# 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 such 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. 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, a small robotic arm intended for research, with a suitable grasper. We also intend to use ROS (Robot Operating System) 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

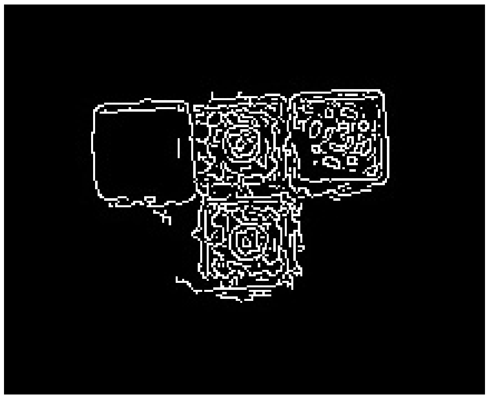
The problem of providing the desired ontology is largely one of object detection, a well-known problem in computer vision. Object detection methods include feature detection methods such as SIFT and SURF as well as template matching, segmentation and classification of segments, and flexible template 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. Matters and Materials

Our experimental setup consists of a Xbox Kinect camera mounted 65cm above the work surface. This distance is 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 is covered in white paper. The lighting is not controlled but is normal office lighting.

We generated three sets of 10-14 images each, under the same setup and lighting conditions. Each image takes a picture of an arrangement of Cubelets. The Cubelet arrangements are randomly determined by us. The first set was used as a training set to train the vision subsystem. The second set was used as a test set to generate the experimental results.

Our software is written in Python and uses ROS with OpenCV and openni_kinect.


(a) Canny Edge Detection

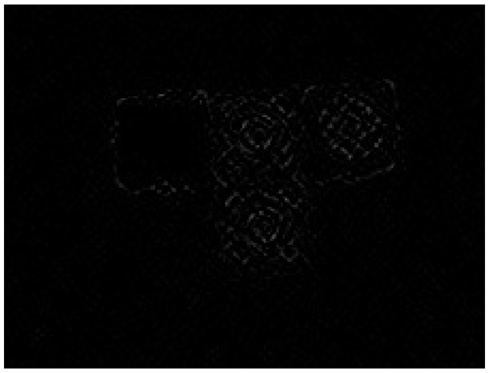
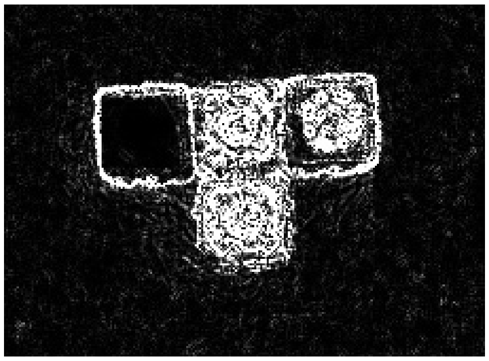

(b) Sobel Edge Detection

Fig. 1. Edge Detectors


(a) Initial Edge Enhancement Filter

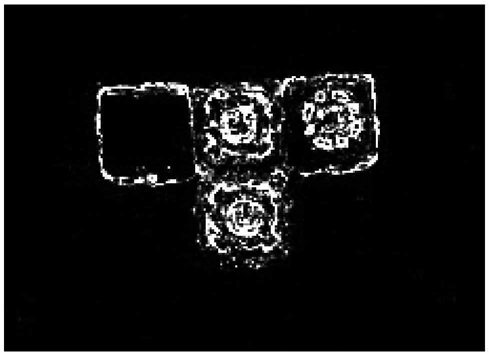

(b) Low Noise Edge Enhancement Filter

Fig. 2. Our Edge Enhancement Filters

## A. General Methods

To manage the information related to the templates, we've created a single file with a variety of template images, scaled to approximately the same size. In addition to this, we have an xml file marking the (x,y) locations of each template cube within the larger image, as well as the idea template radius for that cube. For the actual templates, we use the region around the circlular portion of the cube face.

## B. Edge Enhancement Filter

In order to better match the cubes, specifically the clear cubes, we found it necesary to apply a form of edge detection/enhancement to the images. Without doing this, the features of the clear cubes would not be very distinct, making matching of any kind very difficult. We found the Canny edge detector to be too imprecise; a slight adjustment to the upper and lower edge merge thresholds would change the shape of fine detail edges. Sobel 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 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.

The first variant of the 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.

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

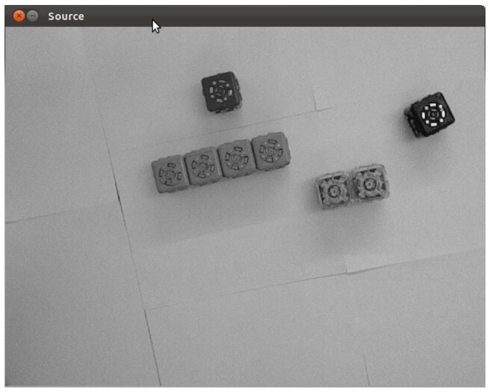
Before performing template matching, we apply these filters to the template image and the target image. Although this removes the color information, it does allow for the template matching process to detect all cubelets indiscriminately.

## C. Rotational Template Matching

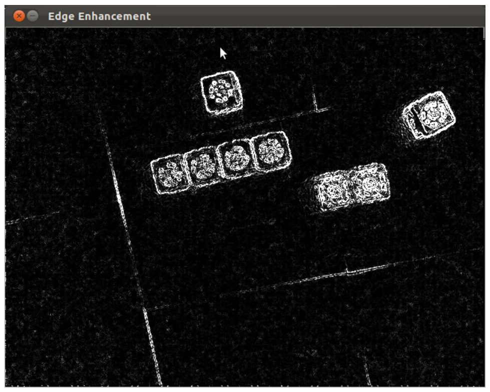
In order to handle the possibility of missing a match due to a rotation, we take each template image, rotate it by 15 degrees, and then template match that rotated version to the target image. To accomplish the actual template matching, we are using the template matching function built into opencv, cv.MatchTemplate. The particular match value calculation equation 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.

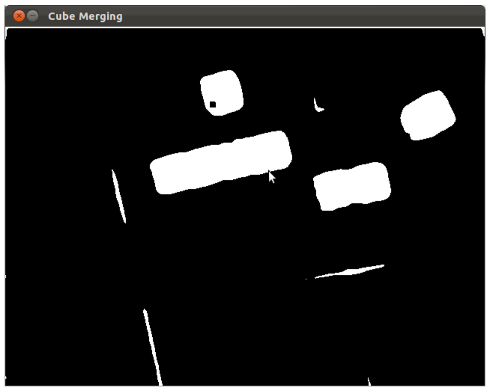


(a) Original Image

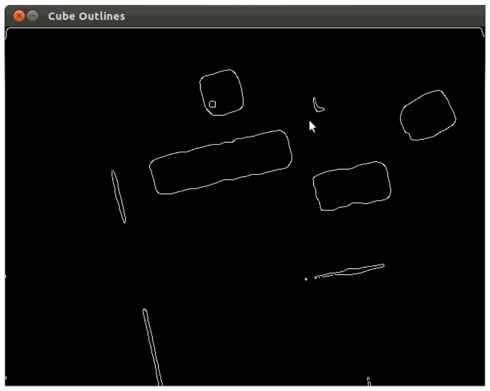


(b) After Edge Enhancement Filter

Fig. 3. Rotation Estimation - Initial Processing

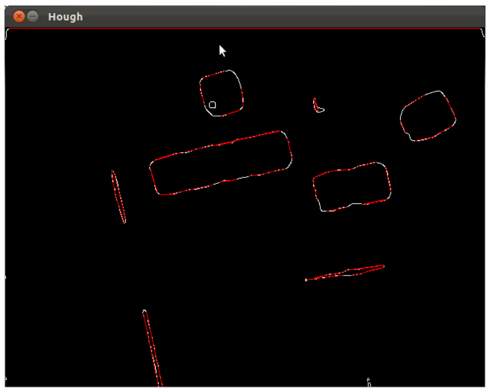


(a) After Successive Blurs, Dilations, and Erosions

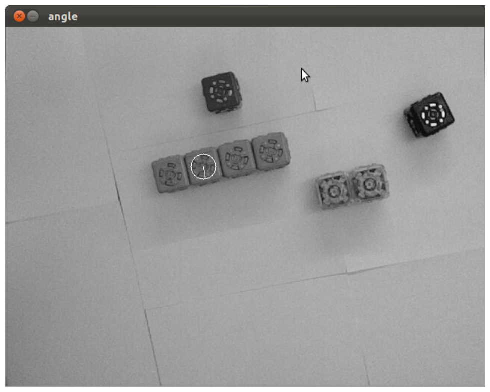


(b) After Canny Edge Detection

Fig. 4. Rotation Estimation - Cube Outlining



(a) Result of Hough Line Detection



(b) Final Estimation

Fig. 5. Rotation Estimation - Line Detection and Estimation

### D. Rotation Estimation

In order to determine the rotation of each cubes, we find try to utilize the relatively distinct edges of the cube. We once again start by applying our edge enhancement filter to the test image. We then blur, dilate, and erode the resulting image in order to merge the edges of each cube into a single connect component. This is to suppress the internal features of the cube, such as the metal connectors, so that when we apply Canny edge detection to the image, we only get the outlines of the cubes.

After Canny edge detection has been applied, we use OpenCV's hough line detector 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 detected cube, we determine all the line segments within some radius of the cube center, then average the calculated angles for those line segments, which is assigned as 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 would hinder that would severely affect the accuracy of this estimation method. In our constrained, relatively clutter free environment however, this should not be a severe issue.

### E. Color Determination

Once we have a list of prospective cube locations, we find the average color around the center of each match, and compare 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 cube has been determined; a large enough offset will result in the background or a nearby cube's pixels to be picked up as part of the average color evaluation.

### F. 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.

### G. Match Evaluation

To evaluate the quality of matchings and rotation estimations, we manually go through each test image and mark down the center of each cube, the color of each cube, and an estimation of cube's rotation. This was done by 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.

TABLE I
CUBE CENTER LOCALIZATION

| | 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 |

TABLE II
COLOR DETERMINATION ACCURACY

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

TABLE III
CUBE ROTATION ESTIMATION

| | 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° |

TABLE IV
FAILURES

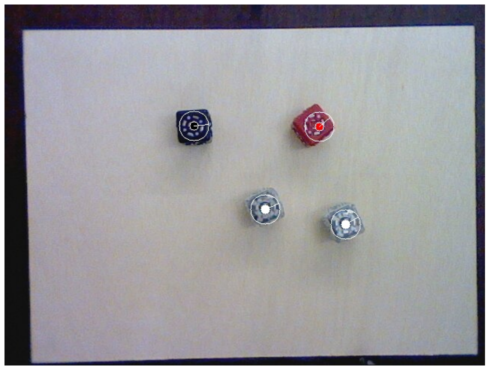
| | False Positives | Missed Cubes |
|---|---|---|
| Set 1 (14 images) | 0 | 0 |
| Set 2 (10 images) | 21 | 5 |
| Set 3 (10 images) | 1 | 6 |

## III. 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 webcam, with the a limited set of cube types sitting on a wood panel background. The second data set was again taken with a webcam, but with the cubes sitting on a white-papered background. This test set also used blue-spray-painted cubelets, as the other cubelets were unavailable during the testing period. The final data set was taken with a kinect sensor's camera, over a clean table background with a fairly bright light source, with the full set of colored cubelets.

## IV. DISCUSSION

In terms of actually detecting the presence of cubes, our process does do a fairly good job. The rare cases of a cube 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 cube tend to map towards similar grayscale color values. This made the internal features of the cube's side less


(a) From Test Set 1

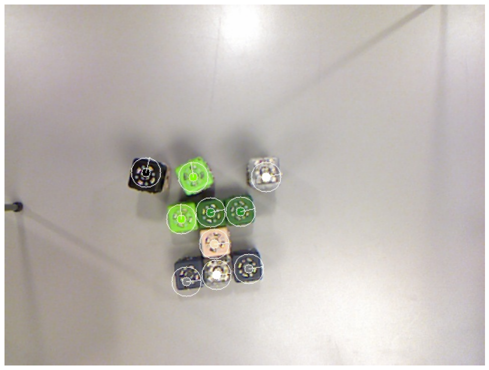
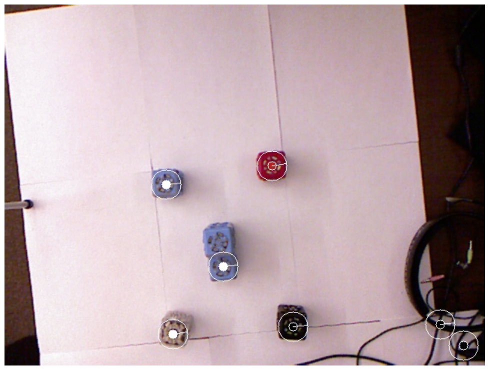

(b) From Test Set 3

Fig. 6. Successful Detections and Estimations


(a) From Test Set 1

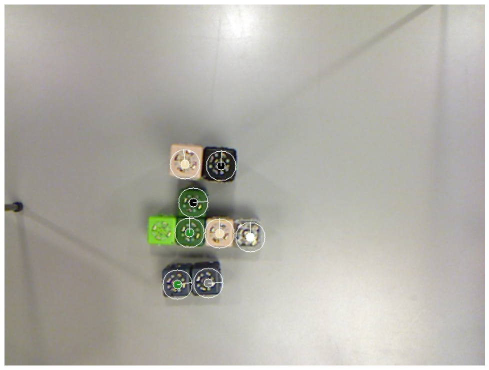

(b) From Test Set 3

Fig. 7. Unsuccessful Detections and Estimations

distinct, and caused the cube 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 a test image and a cube template. 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 particularly worse than the rest, due to several factors. The first issue comes from the blue cubes, which were spray painted blue for the project they were borrowed from. As a consequence, their colors were 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 cubes, 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 cube's center point. Thus, anything that could cause an edge to be less distinct could result in a failure of rotation detection.

## 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 cubes 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), 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, which works on four non-coplanar points.

## V. 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.

### *A. Subsection Heading Here*

Subsection text here.

*1) Subsubsection Heading Here:* Subsubsection text here.

## Acknowledgment

The authors would like to thank...

## References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.