Error in Rotation and Standard Deviation

TABLE II
COLOR DETERMINATION ACCURACY

	Accuracy
Set 1 (14 images)	95%
Set 2 (10 images)	47%
Set 3 (10 images)	84%

TABLE III
DETECTION FAILURE OCCURENCES

	False Positives	Missed Cubes
Set 1 (14 images)	0	0
Set 2 (10 images)	21	5
Set 3 (10 images)	1	6

V. Discussion

In terms of actually detecting the presence of Cubelets, our process does do a fairly good job. The rare cases of a Cubelets being entirely missed in the initial detection stage tended to result from partially occluded cube profiles, or due to a cube's features being washed out by very bright light or rendered indistinct by poor lighting. One particular set of cases to be noted is that of the missed light-green cubes in data set three. Upon closer observation, it seems that when the image is converted to grayscale, the metal connectors and the colored parts of the Cubelet tend to map towards similar grayscale color values. This made the internal features of the Cubelet face less distinct, and caused the Cubelet face detection to fail.

The color determination was not quite as accurate as we would have hoped. This mainly occured with the clear and black cubes, which in dim lighting have somewhat similar colors. A way to address this would be to instead compare color histograms of the match with ones obtained from the Cubelet templates. This would differentiate the black and the clear cubes well, as the black cubes would have a spike in the lower end of the histogram, and the clear cubes would have a more spread out distribution.

The results of the second data set were worse than the results for the other data sets due to several factors. The first issue comes from the blue spraypainted Cubelets. This color was not as consistent and their features were more subdued. This caused them to be occasionally misidentified or not identified at all. The clutter in the scene also posed difficulty for our process. The wires present matched the templates just enough to register as Cubelets, which unfortunately demonstrates that this process is not very robust in less controlled situations.

On some occasions, the rotation estimation failed to give any estimation at all. This was due to a lack of strong edges detected within the proximity of a Cubelet's center point. Thus, anything that could cause an edge to be less distinct could result in a failure of rotation detection. Cubelets that were completely surrounded were completely prone to this, but both surrounded Cubelets and less distinct edge issues could possibly be solved by comparing to Cubelets that were close enough that they were likely attached to the undetermined Cubelet.

A. Further Work

The major limitation of our approach is that it requires a fixed overhead view of the scene. This constraint allowed us to use template matching to find Cubelets easily within the scene, limits its usefulness on a robot that can look around in its environment. Our methods do not work very well if the Cubelets were to be seen from non-normal angle. This change in perspective would require us to have a more thorough method of pose estimation.

In order to address these concerns, we would need to find an object recognition method that is robust against scale, rotation, brightness differences, and affine transformations. As we have already found that SURF is insufficient, we would investigate matching methods using other feature detectors, such as MSER (Maximally-Stable Extremal Region) [13], which is very robust against affine transformations.

Given a set of matching points between a training image and the target image, we could find the transform relating those sets of points, and calculate the pose of those points in the camera space, provided that the camera parameters were known. Many different methods for doing this exist, including POSIT [14], which works on four non-coplanar points.

VI. Conclusion

We believe that the work done has demonstrated that template matching can be an effective method of object detection in the constrained circumstances tested. In designing and testing this method, we have also identified concerns regarding object detection and recognition that can be addressed in further research that would extend the capabilities of this system past its operating constraints.

Acknowledgment

The authors would like to thank Professor Nikolaus Correll and the CSCI 7000 class.

References

[1] G. Loy and J. Eklundh, *Detecting Symmetry and Symmetric Constellations of Features* ECCV 2006, Part II, LNCS 3952, pp. 508-521, 2006.

[2] Eric Schweikardt. 2010. Modular robotics studio. In Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction (TEI '11). ACM, New York, NY, USA, 353-356. DOI=10.1145/1935701.1935784 http://doi.acm.org/10.1145/1935701.1935784

[3] http://correll.cs.colorado.edu/clam/

[4] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng. "ROS: an open-source Robot Operating System". http://www.ros.org/

[5] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. 2. pp. 11501157. doi:10.1109/ICCV.1999.790410.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008

[7] http://www.xbox.com/kinect

[8] http://opencv.willowgarage.com/

[9] http://75.98.78.94/default.aspx

[10] Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679698, 1986.

[11] K. Engel (2006). Real-time volume graphics,. pp. 112114.

[12] http://en.wikipedia.org/wiki/Hough_transform

[13] Donoser, M. and Bischof, H. Efficient Maximally Stable Extremal Region (MSER) Tracking CVPR, 2006.

[14] Daniel DeMenthon and Larry S. Davis, "Model-Based Object Pose in 25 Lines of Code", International Journal of Computer Vision, 15, pp. 123-141, June 1995.