

# Image Processing Algorithm for Detection of Two-Dimensional Visual Code Markers

Andrew B. Graham

**Abstract**—Emerging technologies such as cell phone cameras allows the possibility of visual code marker detection from a relatively poor resolution image. An algorithm has been developed that finds image areas with potential markers, and then looks for specific features of such markers to determine if a marker has been found. The bits in the marker can then be determined. The algorithm has been designed such that markers of nearly any size and orientation can be detected.

## I. INTRODUCTION

WHILE barcodes have found uses in a wide range of applications such as commercial businesses and inventory operations, the required laser scanner makes them impractical for use by everyday consumers. With the emergence of small cameras in cell phones, the tool necessary to read code markers is now available to a large portion of consumers. Making use of the camera's ability to capture two-dimensional data, a visual code marker has been developed that is capable of communicating 83 separate "bits" of information. An example of such a visual code marker is shown in Figure 1. Note that the marker includes two guide bars and three corner elements which are fixed references to determine the orientation of the marker. Interior points are used to store the "bits" of the marker.

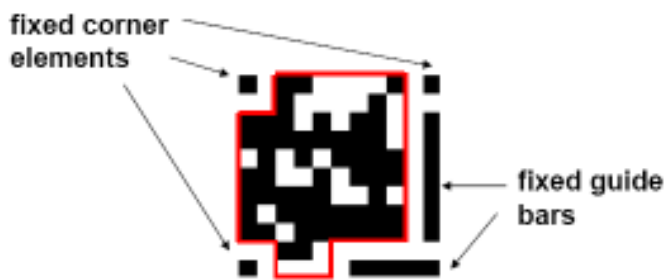


Fig. 1. Visual code marker example showing fixed orientation elements [1].

To read such code markers, algorithms must be able to detect all code markers in a particular image while rejecting other image artifacts that may appear similar. Additionally, algorithms must be able to determine the correct information stored in the code marker regardless of the marker's

orientation in the image. What follows is the description of an image processing algorithm developed to meet these requirements, followed by sample results that come from the intermediate steps of the algorithm when applied to various test images.

## II. DESCRIPTION OF IMAGE PROCESSING ALGORITHM

### A. Key Features

In determining the best way to detect the markers, it is important to first consider the key features that are unknown. These include, but are not limited to, the number of code markers in the image, the size of the code markers, the orientation of the code markers, and the location of the code markers. Additionally, the markers have certain features that can be exploited to aid in their detection. Specifically, the fixed elements, square shape, and black and white color will help distinguish the code markers from other image artifacts. By understanding each of these items, an algorithm can successfully detect and read the existing markers.

### B. Fundamental Detection Steps

The most basic property of the code markers, especially in the context of a color image, is that they are black and white. The clear first step in their detection is rejecting portions of the original image based on this fact.

In an effort to reduce the detection time and required resources, it will be advantageous to next find potential code markers within the image and only consider these portions of the original image. Note that a crude estimate of the marker size is necessary to find potential markers. Finding these sub-regions of the image serves as a first step toward determining the marker's size and location.

Having reduced the image to a series of smaller interrogation regions containing the code markers, the orientation of dominant edges can be determined. By requiring that at least two detected edges are roughly perpendicular, this step not only gives some information about the orientation of the markers, but it acts as a first attempt at rejecting false positives as well.

Making use of this basic orientation and the square shape of the markers, the next step is to determine the precise size and center location of the marker. Upon doing so, the sub-region can further be reduced in size to a region containing only the marker. The fixed orientation elements can then be used to

reject any remaining false positives before determining the precise orientation of actual markers. The bit information in the actual markers can then be determined.

A flow chart of the algorithm is shown in Figure 2.

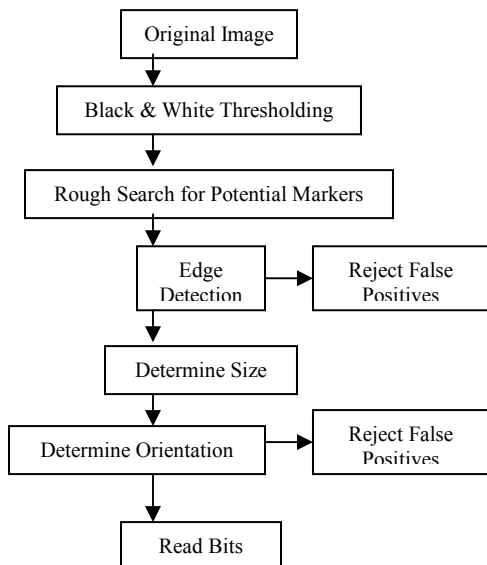


Fig. 2. Flow chart of the basic image processing algorithm.

### C. Black & White Thresholding

Because the white portions of the markers can appear quite dark or discolored depending on the lighting for the picture, simply making use of the three color components to determine “white” is not very effective. Instead, the 8-bit RGB color image was converted to NTSC values, but only the luminance information was kept. This intensity,  $I$ , was calculated from

$$I = 0.299R + 0.587G + 0.114B \quad (1)$$

where  $R$ ,  $G$ , and  $B$  are the red, green, and blue color components. Because black portions of the image are not as affected by lighting as the white portions, this intensity image was converted to a binary form, ensuring even poorly lit white regions would separate themselves from the darker regions. The threshold value was optimized using a series of 12 test images. Following this, the binary image was negated so that bits in the code marker were tagged as high valued and empty spaces as low valued. Finally, connected areas larger than a certain threshold were eliminated. This threshold value was set so that markers would not be affected unless they occupied a majority of the image and were fully connected.

### D. Search for Potential Code Markers

Because each marker varies in the area occupied by bits, a generic marker shape is hard to determine. In an attempt to make all markers more uniform, the binary image was dilated and holes were filled to achieve a marker more closely resembling a solid square. Because the orientation is unknown, a circular structuring element was used to provide the best results for all cases. Following this, extremely large

areas were again eliminated to remove any artifacts that may have grown in connectedness due to the dilation.

To detect potential code markers, a form of template matching was implemented which correlated a representative marker shape with the image. While a solid square would be the ideal shape, correlations would have to be performed iteratively over a range of orientations from  $0^\circ$  to  $90^\circ$ . In an effort to reduce the detection time of the overall scheme, a circular template was instead chosen to eliminate the variability associated with the orientation. To account for the unknown size of the marker, the circular template was convolved for incrementally increasing radii. For each iteration, areas of regional maxima were detected. These regions were then averaged to roughly determine the center of the peaks. While it is clear this method would indicate circular objects in the image as potential markers, it is also very effective in indicating true markers as potential markers. This was confirmed in the 12 test images where every true code marker was detected using this method.

### E. Edge and Dominant Orientation Detection

With a first estimate of the center location of potential code markers, smaller sub-regions of the un-dilated, binary image were pulled from the larger image. The size of these regions was determined using the relative spread of the correlation peaks with a healthy margin of safety. Recall that these smaller images should have roughly centered potential markers in binary form.

To improve the process of determining the dominant orientations present in the sub-region, an edge detection algorithm was first employed. This was accomplished using the two Roberts filters:

$$R_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad R_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

By convolving each of these filters with the sub-region and combining the results, a new image results that is merely the outline of all objects in the original sub-region.

Using these outlines to find the orientations of the two guide bars (the dominant orientation), the  $360^\circ$  of possible marker rotations is reduced to four specific orientations. Clearly, this information reduces the remaining processing. To do this, a Radon transform was applied to this new sub-region over a range of  $0^\circ$  to  $180^\circ$ . Peaks were then detected in the result to determine the dominant orientations present in this portion of the image. Because the number of peaks ranged from a single peak to several, an opportunity was available to reject sub-regions that were incorrectly indicated as potential markers from previous processing. In the simplest case where only a single peak was detected, the sub-region could immediately be dismissed since the two guide bars should always give two orientation peaks. In the case of more than one peak, a condition was set requiring that two of the peaks have a difference of  $90^\circ \pm 10^\circ$  or, again, that sub-region was rejected

as not containing a code marker. The seemingly large tolerance in the separation of the peaks allows for error in the Radon transform peak detection as well as variation in the actual angle between the fixed guide bars. It allows for the possibility that the camera may not have been directly perpendicular to the code marker when the image was taken, thus potentially reducing or enlarging the separation angle. If the condition was satisfied, the associated sub-region was passed on to the next stage in the algorithm. If the condition was satisfied for more than one dominant orientation, the sub-region was passed on as a separate potential marker for each case.

#### F. Marker Size Determination

Whereas the first attempts at determining potential marker locations made use of a circular template, the newly acquired information regarding the dominant orientation promotes the use of a template more closely resembling the actual code marker. As before, however, the template matching technique still requires iterating through varying template sizes. But the size is, after all, the next piece of information required.

Early attempts at determining the size were performed in conjunction with attempts to determine the precise orientation among the four remaining possibilities. In doing this, several different templates were created which contained only the fixed elements of the marker. By varying the size and orientation of the template while normalizing appropriately, it was hoped that a peak would emerge that correctly characterized the marker. This method was unsuccessful, however, as the interior elements of the marker often provided strong peaks for templates significantly smaller than the actual size. This led to the separation of the two problems of determining the precise size and determining the orientation.

Returning to the idea of using a template immune to the precise orientation of the marker, a template was created consisting of a solid square rotated according to the dominant orientation. This template was then correlated with the appropriate sub-region of the dilated and filled binary image. Noting that the output of a template exactly the correct size and that of a template only slightly larger would be the same due to the required empty space surrounding the marker (that is, at the point where the output is limited by the size of the marker, not the template), the size of the template was increased in small increments (2 pixels) until the peak output was unchanged. The precise size of the marker could then be determined by knowing the size of the structuring element used to perform the dilation and the size of the template just before the output stopped changing. Additionally, the location of the correlation peak corresponded to the precise location of the marker's center. In the case that the correlation peak did not level off, the sub-region was rejected as not containing a code marker.

#### G. Determination of the Precise Orientation

Knowing the size, center, and dominant orientation of the potential marker, it is fairly simple to make the final

determination of the correct orientation by considering the fixed elements. To simplify the problem, the sub-region was rotated using a bilinear interpolation scheme so that the code marker was square with the x-y coordinate system. Additionally, the area outside of the code marker's known domain was ignored. It was then determined whether or not the fixed guide bars were oriented as shown in Figure 1 by simply averaging small elements in those locations and ensuring none were identically zero (recall that the bits and fixed elements are high valued). The image was then rotated 90° and checked again. If a single pair of guide bars was found, the precise orientation of the code marker was known. If guide bars were detected for more than one orientation or if no guide bars were detected, the image was rejected as not being a code marker.

#### H. Reading the Code marker

Upon confirmation of the guide bars the appropriate bits were probed to determine their value and stored to an array. This was done by simply averaging a small region located at the center of each bit (as determined according to the marker's size) and assigning it a 0 or a 1 accordingly. Lastly, all of the various orientation and sub-region domain information was used to determine the coordinates of the upper left fixed corner element relative to the global coordinates of the original image.

### III. SAMPLE RESULTS FROM TEST IMAGES

Figure 3 shows the intermediate results for part of a sample image obtained from the black and white thresholding step. Note how, in Figure 3(c), the code marker is well preserved.

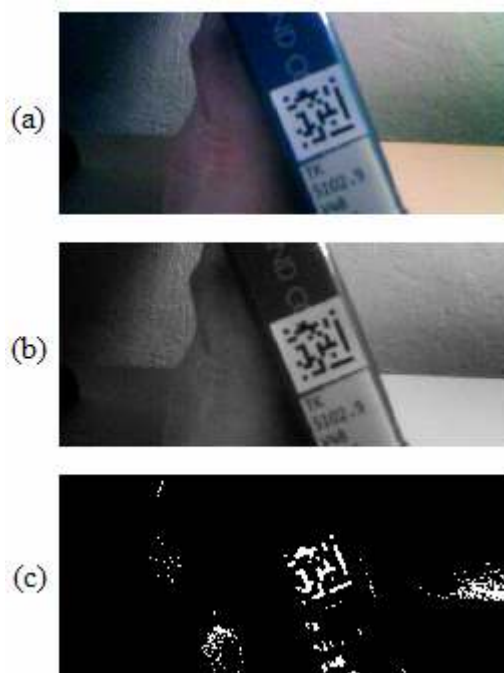


Fig. 3. (a) Original image; (b) NTSC luminance image; (c) Binary thresholded and negated image.

Figure 4 shows the averaged result of template matching on the dilated and filled binary image with a circular template and Figure 5 shows the smaller interrogation regions that were pulled from the binary image shown in Figure 3(c). The sub-regions shown in Figure 5 are enlarged from the scale of Figure 3. Regions of correlation peaks were thresholded before averaging was performed. As can be seen, two of the three initial matches for potential code markers are not actually code markers. Further processing will eventually reject both.



Fig. 4. Results of template matching with circular template.



Fig. 5. Sub-regions containing potential markers after first detection attempt.

Figure 6 shows the resulting sub-region containing the actual code marker after performing the described edge detection routine.

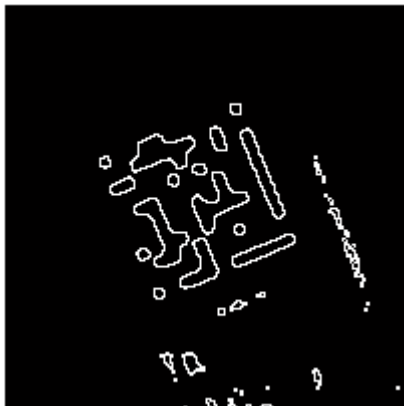


Fig. 6. Edge detection result.

After performing the edge detection, the Radon transform was used to determine the dominant orientation. The resulting values are shown in Figure 7. It is from this data that peaks were found and the perpendicular assumption was checked. This step led to the rejection of the two false positives previously found for this test image.

Figures 8(a) and 8(b) show the portion of the binary image pulled based on the already determined size and the portion rotated to square and trimmed to size, respectively. Notice that, in this case, the starting point for determining the precise orientation was already correct.

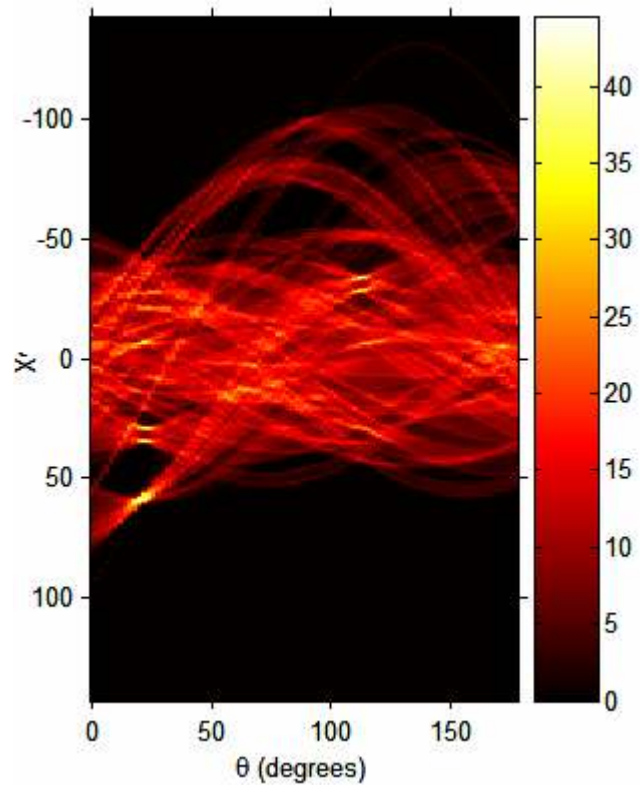


Fig. 7. Radon transform results showing peaks for certain orientations.



Fig. 8. (a) Detected portion as it appears in the binary image; (b) Portion rotated to square and trimmed of excess area.

#### IV. DISCUSSION OF RESULTS

While the described algorithm was very successful in detecting and reading the test image shown in Figure 1, weaknesses in the algorithm were made evident from other test images. Images that leave numerous artifacts after the black and white thresholding steps tend to create problems when looking for the dominant orientation. If strong peaks exist for other angles, the peaks from the guide bars can sometimes fall below the threshold needed to find both. While attempts were made to remedy this by lowering the threshold and improving the peak detection routine, it most often led to an increase in false positives.

Another weakness of the algorithm shows up when markers are very near other peaks in the initial binary image. In some cases, the center of the peak was skewed toward the other artifacts resulting in a sub-region which only had a partial marker.

The final and most difficult problem to overcome is the problem of the image having been taken from an extreme angle. This can lead to two possible problems. First, the marker can appear rectangular rather than square. While the marker typically makes it through the processing, the determined size of the marker is larger in one direction than another. This leads to errors when attempting to read the bits of the code marker. The second problem leads to the same result, but begins when the marker is more of a diamond shape than a square. Attempts to account for these possibilities were not ineffective, but dramatically increased the computation time.

The areas that would lead to the most improvement would involve improved two-dimensional peak detection, and faster, more efficient means of checking for skewed markers.

These weaknesses aside, the algorithm is fairly robust in its ability to detect correct markers amidst other image artifacts that would appear to give problems. Black and white, square shaped patterns of similar size to the markers make it through early portions of the algorithm, but were rejected under at least one of the previously discussed safeguards. From the 12 test images, not a single false positive was found.

## V. CONCLUSION

With the intention of detecting and reading two-dimensional visual code markers, an image processing algorithm has been developed which seeks to detect possible code markers and, through a series of steps, pass or reject these possibilities. As part of these steps, information necessary to eventually characterize and read the markers is determined and used to more efficiently check those possibilities that remain. While the algorithm does have its weaknesses, it has proven successful at detecting and correctly reading visual code markers from a varied assortment of test images.

## REFERENCES

- [1] M. Rohs, "Real-World Interaction with Camera Phones," in *Lecture Notes in Computer Science*, vol. 3598, Springer-Berlin/Heidelberg, 2005, pp. 74–89.