Coin Detection System for COMS4036

Liam Pulles, student number 855442

October 4, 2017

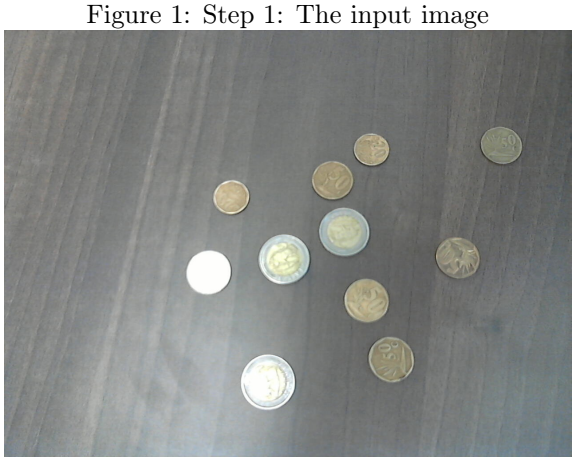
1 Introduction

This project deals with the segmentation and classification of coins in images. In this report, I will outline the pipeline from input image to coins, as well as give a brief overview of the algorithms involved. I shall not detail the specifics of the GUI, or small steps in between the major processes which don't do significant work. I do not intend to explain the gritty detail of each algorithm, but rather to give reasons as to why I used each method in addition to a high level description.

2 Pipeline

1. Load an RGB image into memory.

Figure 1: Step 1: The input image



2. Convert the image to the HSV color space.
3. Average the saturation and value channels of the HSV image into a greyscale image.
4. Perform a moderate, constant contrast stretch on the image.
5. Do a perspective transform on the stretched image, so that coins are uniformly circular across the image.
6. Add a 100 pixel border around the transformed image, to account for window operations near the edge.
7. Perform a morphological closing, then opening on the bordered image using a radius 10 ellipse element.
8. Perform the same perspective change on the HSV image (this is a second image being worked on separately to the image in the previous step).
9. Smooth the transformed second image using a bilateral filter.
10. Perform a morphological opening on the smoothed image, using the same element as previously.

Figure 2: Step 3: Combine the saturation and value channels.

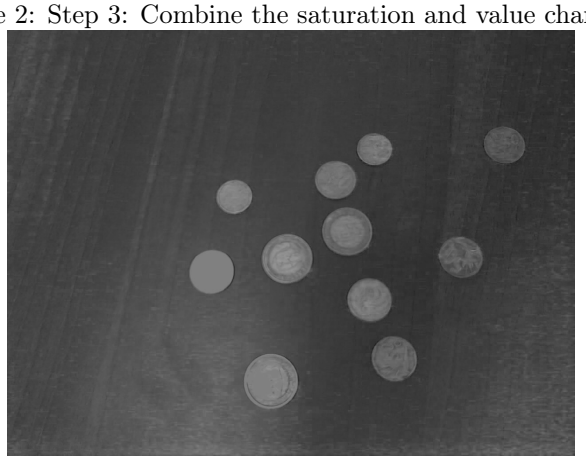
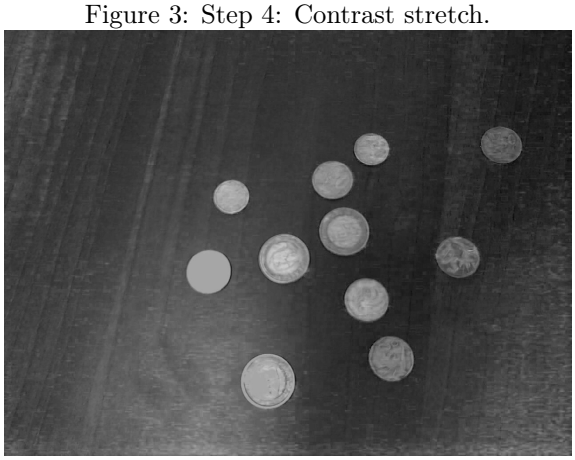


Figure 3: Step 4: Contrast stretch.



11. Perform a "material labeling" on the morphed second image.
12. Run the canny edge detector on the morphed first image, and then apply a circular hough transform to detect circles in the image.
13. Extract a feature set from each circle using the material labels found previously.
14. Classify the circle as being one of the general South African coins, or a non-coin, using a Random Forest classifier.
15. Draw a circle and relevant coin value text for each circle.
16. Return the summed values from each coin, and the highlighted image.

3 Methods

3.1 HSV color space conversion

The HSV color space can be considered a polar coordinate transform of the RGB space. The HSV space naturally separates the color attribute (hue) of each pixel from its vibrancy (saturation) and intensity (value) attributes. This is useful, because objects tend to have relatively constant hue, and are likely more easily segmented in the HSV space (relative to the RGB space). It is also often convenient to find edges in the value channel of the transformed image.

It turns out that in this case, despite conventional thinking, The hue is not particularly useful. This is because many of the pixels in the image have either very high value or very low saturation, meaning that noise (and we consider lighting variation to be noise in this context) can cause the position of the pixel in the polar coordinate space to jump across an axis perpendicular to hue, and lead to totally different values of hue.

Figure 4: Steps 4,5: Transform the image shape

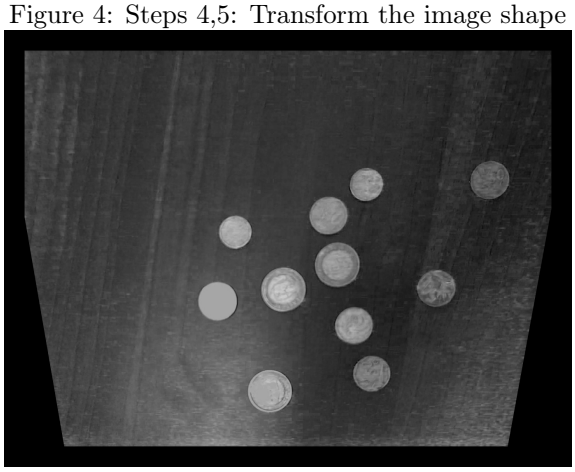
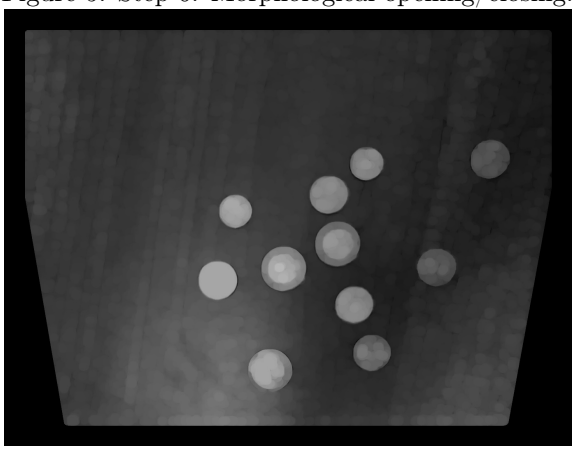


Figure 5: Step 6: Morphological opening/closing.



However, a direct equal average of the saturation and value channels of the image provides a greyscale result which leaves the coins a fair bit brighter than their surrounding pixels, which will aid edge detection later.

3.2 Contrast Stretch

This process is very simple. We subtract a fixed amount (larger than 0) from the greyscale image and scale it by a fixed amount (larger than 1). The result is an image with higher contrast, and more visible edges.

3.3 Perspective shift

A perspective shift is defined as an affine transform (a 3d matrix in this case for a 2d image). That is, it can involve a combination of shear, rotation, translation and/or scaling transforms. In our case, we wish to horizontally compress the bottom part of the image so as to make all the coins in the image more circular (as in, closer to a circle than an oval). This will aid the hough transform later. We are careful to fit the whole transformed image into the dimensions of the original image (excluding small parts on the top of the image), so as to reduce the cutting off of prospective coins. This results in a large black triangle being present on both of the horizontal sides of the image, but it doesn't harm further processing.

3.4 Morphological Operations

Morphological operations operate by shaping the contours of a single image channel. All morphological operations operate using a "structuring element", which is generally a small image in the

Figure 6: Steps 7,8,9: Pre-process the image for material labeling.

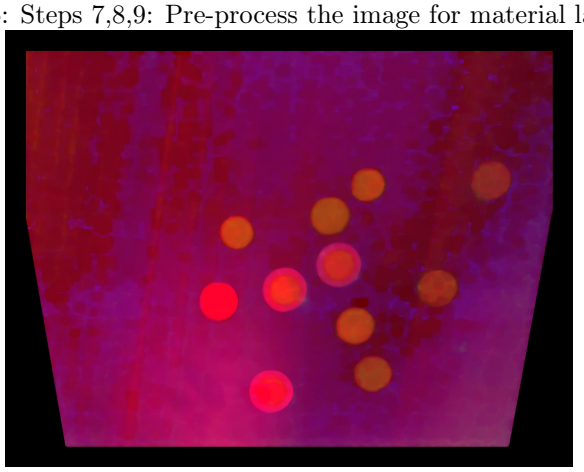
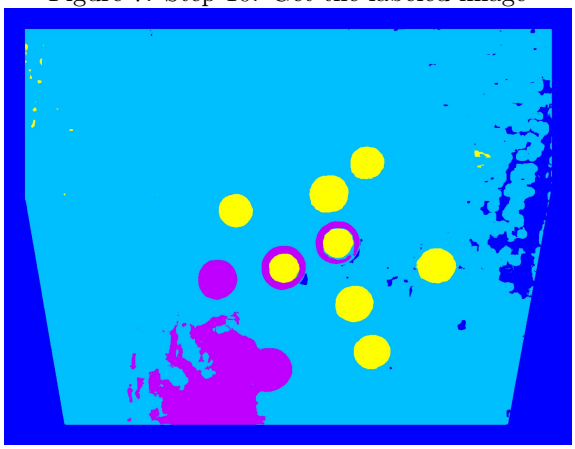


Figure 7: Step 10: Get the labeled image



form of a simple regular shape, such as an ellipse in our case.

First we define the morphological dilation. We apply a dilation on an image with a given morphological element by moving the element over each pixel in the image, and applying it when one of the white pixels in the element overlaps with a white pixel in the image. The element is "added" to the image in this case, and clamped above by white. The effect of this over the whole of the image is to expand it's white borders.

The erosion is in some sense the opposite. It is equivalent to applying the dilation to the inverse of the image, and then inverting the image back. The effect is to shrink white borders.

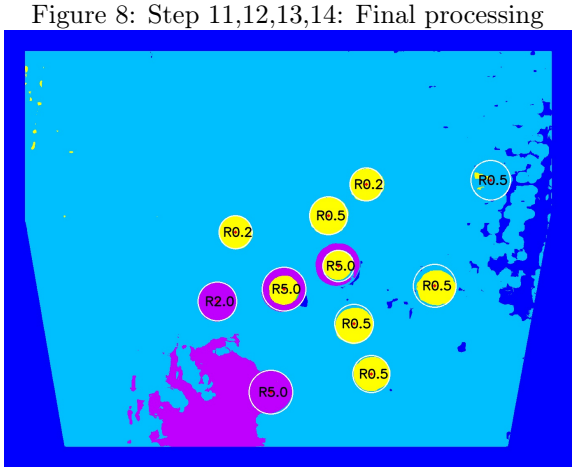
The opening of an image is defined as the dilation of the erosion of an image. The effect is to eliminate white specks in the image which are smaller than the structuring element. Applying it more than once has no effect.

The closing of an image is defined as the erosion of the dilation of an image. The effect of this is the closure of black holes in the image smaller than the structuring element, and the joining of elements which can be connected across within the span of the structuring element. Similar to the opening, applying it multiple times has no effect.

3.5 Bilateral filter

The bilateral filter is non-linear and thus not a convolution - but similar. Instead of taking the average of pixels in a window according to a set of weights, we only do so when the Gaussian of pixel differences between the central pixel and prospective pixel is sufficiently small. This has the effect of preserving edges, at the cost of speed.

Figure 8: Step 11,12,13,14: Final processing



3.6 Material labeling

This process is of my own design. My plan was to attempt a rough segmentation of the image according to what kind of material was most likely at each pixel, and to use this information in aid of classifying the coins later. At each pixel, we retrieve a probability that the pixel belongs to a chosen material, for each classified material. We than take the argument maximum to retrieve the most likely material for each pixel.

We do this with the aid of a set of multivariate normal distributions, one for each material, trained using maximum likelihood (i.e. with standard mean and variance). We use full covariance matrices, though each is an estimation - also arrived at by maximum likelihood. The extraction of relevant data for the training of each model was achieved by the use of masks, created manually for each set of materials using GIMP.

The multivariate normal distributions specifically take input from HSV pixels.

The materials are:

- Black Plastic
- Background
- Blue Pen
- Copper
- Gold
- Purple Plastic
- Ruler
- Silver

Though the segmentation in practice is far from perfect, what's important is that certain materials appear under certain coins in certain proportions - and we shall be able to use this information to our advantage in reducing the possibilities for each coin.

3.7 Canny Edge Detector

The steps used to find edges are as follows:

1. Apply a Gaussian filter to the image so as to smooth it. This is controlled internally and so the precise tuning of the kernel is uncertain.
2. Find the intensity gradient of the image in the horizontal and vertical directions using Sobel.
3. Find the gradient strength and direction for each pixel in the gradient image and bin the direction to four possible angles.

4. Perform non-maximum suppression to remove pixels which are very unlikely to be edges.

5. Perform hysteresis. Keep pixels with gradient strength above a certain threshold. Remove pixels with gradient below another certain threshold. Keep pixels which fall in between the two thresholds if they are connected to an already present edge. The lower threshold is maintained at half the upper threshold, as recommended by Canny.

The result of this process is generally nice edges, and in combination with our fairly well segmented average image, we get good circles.

3.8 Hough Transform

If we can represent a shape in an image (say a line or circle) using a set of equations in polar coordinates, then we can plot the sinusoidal function that describes the family of possible equations for each point as a set of curves on a graph. If a significant number of curves intersect at a single point in the graph, then it means the polar coordinates at that position describe a family of points in the image which are part of the same shape.

The hough circle transform in OpenCV uses a special gradient method, but it is beyond the scope of this report to describe that process.

3.9 Feature Extraction

We extract the following information for each circle (prospective coin) by using the Hough transform and material labeling:

- The radius of the circle.
- The total number of silver, copper, gold, copper plus gold, and garbage (purple plastic, ruler, blue pen) pixels (separately) within the area of the circle.
- The total number of silver, etc. pixels within a ring directly surrounding the area of the circle.
- the total number of silver, etc. pixels within a ring directly within the circumference of the circle.
- a boolean value, signifying whether expanding the radius of the circle by twenty would increase the number of circles that the current circle overlaps.

This gives a total of 17 features. The features regarding sub and super rings of the circle give an indication as to whether the circle is not filling the whole coin (consider the case of anew R5 coin having a segmented ring within) and provide indications as to differences in color between the edge and center of the image. Bronze and cooper indicate low rank coins (10c,20c,50c) and silver higher rank (R1,R2,R5).

It is worth noting that I do not know the precise relationships that these features have with specific coins. But some machine learning methods will discover the relationships for us, and allow us to ignore this fact.

3.10 Random forest classification

First, let us describe a decision tree. Given a feature set with known classifications, it is possible to find a set of conditions which can be evaluated against a given set of features in a specified order, so as to maximally reduce the entropy (uncertainty) of the set of possible classifications that would survive that set of conditions. In other words, we can use automatic methods to build a logic tree that can classify data points for us.

The problem with decision trees is that they have a tendency to over-fit. A leaf node (a single possibility of class or split of possible classes in the worst case) is often a place which is only reachable by one or two datapoints. It is quite possible, therefore, that the algorithm is then classifying a given instance by noise rather than the true signal. We would expect some error.

We can get around this by aggregating across several different decision trees for the most common (modal) answer. We can arrive at separate trees by separating the training data between the trees during the training stage.

Experiments indicate 90% accuracy on a testing set after using approximately two thirds of the training data for training and one third for testing. 10 trees were used, and no limitations on depth or leaf nodes were enforced.

The training data was created by (painstaking) manual classification of the detected circles in the training set. Errors are thus more likely to be in the last third of the training set than in the first two thirds (as the split was taken directly on the normally ordered dataset without shuffling).

4 Results

We can describe the performance of the system by the usefulness of two key processes, first by the circle detection, and second by the circle classification. Of course the first is most important, since if a coin is not picked up by the circle detector, it has no chance of getting classified. Indeed this does sometimes occur. In particular, it occurs for brown coins which have very blurry edges, which would indicate that those edges are not picked up by the canny edge detector (or at least, not a significantly large enough part of the edges to allow for detection of the coin by the hough transform).

Classification performance is around 90%, as previously explained. We can expect the algorithm to perform worse on elements where there is a class imbalance (i.e. there are fewer elements of that class in the training set than other classes) and also when there is higher variation in the datapoints in the class. This leads me to believe that the worst offenders would likely by the brown coins, since they can appear in many shades from dark brown to very shiny white (where the material detector sees them as being silver coins). That being said, 90% is quite good performance and is "good enough".

It's also worth noting that some of the error in the circle classifier could be down to my misclassification of some of the training circles. This is because the scale and marking on the coin is not always clear, and not always inferable from neighboring coins. Similarly, the initial masks for the material labeler training data are not perfect. This arises in situations where one has to, for example, choose whether a coin is more gold or more copper.

5 Conclusions

The key strength of my program is the material labeler. It provides enough information about the proportions of visible material of coins to give distinguishing power to the random forest. It does require arduous manual masking and classification of images, but the results are worth it.