Automatic Coin and Bill Detection

Dominic Delgado
Stanford University
650 Serra Mall
Stanford, CA 94305
delgado4@stanford.edu

ABSTRACT

This project combines feature-based methods, the Hough transform, and Linear Discriminant Analysis to combine and extend past projects [1,2] in the field of currency detection. The proposed method attains reasonable accuracy for both coin and bill detection while allowing for more general configurations of coins and performing coin recognition significantly faster than had been achieved previously.

1. INTRODUCTION

As more and more of our daily transactions become electronic, the prospect of counting more than a few coins, or even small bills, begins to feel all the more tedious. For someone who has trouble seeing at all or a traveler unfamiliar with the local currency, counting money quickly and accurately may seem all the more daunting. A couple of previous projects have presented solutions to the problems of counting coins [1] and bills[2] with varying degrees of success. This paper seeks to provide a unified solution that also offers performance suitable for mobile applications.

The coin and bill detection algorithm solves the problem of computing the total value of U.S. currency displayed in a given image by solving a set of detection problems – namely, to detect the number of each possible bill and each possible coin in the image – and then aggregates over the results. Although both classes of objects are semantically similar, their visual properties and relative expected quantities recommend a distinct approach for each. Dollar bills reflect light relatively uniformly and have a diverse combination of distinctive patterns that identify each. This suggests that a feature-based approach should be effective. Since such an approach requires exhaustively comparing candidate images from a database, there should be relatively few objects to match in a given image for reasonable runtime on compute-constrained systems. This is a reasonable assumption for bill counting because bills occur in much smaller amounts than coins. They are also physically much larger than coins – there is roughly an order of magnitude of difference in area between a bill and the largest coin – so for a fixed image size, one could capture ten or more times as many coins as dollars.

Naturally, the converse applies for coins. With a relatively large number of coins expected, we would like as simple and computationally efficient an algorithm as possible so that reasonable runtime may be expected for dozens of coins, even on a phone. These two philosophies suggest separate pipelines for each class of currency. In this algorithm, those two pipelines are implemented in terms of the Scale-Invariant Feature Transform (SIFT) [??] and the circular Hough transform [??], respectively.

2. METHODOLOGY

2.1 Bills

2.1.1 Feature-Based Detection

The bill detection part of the pipeline implements a relatively standard feature-based detection algorithm inspired by previous work on foreign bill detection [2]. When an image is taken, the algorithm uses SIFT to extract local gradient information at points of interest, where a point of interest is one that maximizes the response of the Harris-Laplacian detector. These features are compared with similarly extracted features from images of the obverse and reverse sides of each valid bill, which have been stored in a database.[1] The algorithm uses the distance ratio test to detect correspondences between features in each image. [3] Two features are said to match if the Euclidean distance between them is less than 1.5 times the Euclidean distance between each of the two features and any other feature in the other image. Outliers are removed with Random Sample Consensus (RANSAC) [4], which estimates the best homography mapping one image to another. The above process repeats for each bill in the database while the number of inliers detected by RANSAC exceeds an empirically determined threshold.

Figure 1. Homography Estimation

2.1.2 Inlier Removal

In order to avoid counting bills multiple times, we must remove the inliers detected by RANSAC at each iteration that the algorithm determines a match. Naively removing only those inliers does not suffice to prevent double-counting, since not all feature correspondences are detected by RANSAC. To solve this, we remove all features within an elliptical region. This ellipse is a scaled version of the ellipse specified by the covariance matrix of

---

[1] In this implementation, the database stores 500-600 features per side.

[2] Note that by "downsample", I mean apply antialiasing filter, downsample, etc., not just discarding the unnecessary pixels.

[3] For the sake of simplicity, we assume that, if there are any coins

the inlier positions (x,y). The ellipse is a compromise between, on the one hand, the minimal approach of using a bounding box based on the smallest and largest x and y values, and, on the other, eliminating all feature points located within the convex hull of the inliers. Using an ellipse retains the linear computation time and minimal comparisons of the former while roughly approximating the shape and orientation of the latter.

Figure 2. Inlier removal in elliptic region

2.2 Coins

As mentioned previously, our primary concern with coin detection, beyond accuracy, is speed. To that end, Pendise and Wang [2] developed an algorithm that relied on the relative sizes of American coins and the difference in hue between pennies and other coins, rather than feature matching. Unfortunately, they did not achieve their goal of reasonable computation time, with runtimes ranging between 20 seconds and 1 minute in MATLAB. This coin detection algorithm follows a similar progression, but it achieves much more reasonable runtimes due to some modifications to the circle detection step. Note that the penny classification step also differs.

2.2.1 Preprocessing

Before the detection step, the algorithm binarizes the input image adaptively using Bradley's Method, which is implemented in MATLAB's *imbinarize* function. Afterwards, small, dark regions within connected light regions are removed. This combination of algorithms aims to remove small, distracting regions prior to employing the Circular Hough Transform.

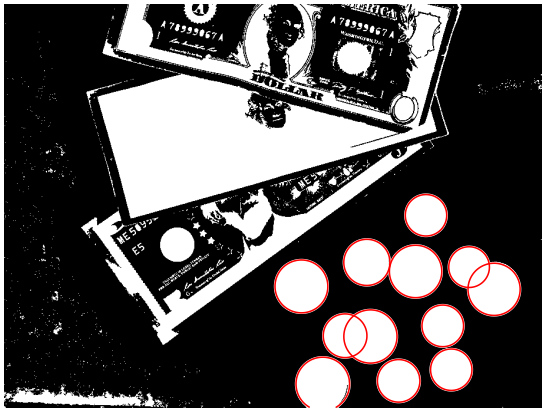

Figure 3. Results of fast circle detection displayed on top of preprocessed image

2.2.2 Fast Circle Detection

The standard method for detecting circular objects in an image is the Circular Hough Transform (CHT). The CHT parameterizes a circle as (x-x_o).^2 + (y-y_o).^2 = R for some radius R and computes votes for each edge pixel as to where the (x_o, y_o) might be. The pixels with votes above a threshold are classified as centers of circles. To find circles of uncertain radius, this process is repeated over a range of possible radii. For a large radius range, the process is slow, and this is cited as the primary reason for the slow runtime of [1].

In order to overcome the slow runtime, we employ successively decreasing downsampling rates to restrict the CHT computation to a small range that can be evaluated quickly. To begin, we assume that any single coin in the image will not take up more than a certain amount of the field of view, say half of the minimal dimension of the image. We downsample[2] the image so that a coin of this maximum size would have a radius less than a specified radius R_max. R_max is 20 in this implementation. We then search the much smaller range [R_min, R_max], which is much easier to search than the original space. If a circle is detected, then we have an idea of the scale of all coins in the image, since the radii of each type of coin is known. Otherwise, we downsample the original image by a smaller factor such that a coin that was of radius R_min in the first downsampled image is now of radius R_max, and again we search for circles in the same small range [R_min, R_max]. In this way, we may exhaustively search all possible radii until either we find a coin or a stopping criterion is met. This computation takes several seconds on an average image, compared to the naïve approach, which takes 20-60s.

2.2.3 Classification

Our coin classification uses the relative size of coins as they appear in the image. We assume that the camera normal is perpendicular to the surface on which the coins lie. Since photographs may be taken at arbitrary distances, classification based on coin size requires calibration. As in [1], we calibrate based on the penny.[3] The penny is already reasonably well-localized, but to employ the simplest classifier possible, we find the projection from the color space into the real numbers that maximizes the separation between pennies and non-pennies using Fisher Linear Discriminant Analysis (LDA) . The classes are well-separated, and a reasonably accurate classifier may then be implemented via thresholding in the reduced space.

At this point, we are ready to classify our coins. We classify as pennies any circle that contains more penny pixels than non-penny pixels. The average radius of these pennies becomes our reference, and we can compute the expected radii of the other coin classes and use minimum-distance detection to determine which coins are of which class. If the radius of a coin of a given class is distributed i.i.d. Gaussian, and if each class has the same variance and prior probabilities, this is the MAP estimate.

---

[2] Note that by "downsample", I mean apply antialiasing filter, downsample, etc., not just discarding the unnecessary pixels.

[3] For the sake of simplicity, we assume that, if there are any coins in the image, at least one is a penny. Otherwise, we must resort to SIFT to calibrate our measurements as in [1], which has horrible performance.

Figure 4. Labeled coins. Quarters (pink), nickels (cyan), pennies (blue).

3. EVALUATION

Performance of this algorithm was evaluated on images collectively containing roughly 50 bills and over 100 coins.

3.1 Runtime

The bill detection pipeline averaged 2.46s when there were no bills present in the image and 4.41s with seven bills, exhibiting the expected dependence on the number of bills present. This study uses a relatively small number of reference bills ($1, $5, and $20), so the runtime could easily be expected to reach into the tens of seconds if it were to check all possible U.S. bills. Parallelization is required for computation times that would be acceptable for a viable product. Fortunately, searching for a particular type of bill is independent of searching for any other type of bill, so we can expect nearly linear gains from parallelization.

The coin detection pipeline averages just over 3s for images that don't have coins, while it does not exceed 1s for images with up to 40 coins. This trend may seem odd, but the long runtime for no coins is a result of the fast circle detection algorithm iterating until it reaches its stopping criterion. When a coin is present, the circle detection algorithm has to iterate many fewer times before the first circle is detected.

3.2 Detection

Under ideal conditions, this automated coin and bill detection algorithm easily recognizes various assortments and arrangements of dollar bills, and it does quite well on coins, too. In both cases, the algorithm can distinguish partially occluded currency. The conditions under which the algorithm is most successful are: 1) more than just the hair of the face on any bill is visible; 2) pennies are not heavily tarnished; 3) coins do not overlap bills, and vice versa; 4) coins overlap by less than 50%. These conditions are already more strict than previous work [1,2], which did not allow coins to overlap. The conditions under which I tested this algorithm were slightly more general: 1) more than ½ in. of a bill must be visible; 2) coins do not overlap bills, and vice versa. Under these conditions, the algorithm achieves correct detection and classification of bills 87% of the time and correctly identifies coins with 83% accuracy. Figure 7 describes the denominations more precisely.

The algorithm has particular difficulty with pennies that are nearly black in color, which is likely due to the lack of copper hue that distinguishes a penny from other coins. Unsurprisingly, the algorithm misclassifies coins that overlap with pennies by more than 50%, since it is an explicit assumption of our model that coins will not overlap by more than 50%. The algorithm also struggles a bit with distinguishing nickels and quarters because of their similarity in size. This confusion is exacerbated by any perspective change caused by tilting the camera.

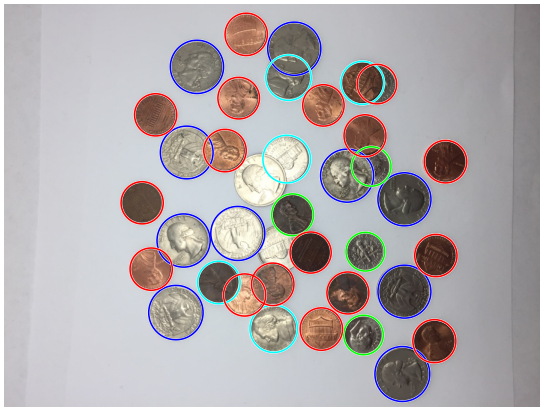

Figure 5. Results for thumb_IMG_1352_1024. Quarter (blue), Nickel (cyan), Dime (green), Penny (red).

Figure 6. Results for thumb_IMG_1360_1024

4. Future Work

This project has continued the efforts of past currency detection algorithms by proposing a method for fast localization and measurement of circular objects, fast classification of pennies, and orientation aware removal of inlier features. Moving forward, the most immediate impediment is robustness to perspective changes. Even a small tilt of the camera can make nickels seem like quarters and vice versa. A simple approach would be to use the homography estimate from the RANSAC step of bill detection to estimate the perspective of the image relative to a flat reference, but I did not find that effective. Perhaps an EM-based approach could jointly estimate the classes of coins and the perspective.

More effective segmentation is also required, both to eliminate misclassifications due to excessive tarnishing of pennies and to enable detection of coins on top of dollar bills. Ultimately, this project would also need to be ported over to Android/iOS to provide utility to anyone.

Image (thumb_IMG_##.jpg)	**Bills**	**Coins**	**Quarters**	**Dimes**	**Nickels**	**Pennies**
1323_1024	7/7	0/0	0/0	0/0	0/0	0/0
1327_1024	6/7	0/0	0/0	0/0	0/0	0/0
1329_1024	7/7	0/0	0/0	0/0	0/0	0/0
1330_1024	5/7	0/0	0/0	0/0	0/0	0/0
1331_1024	5/7	0/0	0/0	0/0	0/0	0/0
1333_1024	0/0	15/20	0/0	0/0	0/0	15/20
1334_1024	0/0	21/28	0/0	0/0	0/0	21/28
1340_1024	0/0	15/16	4/4	4/4	3/4	4/4
1352_1024	0/0	32/38	10/11	3/4	3/4	16/19
1360_1024	3/3	9/9	2/2	2/2	0/0	5/5

Figure 7. Table of classification results. Listed as [number correct]/[total].

5. ACKNOWLEDGMENTS

My thanks to my advisor, Gordon Wetzstein, for his patience and guidance. Thanks also to JB Boin for his implementation of SIFT_MATCH, which was my building block for the matching functionality. Finally, I would like to acknowledge the creators of VLFEAT, whose library I used.

6. REFERENCES

[1] Mihir Pendise and Yiwei Wang, 2012. Automated Coin Detection on Android Phone. EE368 Class Project. Unpublished

[2] Michael Digman and Christian Elder, 2013. Foreign Bill Detection, Identification and Currency Conversion Using Sift. EE368 Class Project. Unpublished.

[3] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* 60, 2 (November 2004), 91-110. DOI=http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[4] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (June 1981), 381-395. DOI=http://dx.doi.org/10.1145/358669.358692

[5] Derek Bradley and Gerhard Roth. 2007. Adaptive Thresholding using the Integral Image. *Journal of Graphics, GPU, and Game Tools* 12, 2 (2007), 13–21. DOI:http://dx.doi.org/10.1080/2151237x.2007.10129236

[6] Michael Digman and Christian Elder, 2013. Foreign Bill Detection, Identification and Currency Conversion Using Sift.