Localization and Segmentation of A 2D High Capacity Color Barcode

Devi Parikh
dparikh@cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213

Gavin Jancke
gavinj@microsoft.com
Microsoft Research
Redmond, WA 98052

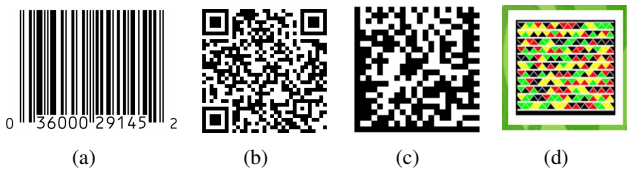
Abstract

A 2D color barcode can hold much more information than a binary barcode. Barcodes are often intended for consumer use where using a cellphone, a consumer can take an image of a barcode on a product, and retrieve relevant information about the product. The barcode must be read using computer vision techniques. While a color barcode can hold more information, it makes this vision task in consumer scenarios unusually challenging. We present our approach to the localization and segmentation of a 2D color barcode in such challenging scenarios, along with its evaluation on a diverse collection of images of Microsoft's recently launched High Capacity Color Barcode (HCCB). We exploit the unique trait of barcode reading: the barcode decoder can give the vision algorithm feedback, and develop a progressive strategy to achieve both - high accuracy in diverse scenarios as well as computational efficiency.

1 Introduction

With the proliferation of inexpensive cameras such as in cellphones or webcams, the consumer use of barcodes is becoming popular. A consumer can take an image of the back of a product with the barcode printed on it with his cellphone camera or webcam. A computer vision algorithm localizes and segments the barcode, and the bits extracted are passed on to the appropriate decoder, and once the product is identified, the relevant information about the product can be retrieved.

Most traditional barcodes are binary barcodes, be it linear barcodes such as the popular UPC (Universal Product Code) barcode shown in Figure 1(a), or 2D barcodes such as the QR code shown in Figure 1(b), or Datamatrix shown in Figure 1(c). These barcodes often contain distinct visual cues, albeit at the cost of expensive estate, such as the three square patterns on three corners of the QR code, which hold no information. On the other hand consider 2D color barcodes with minimal visual cues as seen in Figure 1(d). Not only do they have added aesthetic value, they hold much more information in the same physical size of the code.



(a) (b) (c) (d)

Figure 1: *(a) UPC code (b) QR code (c) Datamatrix (d) Microsoft's High Capacity Color Barcode (HCCB) (Viewed better in color).*

However, these added benefits come with an added cost. Reading these 2D high capacity color barcodes portrays a significant computer vision challenge. This is because of several factors. The color balancing in different cellphone cameras and webcams is drastically different. Since we are dealing with images taken by consumers, the location of the barcode in the image, the orientation of the barcode, etc. are mostly unconstrained. Possible perspective transforms can distort the geometry of the barcode. For different products, the densities and sizes of the barcodes may vary. The lighting conditions under which the images are taken can vary drastically, and given the state of current cellphone cameras, the images can be quite blurred and of poor quality. In this paper we present an approach to localize and segment a 2D high capacity color barcode in such consumer images.

2 Microsoft's HCCB

We work with Microsoft's recently introduced 2D High Capacity Color Barcode (HCCB) [1, 2]. While it may be used for a variety of applications, the most immediate application is for uniquely identifying commercial audiovisual works such as motion pictures, video games, broadcasts, digital video recordings and other media. HCCB, as it is designed, is shown in Figure 1(d). It has rows of strings of symbols (triangles), which we wish to identify, each of which can be one of four colors: black, red, green and yellow. The number of symbols in each row is always an integral multiple of the number of rows, which can vary. HCCB is designed to have a black boundary around it, further surrounded by

Figure 2: *Example images of the Microsoft High Capacity Barcode (Viewed better in color).*

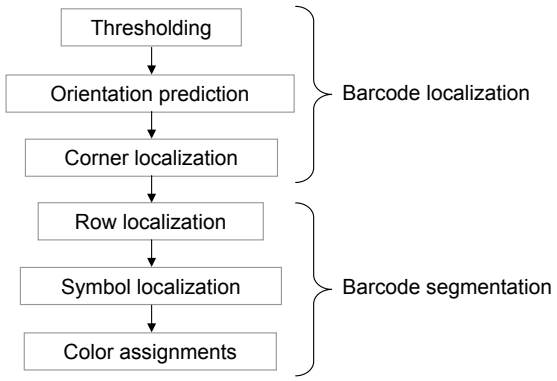


Figure 3: *Overview of our approach to the localization and segmentation of a 2D color barcode.*

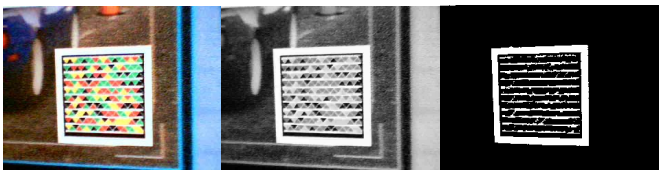


Figure 4: *Left to Right: The color input image I_c, the computed greyscale image I_g and the corresponding thresholded image I_w*

a thick white band. These patterns are designed to act as visual landmarks to locate the barcode in an image. The black boundary at the bottom of HCCB is thicker than the other three sides, which acts as an orientation landmark to account for the fact that the barcode may be at an arbitrary orientation in the image. The last eight symbols on the last row are always in the fixed order of black, red, green and yellow (two symbols per color) and can be used as a palette. There is a white line between consecutive rows. Examples of the real images from which HCCBs are to be read are shown in Figure 2, and demonstrate the challenges posed.

Since these are consumer applications, the approach should be computationally light. An overview of our approach is shown in Figure 3. We assume that one point in the image that lies within the barcode is known. This can be the location of the cross hair when capturing the image. This constrain makes the barcode localization problem more reliable. Most existing barcode reading algorithms are proprietary information and hence there is minimal literature publicly available on them. To our knowledge, this is one of the few initiatives to make such information available to the community.

The rest of the paper is organized as follows. Detailed descriptions of the barcode localization and segmentation approaches are given in Sections 3 and 4 respectively. We describe an interesting progressive strategy to address this problem in Section 5 followed by results in Section 6 and conclusions in Section 7.

3 Barcode localization

3.1 Thresholding

We first threshold the input color image I_c to retain only the white portions of the image corresponding to the thick white band surrounding the barcode and the white lines separating the rows that are a part of the barcode design. To account for the varying lighting conditions across images, we wish to normalize the image before thresholding it; however, to account for varying lighting conditions within a single image, we do so adaptively. We convert I_c to grey scale I_g, divide it into four equal blocks, and normalize each block individually so that the pixels cover the range from 0 to 1. We then threshold the entire image at 0.7 to get the thresholded image I_w. An example is shown in Figure 4.

3.2 Orientation prediction

In order to determine the orientation of the barcode in I_c, we use the repeated pattern of the rows found in the barcode. We work with I_w and the single point known to lie inside the barcode. We extract a $\frac{t}{4} \times \frac{t}{4}$ patch around this point from I_w, where t is the minimum of the width and height of I_w. We compute the Hough Transform of this patch to detect lines. The rows being mostly parallel, we expect to see a strong response for one of the orientation values θ corresponding to the orientation of the rows, which we determine by summing out the other dimension of the hough transform and retaining a 1D profile corresponding to different values of θ, as seen in Figure 5. The orientation of the barcode is determined to be the value of θ corresponding to the maximum value in this profile. Having determined the orientation of the barcode in the image, we rotate I_c, I_g and I_w accordingly so that the barcode is now upright, and we work only with these from here on.

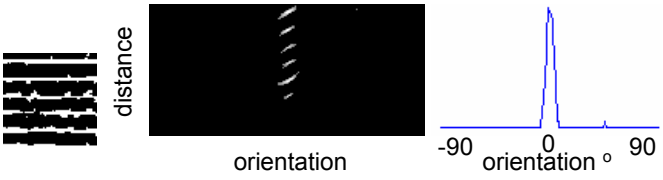


Figure 5: *Left to Right: The patch extracted from thresholded image, the Hough transform of the patch, and the 1D orientation profile obtained by summing the Hough transform along* distance, *where the peak corresponding to the orientation of the barcode is evident.*

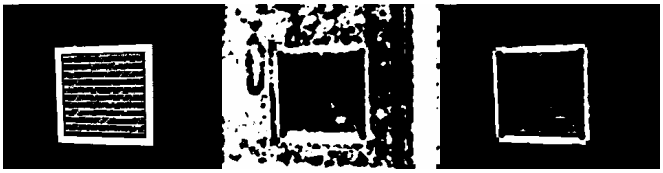


Figure 6: *Left to Right: The thresholded image I_w based on whiteness, the output of texture classifier, the final mask I_x obtained by combining the two.*

3.3 Corner localization

We wish to find the four corners of the barcode that enclose the symbols. We first estimate rough locations for the corners, and then locally refine them. We start from the point known to lie inside the barcode in I_w, and grow a rectangle around it till it lies entirely inside the thick white bands. However, the yellow color inside the barcode is often classified as white in I_w due to poor image quality. These false positives prove to be significant distractions. To remove them, we exploit the fact that the white bands that we are interested in are textureless, where as the inside of the barcode where these yellow regions are found is highly textured.

3.3.1 Texture classifier

We build a simple binary texture classifier which separates the textured regions from the non-textured regions. We compute the output of a Harris corner detector [3] on I_g. The regions with a response lower than 0.01 are classified to be textureless, and rest are considered to be textured. We combine this map with I_w to obtain a new binary mask for the image I_x, which is on for white and textureless regions only. An example is shown in Figure 6. It can be seen that the texture classifier cleans up I_w significantly to obtain I_x.

3.3.2 Rough corner localization

In order to find the rough corners, we start with a $\frac{t}{10} \times \frac{t}{10}$ square in I_x surrounding the point known to lie inside the barcode. We swipe the right edge of this square outwards in I_x till the average values of the pixels that lie on this edge is above a certain threshold. We set this threshold to τ_w (value will be made clear in later sections). We do this for the left, bottom and top edges. The four edges are now in the thick white band, which gives us a rough estimate of the corners (usually within 30 pixels of the true corner location). An example of this is shown in Figure 7.

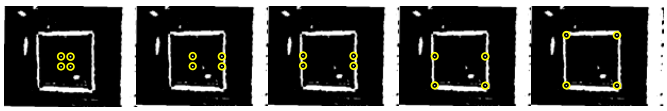


Figure 7: *Left to Right: Starting with a small square around the point known to lie inside the barcode, we swipe each edge till it is mostly white. This gives us the initial rough corner estimates.*

Figure 8: *Left to Right: Patch extracted around marked estimated corner, gradient of the patch m respecting expected gradient directions, filtered gradient m_f, m_f weighted by a gaussian, refined location of corner.*

3.3.3 Gradient based refined corner localization

In order to further process the barcode, we need accurate estimates of the corners. So we refine each of the four corners locally. We extract 50×50 patches from I_g around the each of the four estimated rough corner locations, and wish to find the location of the true corners in these patches. We know that the true corner should have a high gradient value.

Exploiting expected gradient directions: Because of the design of the barcode, we know the directions of the gradients near each of the four corners. For instance, the top-left corner of the barcode should have a dark portion on its right and bottom, and a bright portion on its left and top. So it would have a highly negative gradient in both x and y directions. The magnitude of the gradient is computed as $m = -ix - iy$, where ix and iy are the gradients along the x and y directions respectively, as shown in Figure 8. This is repeated for all four patches with appropriate gradient directions.

Estimating blur: m is often noisy, so we filter it with a median filter. In our experiments we found that the size of this filter is crucial and is dependent on the blur of the patch. We use $M = max(m)$ to estimate the blur of the patch. The sharper the patch, the higher the contrast between the white band and the black border will be and the higher the value of M. We empirically determined a mapping from M to the

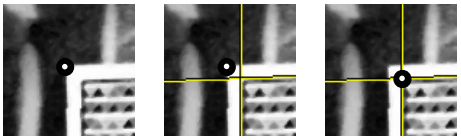


Figure 9: *Left to Right: Patch extracted around estimated corner, strongest horizontal and vertical lines localized, refined location of corner at intersection of two localized lines.*

appropriate filter sizes to filter m and obtain m_f as shown in Figure 8.

Down weighing background: m_f is weighted by a Gaussian fall-of so that the center of the patch has a higher weight than the periphery. This is to avoid any high gradient information that may be present towards the outside of the patch due to background clutter that may have been picked up. The refined location of the corner is the location of $max(mf)$, as shown in Figure 8.

3.3.4 Line based refined corner localization

The above refining method gives us accurate estimates of the corners most of the time. To eliminate the errors further, we employ one further local refinement step. We consider another 30×30 local patch from I_g around the estimated corner positions. We use hough transform to find the strongest horizontal and vertical lines in this patch. The location of the refined corner is the intersection of these lines, as shown in Figure 9.

4 Barcode segmentation

Having determined accurate locations of the four corners, we have successfully identified the portion of I_c (and I_g) that contains the barcode and only the barcode. Lets call this region of interest B. The barcode could be perspectively transformed, and so B is an arbitrary quadrangle. If we can transform B to be a rectangle, the rest of the processing would be much more straightforward. Before we can transform B, we need to choose a meaningful aspect ratio r for the rectangle. An intelligence choice is made as follows. Let s_1 be the length of the left side of the quadrangle, and s_2, s_3 and s_4 be the lengths of the other sides in clock-size order. Let s_l be the average of s_1 and s_3, s_w of s_2 and s_4. Then $\hat{r}$ is $\frac{s_w}{s_l}$ and r is the integer closest to $\hat{r}$.

We determine the perspective transform T required to transform B to a rectangle of size $200 \times 200r$. We use T to transform B from I_g as well as I_c to obtain B_g and B_c respectively. We now wish to identify the string of colors of the symbols in B_c. Illustrations are shown in Figure 10

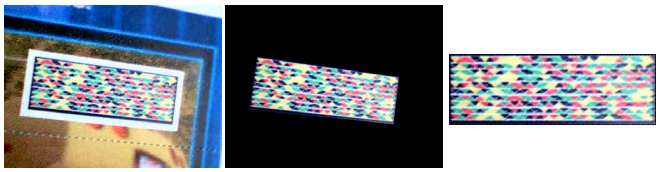


Figure 10: *Left to Right: Input image, region of the input image lying within the corners localized, barcode extracted and transformed to a canonical rectangle.*

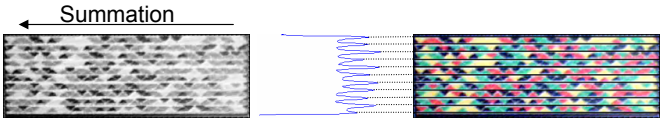


Figure 11: *Grayscale extracted barcode summed along one dimension to get the intensity profile seen in the middle. The peaks located in this intensity profile corresponding to the the locations of the row separators is also shown.*

It should be noted that the orientation predicted in Section 3.2 is mod 180°, so we could not have differentiated an upside down barcode from an upright one. We now look for the thick black line at the bottom of the barcode that is a part of the design to act as the orientation landmark, to compute the orientation upto 360°.

4.1 Row localization

We first try to identify the location of the white lines that act as rows separators. In order to do this, we work with B_g, which we know has the rows all in horizontal orientation. We sum B_g along the horizontal dimension and get a 1D profile as illustrated in Figure 11. The peaks corresponding to the white row separators are evident. Given this 1D profile we determine an approximation of the width W of the rows using FFT. We use non-local maxima suppression to determine the exact location of the peaks, using a window size that is about $\frac{W}{2}$. The locations of the row separators is also shown in Figure 11.

4.2 Symbol localization

Having localized the row separators, we now analyze each row to determine the location of the centers of the symbols/triangles. We first need to determine the number of triangles per row. The barcode is designed so that the number of symbols per row is $(r+1)R$, where R is the number of rows in the barcode. We can sample each row uniformly (considering every other triangle to be inverted) and those would be the locations of the centers. However, due to slight errors in corner localization, which may further get amplified while computing the perspective transform T, this is

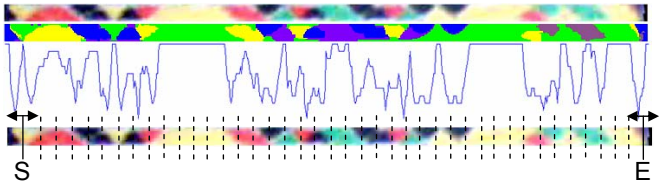


Figure 12: *Top to bottom: The row being currently analyzed, the clusters assigned to each of the pixels in the row, the quality measure to evaluate each sample placed, and the search performed over different values of S and E. (Viewed better in color).*

not accurate enough. For a single row, the start (left) and end (right) locations of the sampling, S and E respectively, are the main degrees of freedom. So we search over multiple values of S and E. For every pair (S,E), we compute a quality score as the sum of the quality scores of each sample that would be obtained if we were to sample uniformly within the range between S and E. To compute the quality of samples, we cluster the pixels in the row into four clusters using mean shift [4, 5]. The quality of a sample at a certain horizontal location in the row is the maximum proportion of height of the row at that location assigned to the same color. So if a sample is located at the center of a triangle the quality of the sample will be high. Figure 12 illustrates these ideas. Having computed scores for different values of (S,E), we pick (S,E) with the highest score. This is the best global strategy for the samples in the row, but there is room for improvement locally. So given these uniformly placed samples between S and E, we search a small neighborhood (3 pixels wide) around each sample, and shift the sample to the local maxima in terms of the quality measure. These are the final locations for the symbols in this row.

4.3 Color assignments

Even though we computed the four color clusters using mean shift, it is not clear which cluster corresponds to which one of the four colors: black, red, green and yellow. Naive nearest neighbor assignments of clusters to the corners of the color-cube does not work because of poor image quality and drastically varying color balancing in cameras. We use the palette at the bottom right corner of the barcode to assign the colors.

5 Progressive strategy

While doing experiments with images of Microsoft's HCCB, we found that the percentage of barcodes that could be correctly decoded given the output of our algorithm was not satisfactorily high. However, given the correct (hand-clicked with likely errors of a few pixels) corner locations for the barcodes, a high percentage of barcodes were successfully decoded. This indicated that the errors were mainly on the part of our corner localization approach. We evaluated several different combinations of the approaches presented in Section 3.3 (for instance, rough corner estimation immediately followed by the line-based refinement, or the rough estimation followed by no refinement at all, etc.). We found that no single combination worked well for all images, however the errors of different combinations were complementary. Also, barcode reading is different from most other localization tasks in computer vision in the sense that the results are verifiable - the decoder can provide feedback to the vision algorithm about whether the barcode was decoded successfully or not. We exploited the combination of these two factors and developed a progressive strategy.

For a given input image, we hypothesize a set of corners using one approach, and if the barcode can not be decoded successfully, we hypothesize again using another approach and so on. Since each hypothesis individually is computationally inexpensive, this is much more feasible than attempting to develop a single strong approach that is effective for the entire diverse collection of images (similar in philosophy to ensemble of classifiers in machine learning). The order in which we employ the different approaches is determined to optimize the computational time i.e. the approach that is shown to be effective for most images is employed first and so on, so that most barcodes are decoded successfully in the shortest time, and very few barcodes take longer. We design 12 different approaches, the variables being the threshold τ_w used in Section 3.3.2 (0.2, 0.5 or 0.9) and the strategies to locate the corners i.e. only rough corner estimation or one or both of the local refinements approaches (gradient based or line based).

6 Results

We evaluate our approach on a 1000 images of Microsoft's High Capacity Color Barcodes of varying densities and sizes taken under varying conditions such as those depicted in Figure 2 which represent typical consumer usage scenarios. The barcodes had anywhere from 10 to 60 rows, with 20 to 120 symbols per row, making the number of symbols per barcode to range from 200 to 7200. The known point inside the barcode was assumed to be the center of the image. The ground truth sequence of colors present in the barcodes was available.

Let us first look at the results of the segmentation part of the approach as a stand alone piece. We hand-clicked corners of about 500 of the 1000 images and provided these corner co-ordinates as input to the barcode segmentation module. We compared our predicted string of colors with the ground truth, and computed the accuracy per image as the percentage of symbols correctly identified. Averaging across the 500 images, we found that we can correctly iden-

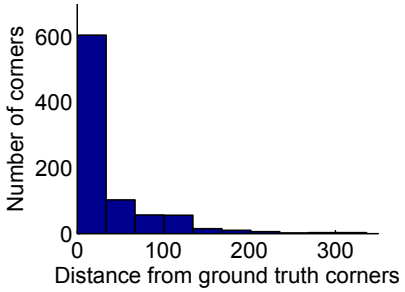


Figure 13: *Histogram of errors in corner localization using the approach as described in Section 3.3. While most errors are small enough, several are too large.*

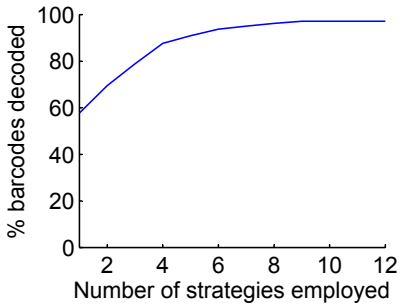


Figure 14: *Percentage of barcodes successfully decoded as more approaches are employed as part of the progressive strategy. It can be seen that while most barcodes can be successfully decoded using the first few approaches, the subsequent approaches help if higher accuracies are desired.*

tify the colors of 94% of the symbols. This shows that given good corner co-ordinates, the segmentation module of the approach works well.

We now evaluate our entire corner localization approach as described in Section 3.3 on same 500 images, assuming the hand-clicked corners to be ground truth. The histogram of errors of the corners is shown in Figure 13. It can be seen that while most corners are found accurately, several have large errors.

We now show the behavior of the progressive strategy. In order to do so, we need the decoder in the loop to provide feedback to the vision algorithm. Since the focus of this paper is on the vision algorithm, we do not discuss the details of the decoder. However, let us use a simple model to simulate the decoder. Let's say if the colors of 85% or higher of the symbols in a barcode are correctly identified, the barcode can be successfully decoded through its error correction scheme (this is a realistic number for the decoder being used with Microsoft's HCCB), otherwise it can not.

Figure 14 shows the percentage of barcodes of the 1000 images that are successfully decoded with each added hypothesis in the progressive strategy. We can see that the first few strategies can successfully decode most barcodes, and as we keep adding more strategies we get more success, however at a diminishing rate, with the curve flattening out at about 97.2%. We can see that by employing just one best strategy we would have a performance of only about 60%. The computation cost at which these added successes are obtained is about 6.1 seconds per strategy (using unoptimized Matlab code run on a standard desktop computer). For different applications, we would want to function on different operating points of the tradeoff between accuracy and computational expense. The progressive strategy employed gives us the freedom to easily manipulate these design choices.

7 Conclusions

We presented our approach to the localization and segmentation of a 2D high capacity color barcode, under various challenging scenarios of consumer use. Our approach is fairly computationally inexpensive, and yet accurate on images of Microsoft's recently launched 2D High Capacity Color Barcode (HCCB). We exploited the unique nature of reading barcodes as compared to other computer vision detection tasks, in that the output of the vision algorithm is verifiable with the barcode decoder in the loop, and proposed a progressive strategy that is similar in philosophy to ensemble of classifiers, where we use multiple simple approaches instead of a single strong one. This also allows for an explicit design choice to trade-off accuracy and computational time.

Acknowledgments

We would like to thank Larry Zitnick, Andy Wilson and Zhengyou Zhang for useful discussions over the course of this work.

References

[1] News article at: http://www.news.com/Microsoft+gives+bar+codes+a+splash+of+color/2100-1008_3-6175909.html?tag=cd.lede

[2] News article at: http://seattlepi.nwsource.com/business/311712_software16.html

[3] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[4] K. Fukunaga and L.D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. In *IEEE Transactions in Information Theory*, 1975.

[5] Y. Cheng. Mean shift, mode seeking, and clustering. In *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 1995.