Visual Code Marker Detection

Brian Percival (05335661)

Stanford EE 368 Final Project
bperci@stanford.edu

I. INTRODUCTION

A. *Description of marker*

The marker has certain characteristics as described in the project assignment. As seen in

, of particular importance are the guide markers that assist in both identifying markers as well as properly orienting the marker for reading. The two fixed guide bars at the bottom and right side of the marker are what are sought in the first step of the algorithm. The fixed dots in the corners are then used to verify that the candidates are in fact markers. Finally, the fixed guide markers are then used to read the marker data by providing an origin and a pair of basis vectors from which the data can be read.

B. *Algorithmoverview*

The overall approach taken breaks the task into four major components:

- Scanning the image to locate candidates for the "L" shape formed by the two fixed guide bars at the bottom and right of the marker
- Inspecting candidates and determining if they conform to the marker morphology rules
- Reading the marker data
- Eliminating candidates that likely represent duplicates of the marker found.

Although the original images are color images, the approach performs all analysis on a gray scale version of the image, as computed according to the NTSC standard.

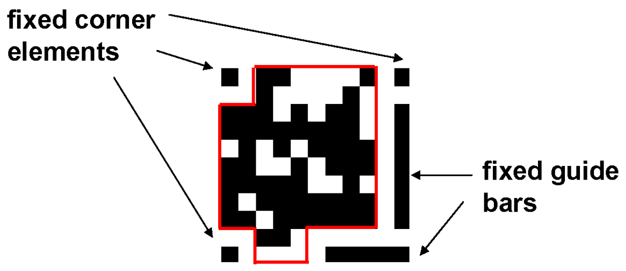

Fig. 1 Marker morphology.

In the first step, the algorithm scans across the image, inspecting one subset of the image at a time, attempting to identify locations where parallel edges intersect. The goal is to find likely locations of the "L" formed by the two fixed guide bars in the marker. Edge detection is performed using the Prewitt method to create a binary representation of the sub-image, with only pixels lying along an edge having a value of 1. The Hough transform is then applied to this binary image to identify edge lines. When a pair of line segments are parallel and lie next to each other, they are considered a pair. When two pairs intersect at an "L", and are spaced similarly apart, they are considered a candidate location of a marker.

In the second step, candidates that were identified during the scanning process are evaluated to determine if they actually represent a marker. The algorithm verifies that black bars exist at the locations indicated by the parallel line segments, that single dots exist at the expected corners of the marker, and that the bars and end dots are separated from other black regions by white space.

In the third step, if a candidate marker has passed all tests in the second step, then it is considered a valid marker and the data are read from the marker. Using the two guide dots along the edges containing the guide bars as well as the centroids of the guide bars themselves, a pair of basis vectors are calculated to be used in determining the image location of each bit of marker data.

Finally, once the marker has been read, any remaining marker candidates that are located within a small radius of the previous marker, or are located within the area covered by the marker region are removed from consideration.

II. DETAILED METHOD

A. *Scanning the image to locate "L" shape*

The first step in locating Ls is to determine the size of the scanning window. This was selected as a fixed fraction of the overall image size. From training data, windows that were 20% of the original image were an appropriate balance. Larger images were likely to allow long lines to dominate the relatively short lines represented in the guide bars. Smaller images would either take too much processing time, or would not include enough of the marker for detection to be possible.

The scanning window is then moved across and down the original image so that the entire image is inspected by the scanning window. Adjacent scanning windows actually overlapped to avoid unfortunate alignment of markers and edges of scanning windows. Again, to balance accuracy and processing time, a fraction of the scanning window size was selected empirically from experiments with the training data. The algorithm takes steps that are 25% of the full dimensions of the scanning window. This ensures that most of the image is inspected multiple times by different scanning windows.

For each sub-image captured inside the scanning window, edge detection is performed. The Prewitt method of estimating the gradient at each pixel is performed. Any pixels at which the magnitude of the gradient is above a threshold (determined separately for each sub-image) are set to 1; all others are set to 0. This results in a binary image with only prominent edges highlighted.

This binary image is then analyzed via a Hough transform to identify lines present in the edges. Any lines identified by the Hough transform that are formed by near-contiguous edge pixels are converted into line segments with fixed endpoints. A visualization of a sample result of this process is in Fig. 2.

These line segments are analyzed and grouped by their angle of orientation. Any two lines that have nearly identical orientation (within 2 degrees), have significant overlap, and have a large ratio of length to distance separating the two lines are considered parallel line segments. All possible combinations of parallel line segments are stored in an array.

These parallel line segments are then compared with other parallel line segments. If a set of two line segment pairs have similar spacing between them, have at least a 60 degree angle between the two pairs, and have at least 3 out of all 4 line segments terminating near the same point, and this point of intersection coincides with a black pixel in the original image, then the two pairs of parallel line segments are considered a candidate for being an "L" formed by the two fixed guide bars in the marker. Fig. 3 shows visualization of several examples of this analysis.

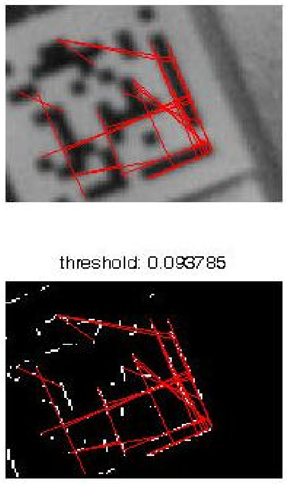

Fig. 2 Edge and line detection. Above: grayscale image with lines found via Hough transform superimposed. Below: binary image in which white areas represent edge pixels, with same lines from above superimposed. As can be seen, there are many candidate lines that include two lines on either side of each the bottom and right bar, indicating a possible candidate.

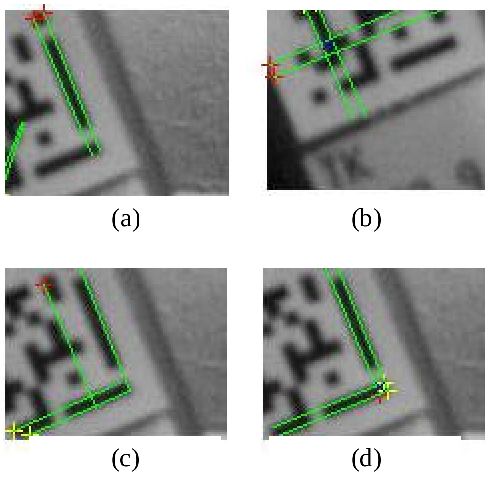

Fig. 3 Analyzing pairs of parallel lines. Green line segments are the detected lines. Red and yellow crosses indicate endpoints. The two pairs of parallel line segments must be separated by an appropriate angle, have roughly equal spacing between the two lines, and must terminate at an L-shaped intersection. (a) Angle between pairs is too small. (b) Pairs do not intersect at an "L". (c) Separation between the line segments is too different in the two pairs. (d) A valid candidate, passing all tests.

Fig. 4 shows an image with all located candidate marker locations, including the candidate intersection and associated line segments that indicate guide bar locations.

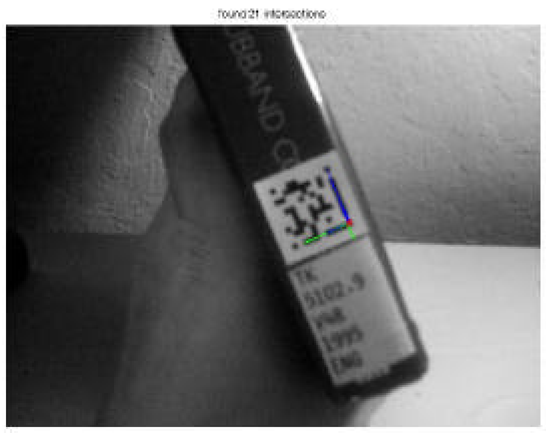

Fig. 4 Full image with candidate marker locations highlighted. Red 'x'es represent possible "L" intersection locations. Green lines indicate possible locations of the bottom guide bar. Blue lines indicate possible locations of the right guide bar.

B. *Inspecting candidates*

Once all candidates for “L” shapes are found in the image, they are evaluated to determine if they actually do represent a marker. For each candidate considered, a sub-image is created from the original gray scale image that includes the estimated region of the marker. The dimensions of the marker are estimated from the orientation of the bars in the L and from the separation between the parallel lines that form the bars. Since the separation between the edges is roughly the width of each bit in the marker, this separation yields a good approximation to the width of each bit in the marker.

The sub-image is then converted to a binary image by applying a threshold value. All pixels whose grayscale intensity are above the threshold are converted to 1s. All other pixels are converted to 0s. The threshold is computed by taking the mean of the two extrema of intensity in the image, 0.5 * (max_pixel_value + min_pixel_value). Because the marker is composed of black and white, it is very likely that the two values are on either side of this threshold. In fact, in many cases, the maximum and minimum values are located within the marker itself (though this is of course not necessary). The image is then logically inverted so that the dark regions of the marker are now represented by 1s for region analysis.

The binary image is then analyzed to determine the regions it contains. The regions are then analyzed to determine several properties that are used in later analysis: eccentricity, centroid location, length of major and minor axis of ellipsoid with same moments.

The next step is to locate the guide bars. The bottom guide bar should coincide with the vertex of the L formed by the guide bars. So the intersection of candidate line segments is matched with the region that contains that pixel. If the region is not elongated, and is not in the expected orientation, the candidate is rejected. The guide bar on the right side of the marker is located by matching a point along the direction of the parallel lines forming the right side of the marker candidate. This region is also verified to be elongated and in the proper orientation.

If both regions appear to be bar-like, the algorithm then recalibrates the distances and orientations of the bits in the marker by using the two fixed bar regions. Using their centroids and orientations, basis vectors are calculated to convert from marker coordinates to image coordinates. These basis vectors use the “L” intersection as their origin. The magnitudes of the basis vectors are determined by the spacing between the intersection at the “L” and the centroids of the regions. The bottom bar’s centroid should be 2 units from the intersection at the “L” of the marker. The right bar’s centroid should be 5 units from the intersection at the “L” of the marker.

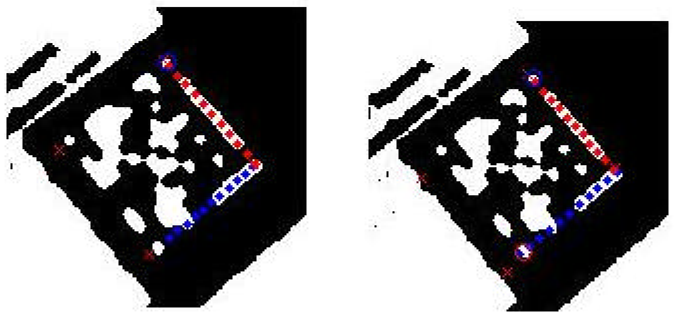
Fig. 5 Verifying marker by finding fixed corner elements. Blue and red dots indicate integral multiples of the basis vectors. Hollow red and blue circles indicate located fixed corner elements. Left: The blue dots representing the “bottom” edge of the marker do not align properly with the elements, in particular, the bottom-most corner element is not found. Right: Both corner elements are found. The marker was correctly identified and read.

With the updated basis vectors, the dots at the end of the two guide bar edges are located. If no region exists at the locations indicated by the recalibration, the candidate is rejected. If either region at these locations has too high eccentricity (not “dot-like” enough), the candidate is rejected. Fig. 5 shows an example of the algorithm not finding the end dots in one candidate, but then successfully finding them with a subsequent candidate. The blue and red diamonds indicate integral multiples of the basis vector, depicting the horizontal and vertical positions of each row and column of bits. The colored circles at the end dots indicate that both end dots have been successfully found.

If the end dots are found, the basis vectors are again recalibrated, this time using the centroids of the bar and the corresponding end point to determine the magnitude of the basis vectors. Because these are further separated than the original two points used to form basis vectors, the updated spacing is more accurate than the original. The updated basis vectors are then used to verify that the white spaces separating the fixed guide bits along the bottom edge and right edge of the marker are in fact white.

C. *Reading marker data*

If all above tests pass, then the candidate is considered an actual marker. The basis vectors and the location of the “L” are preserved for reading the marker data. Each location of the 83 bits is converted from marker coordinates to image coordinates by adding the appropriate linear combination of the two basis vectors.

D. *Duplicate detection and exclusion*

Once the marker is read, the polyhedron containing the area of the marker is computed by intersecting the halfspaces created by each of the four sides of the marker. This polyhedron is stored in the form of a 4x2 matrix and a 4x1 vector for efficient evaluation of candidate “L” intersection locations. If any further candidates lie within the polygon, they are ignored. Further, if any candidate “L” intersections lie within a small radius of the current marker’s “L” intersection, they are also ignored. This prevents duplication and greatly decreases processing time.

III. Results

The following table summarizes the performance on the 12 training images provided. The algorithm encountered zero false positives, but did fail to find several markers. For the markers that were found, high percentages of the bits were read correctly, with more errors in the markers that had more pronounced foreshortening due to the perspective of the image.

TABLE I. Results from Training Images

Figure	Ground Truth # Markers	# Markers Found	False Positives	Bits Correctly read	Accuracy
Training_1.jpg	1	1	0	83	100.0%
Training_2.jpg	2	2	0	165	99.4%
Training_3.jpg	3	3	0	246	98.8%
Training_4.jpg	1	1	0	81	97.6%
Training_5.jpg	3	3	0	231	92.8%
Training_6.jpg	1	1	0	0	0.0%
Training_7.jpg	2	2	0	166	100.0%
Training_8.jpg	1	0	0	0	0.0%
Training_9.jpg	3	3	0	249	100.0%
Training_10.jpg	3	2	0	163	98.2%
Training_11.jpg	1	0	0	0	0.0%
Training_12.jpg	2	1	0	81	97.6%
TOTAL:	**23**	**18**	**0**	**1465**	**76.7%**