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

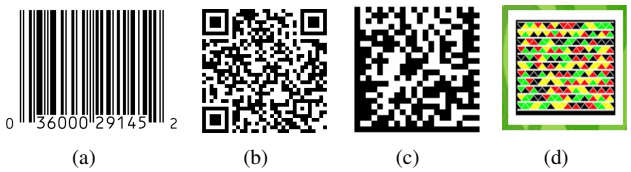
Abstract

A 2D color barcode can hold much more information than a binary barcode. Barcodes are often intended for consumer use, such as, a consumer can take an image with her cellphone camera 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 unusually challenging because of the varying color balancing in different cameras, poor quality of images taken with current cellphone cameras and webcams, varying lighting conditions, arbitrary rotation and even perspective transforms of the barcodes in images. 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 containing Microsoft's recently launched High Capacity Color Barcode (HCCB). The problem of reading barcodes has a special trait compared to other computer vision localization problems - the results are verifiable in that the decoder can give the vision algorithm feedback of whether the barcode was successfully decoded or not. We exploit this trait in an interesting way and develop a progressive strategy to achieve a fine balance between the requirement of the algorithm to be effective in drastically varying scenarios as well as computationally inexpensive.

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 information pertaining to 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 as seen in Figure 1(b), which hold no information. On the other hand consider 2D color barcodes with minimal visual cues. 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 (like with most tradeoffs in design!) come with an added cost. Reading these color barcodes with minimal visual cues in a consumer setting as described above portrays a significant computer vision challenge as compared to reading binary barcodes. 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. In the case of 2D barcodes, 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 that is invariant or robust to these variations. We work with Microsoft's recently introduced 2D color barcode called the High Capacity Color Barcode (HCCB) [1, 2]. The barcode as it is designed is shown in the top-left corner of Figure 1(d). Ex-

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

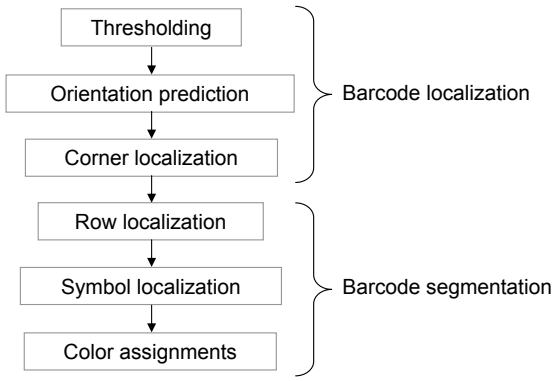


Figure 3: *Flowchart of our approach to the localization and segmentation of 2D color barcode.*

amples of the real images to be dealt with are shown in Figure 2 which demonstrate some of the variations we mentioned that the vision algorithm developed for localizing and segmenting HCCB has to be invariant or robust to. Also, since these are consumer applications, the approach should be computationally light. Most existing barcode reading algorithms are proprietary information and hence there is minimal literature publicly available on them.

While Microsoft's HCCB may be used for a variety of applications, the most immediate application for HCCB is for uniquely identifying commercial audiovisual works such as motion pictures, video games, broadcasts, digital video recordings and other media and Microsoft has recently joined hands with ISAN-IA (International Standard Audiovisual Number International Agency) [3].

The design of HCCB can be seen in Figure 1(d). It has rows of strings of symbols (triangles), and each triangle can be one of four or eight colors. In this paper we focus on the four-colored HCCB, the four colors being 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 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 pallet. There is a white line between consecutive rows.

The rest of the paper is organized as follows. Section 2 briefly introduces our approach to the localization and segmentation of HCCB, followed by detailed descriptions of these two components 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.

2 Approach

The vision algorithm needs to locate and segment the HCCB from a given image, and determine the string of colors of the symbols found in it. Instead of providing a hard decision for which symbol is which color, we wish to provide a soft assignment, i.e. for each symbol we wish to generate a four dimensional vector holding the confidence for the symbol to be each of the four colors.

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. Without this constraint, generalized localization and segmentation of the barcode with industry level accuracies, in the challenging scenarios we describe, is unfeasible.

Our approach is described in the flowchart shown in Figure 3. We describe each of these components in detail below. Since the approach needs to be computationally inexpensive, each of our design choices are simple yet effective.

3 Barcode localization

We first describe our approach to localize the barcode in the input image i.e. to find the location of the four corners of the barcode.

3.1 Thresholding

We first threshold the input color image to attempt to retain only the portions of the image that are white. This is in order to retain the thick white band surrounding the barcode and the white lines separating the rows. This thresholded image will be used for predicting the orientation of the barcode in the image as well as locating the corners

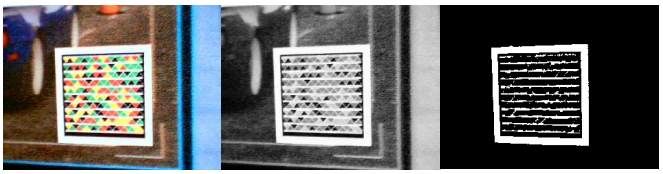


Figure 4: *Left to Right: The color input image I_c, the computed greyscale image I_g and the corresponding thresholded image I_{tw}*

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 one dimension, where the peak corresponding to the orientation of the barcode is evident.*

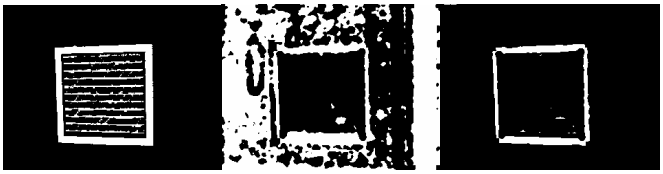


Figure 6: *Left to Right: The thresholded image I_{tw} based on whiteness, the output of texture classifier, the final mask obtained by combining the two.*

of the barcode. To account for the varying lighting conditions across images, we wish to normalize the image before thresholding it. Also, to account for varying lighting conditions within a single image, we use an adaptive normalization. Specifically, we convert the input color image into grey scale and normalize the image adaptively by dividing it into four equal blocks, and normalizing each block individually so that the pixels cover the range from 0 to 1. We then threshold the entire image with a global threshold, which we set to 0.7. An example thresholding result is shown in Figure 4. Let's call the input color image I_c, the greyscale image I_g and the thresholded image that has the white portions retained I_{tw}. It should be noted that we provide intermediate results images at every step in the approach on an intentionally selected good quality image for clarity. But in reality, most images are certainly not as well behaved.

3.2 Orientation prediction

The next step is to determine the orientation of the barcode in the I_c. In order to do this, we rely on the repeated pattern of the rows found in the barcode. We work with I_{tw} 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_{tw}, where t is the minimum of the width and height of I_{tw}. We compute the Hough Transform [5] of this patch. The rows being mostly parallel, we expect to see a strong response for one of the orientation values θ, which we determine by summing out the other dimension of the hough transform and retaining a 1D profile corresponding to different values of θ. An example profile along with the associated patch can be 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_{tw} accordingly so that the barcode is now upright. From here on, whenever we refer to I_c, I_g or I_{tw}, we refer to the rotated version where the barcode is upright.

3.3 Corner localization

Now we wish to find the four corners of the barcode that enclose the symbols, but exclude the surrounding white band. We do this by first estimating rough locations for each of the four corners, and then locally refining them. In order to find the rough estimate of the corners, we work with I_{tw}. The strategy is to start from the point known to lie inside the barcode, 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_{tw} due to poor image quality. These false positives prove to be significant distractions for the corner localization approach. 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 [6] on I_g. And 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_{tw} to obtain a new binary mask for the image. Examples of these images are shown in Figure 6. It can be seen that the texture classifier cleans up the thresholded image significantly. Let's call this cleaner binary image I_{twt}, which is on for white and textureless regions only.

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_{twt} surrounding the point known to lie inside the

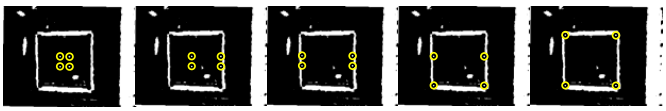


Figure 7: *Left to Right: Starting with a small square around the point known to lie inside the barcode, we swipe each edge till it is mostly white. This gives us the initial rough corner estimates.*

barcode. We start with the right edge of this square and swipe outwards in I_{twt} 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. An example of this is shown in Figure 7.

3.3.3 Gradient based refined corner localization

In order to further process the barcode, we need very accurate estimate of the corners. So we refine each of the four corners locally. We extract square 50×50 patches from I_g around the estimated rough corner location. We compute the gradient of the patch. 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 it's left and top. So the point we are interested in is one where the magnitude of the gradient is high, but respecting these expected directions of change. The magnitude of the gradient respecting this direction is computed at each point in the patch. Let this gradient patch be m.

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 of M to the appropriate filter size, and filter m, which we call mf.

Down weighing background: mf is weighted by a Gaussian fallof 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

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

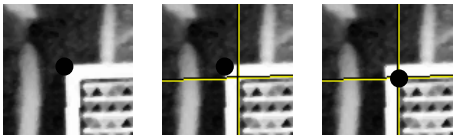


Figure 9: *Left to Right: Patch extracted around estimated corner, strongest horizontal and vertical lines localized, refined location of corner at intersection of two localized lines. (The location of the estimated corner is taken to be the same as the result of the rough corner estimation instead of that refined using the gradient for illustration purposes).*

$max(mf)$. This is repeated for the remaining three corners. An illustration of how this works is shown in Figure 8.

3.3.4 Line based refined corner localization

The above refining method gives us accurate estimate of the corners most of the time. However, to meet industry level accuracies, we wish to eliminate as many errors as possible. So 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. An illustration of this is 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 determine a meaningful aspect ratio r for the rectangle. We compute this 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 $r^{'}$ is $\frac{s_w}{s_l}$ and r is the integer closest to $r^{'}$.

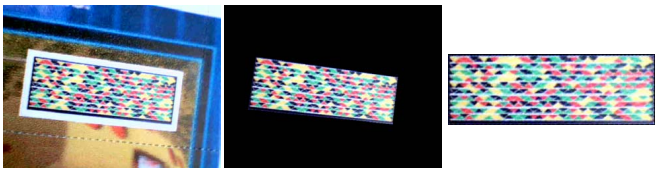


Figure 10: *Left to Right: Input image, region of the input image lying within the corners localized, barcode extracted and transformed to a canonical rectangle.*

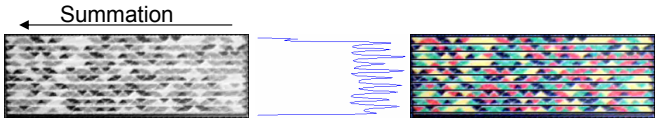


Figure 11: *Left to Right: Grayscale extracted barcode summed along one dimension to get the intensity profile seen in the middle. The peaks located in this intensity profile are color black in the barcode to identify the location of the row separators localized.*

We don't use this aspect ratio for any geometric reasoning, so this approximation is good enough for our purposes to simply transform the localized barcode to a meaningful rectangle size. Having determined 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 of these regions is shown in Figure 10

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 such as that 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 symbols (triangles) in each row. In order to do this, we need to determine the number of triangles per row. Given the design of the barcode, it turns out that the number of triangles per row is $(r+1)R$ and r is the aspect ratio computed earlier and R is the number of rows in the barcode. Having determined the number of triangles per row, we can sample each row uniformly (respecting the geometry of every other triangle being inverted) and those would be the locations of the triangles. However, due to slight errors in corner localization, which may further get amplified while computing the perspective transform T, this does not work accurately 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. This score is computed 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 first perform clustering on the pixels in the row into four clusters using Meanshift clustering [7]. The quality of a sample at a point is the proportion of height of the row at that point assigned to the majority color in that height. So if a sample is located at the center of a triangle, most of the height of the row at that point will be of one color, and the quality of the sample will be high. Figure 12 illustrates these ideas. Having computed scores for all (practical) values of (S,E), we pick (S,E) with the highest score and uniformly sample the row in this range. This is the best global strategy for all the samples in the row, but there is room for improvement locally. So given these uniformly placed samples, we search a small neighborhood (5 pixels wide) around each sample, and shift the sample to the local optimum in terms of the quality measure. These are the final locations for the symbols. This is repeated for every row.

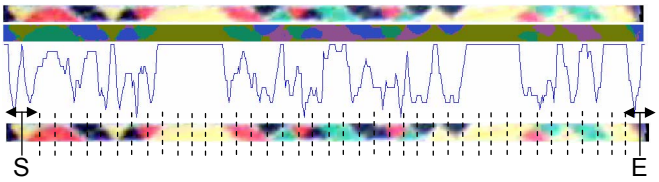


Figure 12: *Top to bottom: The row being currently analyzed, the clusters assigned to each of the pixels in the row, the quality measure to evaluate each sample placed and the search performed over different values of S and E between which samples are placed uniformly. The score associated with a certain value of S and E is the sum of the quality measure shown at each of the sample points.*

4.3 Color assignments

Even though we computed the four color clusters using Meanshift, 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 colors does not work because of poor image quality and drastically varying color balancing in cameras. So we use the pallet at the bottom of the barcode to assign the colors. We should

point out that when the orientation of the barcode was computed, it was mod 180°. So the barcode could be upside down at this stage. In order to fix this, we look for the thick black line at the bottom of the barcode that is part of the design and acts as the orientation landmark. If the thick black line is found on top, we flip the barcode vertically before doing the color assignments.

4.3.1 Local confidence

As stated earlier, we wish to assign a confidence to each symbol to be assigned to each of the four colors. We incorporate a local confidence measure as well as a global confidence measure. In order to compute the local confidence, we look at a small neighborhood of 5×5 pixels around each sample and compute the proportion of pixels assigned to each of the four colors.

4.3.2 Global confidence

Due to poor quality of images, the clustering algorithm often combines two colors into one cluster. This can be localized by exploiting the fact that the barcodes are designed so that the colors are as random as possible i.e. the histogram of the four colors in a barcode is mostly uniform. So if the clustering algorithm has merged two clusters into one, the histogram of our predicted colors in the barcode using these clusters would have one of the colors sharply peaked, and another color nearly zero. If this is localized, we have a reason to believe that these two colors have been confused by the clustering algorithm, and this should be reflected in the confidence values globally. For example, if we believe the green and black have been merged into one cluster which has been assigned to black, whenever we see a black symbol in the barcode, it could have been green with an equal confidence of being black. So among all the local confidence values for all symbols, we simply copy the local confidence values for a symbol being black to the symbol being green (unless the confidence of the symbol being green was assigned a higher value than it being black). We now have the string of colors for the symbols found in the barcode, and associated confidence values that can now be passed on to the decoder to be decoded.

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. Analyzing these errors further, we found that no single settings of parameters throughout the corner localization approach worked well for all images, however the errors of different parameter settings were complementary. Also, barcode reading is different from most other localization tasks in 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 first use one approach to corner localization, and if the barcode can not be decoded successfully, we try another approach and so on. Since each approach 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.

The order in which we try the different approaches is determined to optimize the computational time i.e. the approach that is shown to be effective for most images is tried 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 we consider are the threshold τ_w used in Section 3.3.2 (0.2, 0.5 or 0.9) and the strategies to find corners that are employed 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. 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. It should be noted that since the ground truth is as the sequence of colors, and not confidence measures, we do not have a way to evaluate our confidence measures. From the confidence values, we assign a symbol to the color with the highest confidence, and then compare the string of colors our algorithm produces, with the ground truth.

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

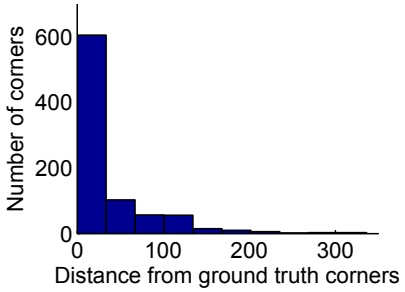


Figure 13: *Histogram of errors in corner localization using the approach as described in Section 3.3. While most errors are small enough, several are too large.*

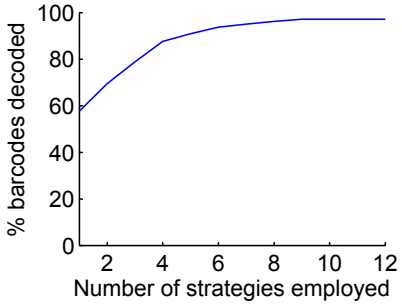


Figure 14: *Percentage of barcodes successfully decoded as more approaches are employed as part of the progressive strategy. It can be seen that while most barcodes can be successfully decoded using the first few approaches, the subsequent approaches help if higher accuracies are desired.*

across the 500 images, we found that we can correctly identify 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 for 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 (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 strategy we employ. We can see that the first strategy can successfully decode most barcodes, and as we keep adding more strategies the number of added successes decreases significantly, with the curve flattening out at about 97.2%. We can see that by employing just one of the best strategies 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 (the code is unoptimized and implemented in Matlab and was 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 operating points.

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 gave industry level accuracies on images of Microsoft's recently launched 2D High Capacity Color Barcode (HCCB). We exploited the unique nature of the reading barcodes as compared to other vision detection tasks, 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 tradeoff between 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] International Standard Audiovisual Number, http://www.isan.org
[4] News release at: http://www.isan.org/portal/page?_pageid=165,41294&_dad=portal&_schema=PORTAL
[5] Hough transform
[6] Harris corner detector
[7] Meanshift