COMP702 Project

Alexander Goudemond, 219030365 and Sumeith Ishwanthlal, 219006284

June 16, 2022

Please note that the code for this project can be found on GitHub at:
https://github.com/SumeithAlexOrgUKZN/COMP702_CourseProject.git

1 Abstract

Bank note recognition is an application of Image Processing, whereby a picture of a paper-based currency is captured and classified. The ability to identify and differentiate between different bills and currencies is desirable for banks, vendors and consumers. This project investigates different techniques to try to classify a digital image of a South African Rand - in denominations of R10, R20, R50, R100 and R200.

2 Introduction

A lot of the literature surrounding this work deals with counterfeiting and the equipment used to fabricate bills. One such example is mentioned by Lamsal [5], as a Fake Note Detector Machine, used by banks to quickly identify counterfeit notes. Lee et. al. [6] goes on to explain how these counterfeit machines have 4 functions: counterfeit banknote detection, serial number recognition, banknote recognition and fitness classification. These machines are capable of taking a variety of pictures, which would aid with bank note classification, including UV light capturing, near infrared capturing and recognizing magnetic components.

With regards to distinguishing counterfeit bills - 4 avenues affect currency verification: Design, Physical Dimensions, Paper Quality and Printing Techniques. Design focuses on icons that can be distinguished (unique to the currency), Physical Dimensions need to be consistent, and the Paper used needs to have the correct texture and thickness. The printing techniques used incorporated contain security features, some of which could be used to differentiate between bills. Examples of these security features include watermarks, different layers of ink (and colours), etching onto the bill, UV reacting ink and magnetic strips. Many of these security features are recognized using the Fake Note Detector machines as well.

Physical currency has 2 sides, and each side can be rotated in 4 ways - leading to 4 possible orientations for each face. Pham et. al. [7] references this, as many recognition methods rely on the bills being in the correct orientation. Ideal bills also have no creases, folds or tears - as automatic identification struggles to match abnormal samples. Other kinds of physical contributors include dirt or water-staining from common use.

Automatic Identification of bills needs to consider these drawbacks - as bills need to be pressed into flat rectangles to ensure the Fake Note Detector Machines can recognize the material. It is inside this machine that a high quality image (with tightly controlled resolution, lighting, and focal distance) is produced and processed.

Lee et. al. [6] mentions how banknote recognition focuses on currency classification by denomination, with some papers even attempting to classify notes across national currencies. Other kinds of applications may be tools/apps for consumer use.

3 Related Work

Lamsal [5] is a PhD in counterfeit detection, which builds a Fake Note Detector Machine to attempt to extract some special printing features (such as Watermarks and Magnetic Strips). Other kinds of work include Pham et. al. [7], who looks at banknote recognition using similarity maps, Wang and Liu [8] who propose a neural network (NN) to predict face and values of a note, as well as Abu Doush and Al-Btoush [1], who apply Scale Invariant Feature Transform (SIFT) onto images for a cellphone app, to classify both coins and bills.

With regards to pre-processing of images, Lamsal [5] uses a Gaussian Filter and increases brightness to attempt to remove any dirt. Lee et. al. [6], on the other hand, contrasts Weiner filtering, Median filtering, grey level reduction, histogram equalization, nearest neighbour interpolation and grey scale conversions as other possible pre-processing options.

To assist their NN, Wang and Liu [8] attempt to first align the bills in the correct orientation and side, using a Principal Component Analysis technique alongside largest contours.

Abu Doush and Al-Btoush [1] crop the image, apply a Discrete Wavelet Transform (DWT) and compress via nearest neighbour interpolation. Thereafter, key points are identified (extrema relative to neighbour) and then converted to a descriptor that can be compared against a reference set.

Regarding Segmentation, Lamsal [5] and Pham et. al. [7] employ an edge detection algorithm to separate foreground and background. Lamsal [5] then applies a scan-line algorithm to highlight desirable areas, whereas Pham et. al. [7] subsamples this and generates a similarity map. Lee et. al [6] employs a serial number recognition, to try separate non-currency pictures early on, in an attempt to discard fake notes immediately. Other techniques proposed by Lee et. al. [6] include corner detection, least square methods, fuzzy systems and component labelling in the YIQ color space.

Looking at feature extraction, Lamsal [5] converts an RGB image into a CIE Luv image (to better mimic the human perception of color), then uses 2 kinds of resources: *Color descriptors* and *Texture descriptors*. Skew, Mean and Standard Deviation are calculated from the color image (Color Descriptors), whereas entropy and correlation are calculated from the co-occurrence matrix (Texture Descriptors).

Lee et. al. [6] mentions several options for feature extraction, which are shown in Table 1 below.
Wang and Liu [8], and Abu Doush and Al-Btoush [1] use SIFT to generate a feature vector.

Pham et. al. [7] mentions how other techniques include finding the Quaternion Wavelet Transform (QWT) coefficients of an image, and then extracting features using Generalized Gaussian Density (GGD). Both mention how extracting values from color channels provides more information, but is more computationally expensive.

Group	**Method**	**Example**
Physical	Bank note dimensions	(512, 1024)
Colour	Colour Information	(RGB, HSV, CIE, etc.)
Edge Detection	Edge Information	(Canny, Prewitt, Sobel, etc.)
Statistical Measures	Histogram Analysis	(Correlation, central moments, mean, etc.)
	Grey level Co-occurrence Matrix	///
Textural Features	Local Binary Pattern extraction	///
	Scale-Invariant Feature Transform or Speeded Up Robust Features extraction	///
	Optical Character Recognition extraction	///
	Region Of Interest extraction	///
Dimensionality Reduction	Principal Component Analysis	///
	Linear Discriminant Analysis	///
	Compressed Sensing	///
Neural Networks	Genetic Algorithm	///
Visualization Tools	Similarity Maps	///
Transformations	Discrete Wavelet Transform	///

Table 1: Table adapted from Lee et. al. [6], showing feature extraction techniques

To classify bills, Lamsal, S [5] uses a Neural Network (NN) in a Region of Interest (ROI) as face/icon detection. Pham et. al. [7] uses a similarity map and a corresponding binary mask to classify a bill, invariant to side or orientation. This is achieved using a Principal Component Analysis algorithm alongside a K-means algorithm.

Pham et. al. [7] also mentions how some papers can use texture characteristics and a Hidden Markov Model (HMM) to classify multiple banknotes.

Abu Doush and Al-Btoush [1] mention how a Feed-Forward Back Propagation Neural Network (FFBPN) could be used. Alternatively, they highlight how an Ensemble Neural Network could be used, as well as a HMM, Support Vector Machine or Vanilla NN.

4 Methods and Techniques

The authors have designed a python program, which has its own GUI. The user is given the choice to predict a bill immediately, or experiment - to visually compare and contrast techniques in real time. The environment combines several windows, a label (on the main GUI window) and occasionally the terminal (in the user's IDE, or elsewhere).

In the experimentation section, the user has access to 13 buttons. Some buttons provide additional menus and choices as well. Where applicable, the user can see the changes being made, or save them to a dedicated folder in the same directory as the program. This functionality allows the user to experiment and interact with the work if they choose - or recreate the environment in a different way.

There will be a README document provided with the software, including dependencies and required files and directories.

The focus of this project is to use several techniques to try and classify an image, from a data-set. The techniques include using 3 Machine Learning Models, and 7 strategies for template matching. The templates were created from the provided data-set, and predictions made using a "Hit Feature Vector".

The algorithm included 1) generating the reference material, 2) extracting the features for an individual image and 3) calculating the shortest Euclidean Distance between each image feature and the corresponding reference feature. This smallest distance should indicate a nearest neighbour, and so a 'tag' is generated for that 1 feature, to demonstrate this. These tags form the Hit Feature Vector, which is implemented as a list of strings.

The features generated included 6 features along colour channels (red average, green average, blue average, red mode, green mode, blue mode) as well as 13 Haralick features for greyscale images. At the end of the algorithm, the Hit Feature Vector is sorted, and the most occurring flags/hits indicate the predicted bill.

The machine learning models used are variations of a Support Vector Machine with a Linear Kernel. Inputs used are Colour Features, Haralick Features and Hu moments.
For a detailed description of the techniques used for template matching, please refer to table 3.

Bank Note Image Preprocessing and Enhancement

Kindly note that in addition to the provided data-set the authors found 5 additional pictures online, from Daniel, L [2]. These show a special kind of Rand with a young Mandela on it. These pictures were included in the data-set

Using the provided data-set and the additional pictures found online, the authors generated a "messed up data-set", which can be found in the folder "MessedUp_Notes_DataSet". The intention behind this is to have a collection of images that are 'damaged', which we would then try fix and classify.

Every image from the data-set was cloned and a combination of random elements were applied to it. These elements were chosen at random and and include:

- Either being brightened or darkened by a random value, ranging from 30 to 110
- Having some noise applied to an image. These noises included Gaussian Noise, Salt and Pepper Noise, Poisson Noise, and Speckle Noise
- Rotating at a random degree from 5° to 360°

The authors attempted to add code to grow and shrink the images as well, however this proved too computationally intense.

For pre-processing - the system first checks that all dependencies are present. These dependencies include identifying the notes data-set, resizing the data-set, generating the additional directories of manipulated images, writing the reference material to a directory and ensuring an initial set of "messed up images" exists.

For consistency, the MessedUp Images are only generated once, and all experiments conducted on the same set. However, the user could override these changes if they would like.

All of the checks run at the beginning of the program, and update only if the content does not exist. This is optimal, as we can save time for the user.

To identify resizing needs, the authors extracted information from the data-set to identify the variation in image size. 29 distinct image dimensions were found out of 55 images. Table 2 shows the results from the data-set.

	Absolute Minimum	Absolute Maximum	Average Value
Height	99	651	423
Width	179	1200	846

Table 2: Table showing the information from the data-set

From table 2, we can see that the average values translate to a ratio of 1:2. So the authors proposed a resize value that keeps this ratio: (512, 1024). The user may modify this, however the authors wanted a value between the extrema.

For the processing of images, some additional processes are required To analyse the messed up images. These processes include: Realignment of an image, Colour Correction and DeNoise. Where necessary, additionally processing may happen (depending on prediction chosen)

Realignment of an image is required because some of the images in the messed up data-set are rotated. In order to realign an image, the image is first converted to a greyscale image, then a binary image. From here, the contours of the image are then determined. The orientation/angle of the image is then calculated using Principal Component Analysis (PCA), and the image is rotated accordingly. A new contour image is then recreated in order to crop the image. This final step is essential because when an image is rotated it is padded with zeroes - which can distort the feature vectors.

Colour Correction is done by using histogram equalisation, either on a Colour Image or a Greyscale Image. The Denoise process was done using openCV with a function called fastNlMeansDenoising(). This function was chosen because it is optimized to run in python.

For processing of greyscale images, the same processes are used.
Figure 1 shows some examples of images created by the system. These explorations of processing included investigating histogram equalization, negative images, masks, smoothing, sharpening, converting the image to binary, applying a morphological operation and applying a transformation. The user may recreate these results, or explore other combinations, using the experiment button of the application. The best techniques appeared to be: re-alignment of an image, through PCA and cropping, applying histogram equalization to fix colour, and using a fast average denoising function.

Bank Note Segmentation

Many segmentation techniques were explored by the authors. Figure 1 contains some examples of the images created by the system. The user may recreate these results, or explore other combinations, using the experiment button of the application. The best segmentation techniques seen incorporated canny edge detection, contour edge detection and K-Means clustering.

Bank Note Features Extraction

For feature extraction, the authors initially explored template matching with colour features. From the resized photos, 11 versions of each bill were collected together (for 55 pictures in total) Next, for each picture, the 3 colour channels were extracted, whereby the average value and mode for each channel were found. After doing this 11 times, for each bill, the results are aggregated - each bill has 6 features: Average for each colour, and Mode for each colour. Variances are included as the range of the 11 data-points, divided by 2, as well.

This collection of numbers is difficult visualize, so the authors created Figure 6 and 7, which shows the corresponding features, visually on a number line. From Figure 6, it seems that the colour features can be used for template matching, as no patterns seem to appear. However, when we trace the connection between the features, as shown by the connected lines, we notice that the R20 and R200 have incredibly similar shapes. This suggested that classification between R20 and R200 may be difficult, using colour features.

Figure 7 is similar to figure 6 - however a different source for the reference material is used. The source used for this reference material is a collection of Histogram Equalized images. The trends, focussing on the blue line in figure 7, suggest that a R20, R50 and R200 may be confused as well.

Bank Note Classification

The following techniques were used to classify an image

6 Colour Features

- ”Raw” Colour Features
  - Here, the original Resized Data-Set is used to generate a set of reference feature vectors
  - Each image is processed, as mentioned in the above section
  - the shortest Euclidean Distance is then calculated for each feature, across the reference/template
  - Shortest distance represents a nearest neighbour, and is a hit
- Histogram Equalized Colour Features
  - This procedure is similar to ”Raw” Colour features, however it generates reference feature vectors by first applying a Histogram Equalization to the original data-set
  - This strategy was used so that there is no disparity between the colour channels of the test set and the reference set
  - Shortest Euclidean Distance and a Hit Vector are generated

13 Greyscale Features

- ”Raw” Colour Features
  - Similar to Colour features, the reference material here is generated from greyscale conversions of the original data-set
  - Shortest Euclidean Distance and a Hit Vector are generated
- Histogram Equalized Greyscale Features
  - Similar to Colour features, the reference material is generated from a set of Histogram Equalized Images
  - Shortest Euclidean Distance and a Hit Vector are generated
- Histogram Equalized Contoured Greyscale Features
  - Here, an attempt was made to extract a Region of Interest (ROI), using complete contours. The original images are converted to greyscale, the ROIs are found, and the reference material generated
  - The test images undergo the same processing as normal, however now an additional step of contouring is applied
  - Shortest Euclidean Distance and a Hit Vector are generated
- Histogram Equalized Canny Edge Negative Features
  - Here, an attempt was made to convert a picture into a form where similar pictures had similar outlines. Canny Edge Detection was applied to a histogram equalized image, and the result was inverted. Taking the negative of the images is important, because we want the majority of the image to contain non-zero values
  - The test images undergo the same processing as normal, however now an additional step of Canny Edge Detection and Negation is applied
  - Shortest Euclidean Distance and a Hit Vector are generated
- Histogram Equalized 5-Means Cluster Features
  - This is an attempt to try and generate similar zones inside an image. The hope is that similar images will have similar clusters, which should translate into the feature space.
  - The images undergo the same processing as normal, however now an additional step of 5-Means Clustering is applied
  - Shortest Euclidean Distance and a Hit Vector are generated

In addition to the above, Linear Support Vector Classification models were created to distinguish between the different bank notes. These classification models were inspired from [3] [4]. These models were created in a python notebook which can be found in "machine_learning_model.ipynb".

The authors chose to use a Jupyter Notebook for this, and not include the models in the main program. This was because variation of parameters and model design is easier to implement and experiment in a Jupyter Notebook. The output of the cells are also retained - which is beneficial to a user wanting to read the cells without running the code. However, running the notebook will classify the messed up images.

There were three different models which were created. The Linear Support Vector Classification model uses different features as input in order to classify the different pictures. The three different models are mentioned here:

- There is a model called modelHaralick, which utilises Haralick features to classify an image
- There is a model called modelColourChanels, which utilises the different colour channels to classify an image
- There is a model called modelHuMoments, which utilises the different Hu moments of an image to classify an image

The different models are saved as .sav files and one can easily use the saved models using the pickle package and the load function: pickle.load(open(filename, 'rb')).

5 Results and Discussion

When using the Program, the user has the option to predict either an individual image or the entire folder of Messed Up Images. There are 7 Radio Buttons available for either individual image predictions or bulk image predictions (14 in total). These 7 predictions are listed and summarized in the table 3.

	Prediction Description	**Pictures to Reference**	**Overall Accuracy (%)**	**Approximate Time for Bulk Prediction (H:M:S)**
6 Colour Features	"Raw" Colour Features	Notes DataSet	19.0	00:01:46
	Histogram Equalized Colour Features	HistEqColour Notes DataSet	21	00:01:45
13 Greyscale Features	"Raw" Colour Features	Notes DataSet	20.6	00:02:11
	Histogram Equalized Greyscale Features	HistEqGrey Notes DataSet	17.8	00:02:07
	Histogram Equalized, Contoured Greyscale Features	Contoured HistEqGrey Resized Notes DataSet	16.5	00:02:07
	Histogram Equalized Canny Edge Negative Features	Canny Negative HistEqGrey Resized Notes DataSet	20.4	00:02:010
	Histogram Equalized 5-Means Cluster Features	KMeansCluster HistEqGrey Resized Note DataSet	24.5	00:02:10

Table 3: Table showing the techniques used and the corresponding Accuracies

T In addition to the above, the 3 Machine learning Models (modelHaralick, modelColourChanels, modelHuMoments) produced the following results.

- modelColourChanels produced an accuracy of 50%.
- modelHaralick produced an accuracy of 31.82%.
- modelHuMoments produced an accuracy of 25%.

Because the accuracy's are low, consistently, regardless of the technique - we suspect that the problem is our Messed Up Data-Set. Tables 4-10 summarize the predictions of the template matching. The diagonal lines in the confusion matrices indicate correct predicts.

Figure 6-8 are visual tools to show the relationship between different features for each bill. Figure 6 suggested that R20 and R200 notes would be confused often, and in the colour results shown in Tables 4-5, this trend may appear to be true.

A similar argument regarding R20, R50 and R200 notes from Colour Channels of Histogram Equalized Images is present in tables 4-5 as well.

Figure 8 shows the pattern forming across the greyscale Haralick features. Because these 13 features vary in magnitude, it is only necessary to design 1 figure. However, it too suggests that bills may be confused. Tables 6 - 10 show a tendency to classify things as R20 and R20, in particular.

When creating the messed up data-set, we believe that many of the key features (used to classify the images) were lost in the process. This had a major negative impact on the performance of the classification techniques.

If we refer to tables 4 - 10 carefully, we can see that the R10 hits take (up to) half of the total hits, in most of the systems.

The specific aspects of the Messed Up Images, may be as a result of the colour changes. Manipulating the Brightness and Darkness of an image makes it difficult to automatically fix - and one of the best strategies is to employ histogram equalization. However, Figures 3 - 6 show a loss of quality and a variation in the colour channels.

It is likely that the Messed Up Data-Set we used was too badly damaged to reliably predict. However, the 50% accuracy in the colour channel machine learning model may indicate that the field of research to benefit from this is Machine Learning and not Template Matching.

It is worth recreating the environment, with different parameters for the Messed Up data-set to see if the results are consistent.

Other observations:
The Realignment process for greyscale images is same as the realignment process for colour images, however the realignment of a greyscale images are of a poorer quality than the realignment of a colour image. This may be due to the loss in information as a result of rotation and cropping.

6 Conclusion

There are several possible explanations for this poor performance. One possibility is the in the characteristics of the Messed Up images. These pictures contain a lot of undesirable features, which may be exaggerations or extreme examples of real world images. For example, a combination of brightness, poisson noise, and a rotation around an angle could result in Figure 3 or 4. As can be seen by the image, histogram equalization may 'dull' the natural colours, and attempting to remove noise results in blurring - affecting quality. And images that contain a lot of noise and brightness changes are mis-categorized the most. This is clear in the case of Figure 4, as the Noise is poorly removed by the system.

Alternatively, the original data-set may have been too small, with too much variation amongst the image dimensions. The authors suspect that some images had poor resolution, and so the pixelation upon resizing may have harshly affected the feature vectors. Another consideration is that the variation in objects on each bill may conflate the results.

Another possibility is that the processing techniques could have removed the nuances amongst the data-set. With reference to Figure 3, the colour channels after processing are almost always entirely distinct from the original. In some cases, like in Figure 4, the processed image is barely an improvement because of the original damage done.

Other possibilities include too few features, the code implementation itself, a lack of a tie-breaker rule, or even inadequate features.

Future work may include recreating the environment with a more strictly controlled set of messed up images.

Alternatively, experiments could be focused on finding a colour fixing strategy that re-aligns an image with the 'natural' colours present in a South African Bill. Histogram Equalization may remove these distinct attributes, so a more specialized algorithm may need to be developed.

A larger set of image may need to used, from only 1 time slice (no variation in the bill's icons)

Finally, the reader may want to use different Neural Networks to recreate the environment, such as a Convolotional Neural network.

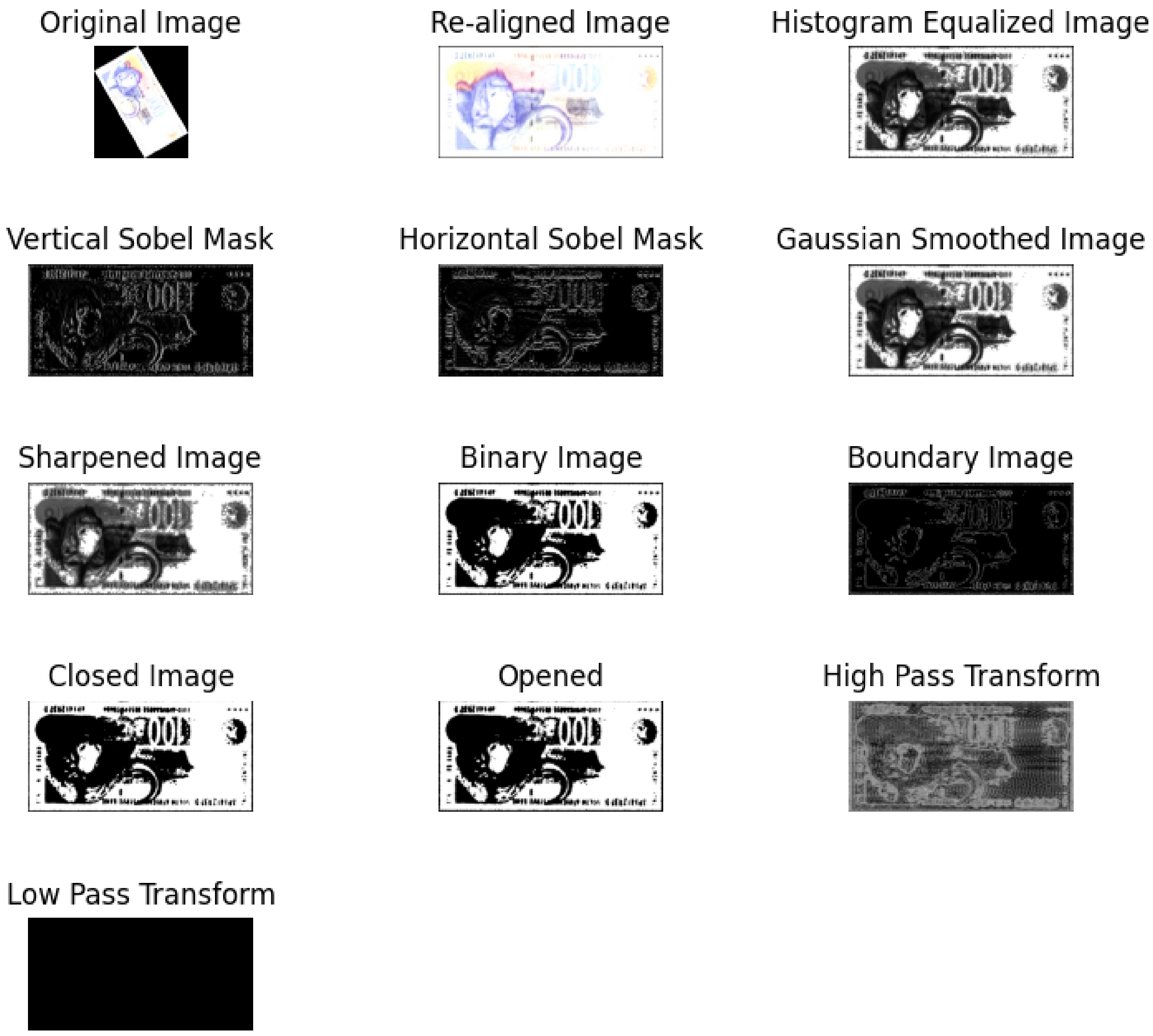

Figure 1: Different Processing Options Considered

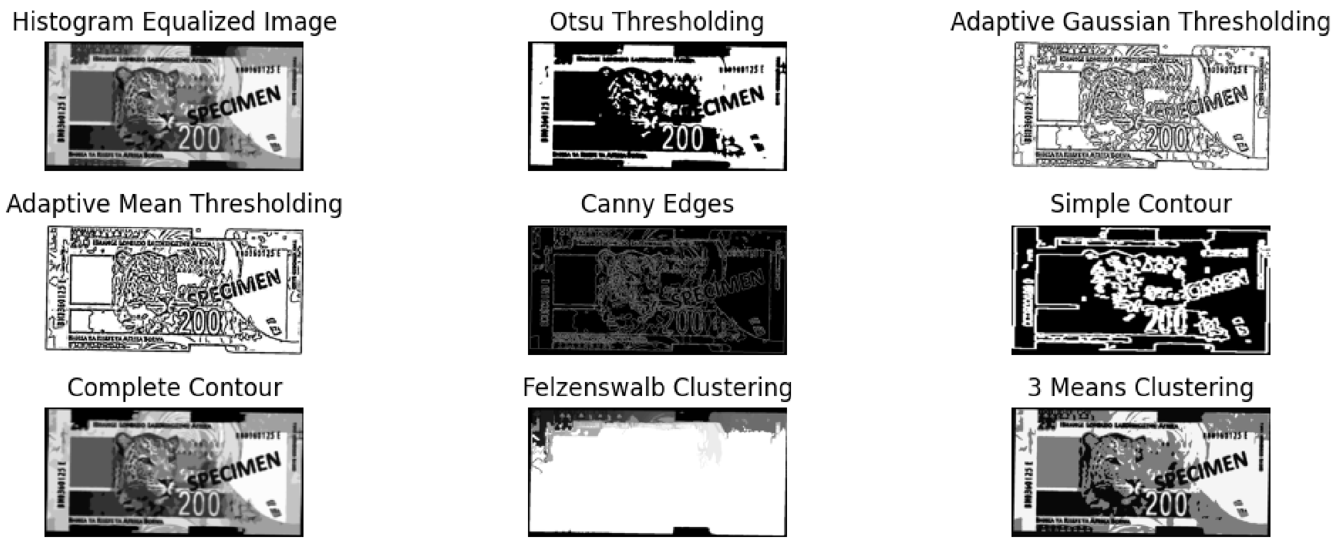

Figure 2: Different Segmentation Options Considered

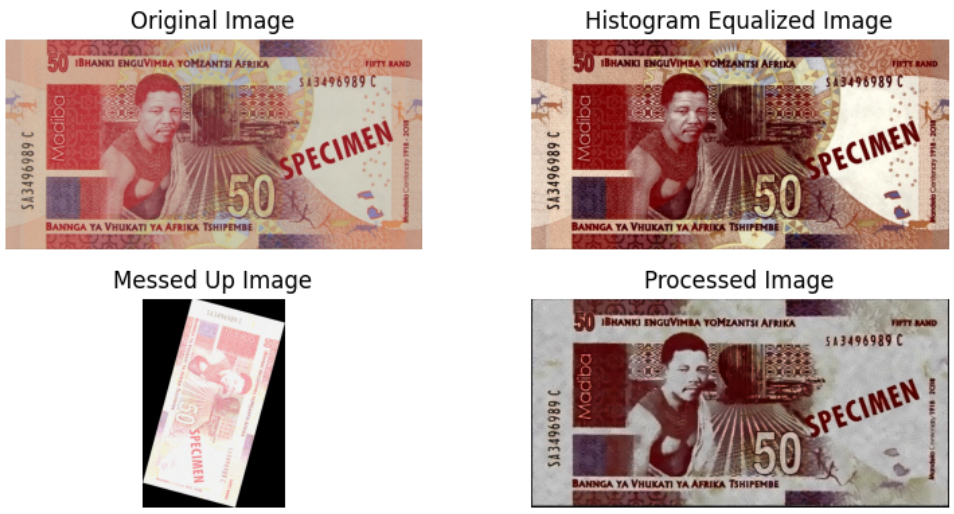

Figure 3: Comparison between Colour Channels and Processed Images

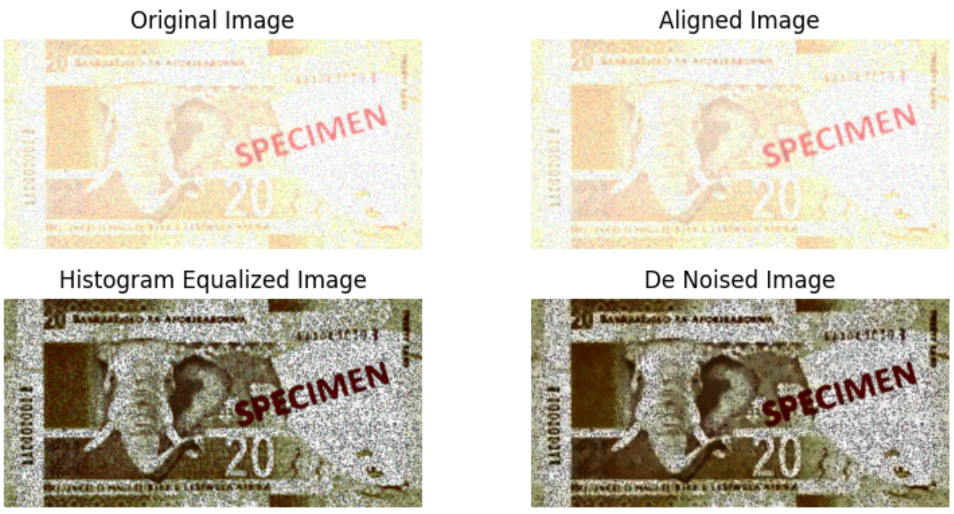

Figure 4: Comparison between Colour Channels and Processed Images

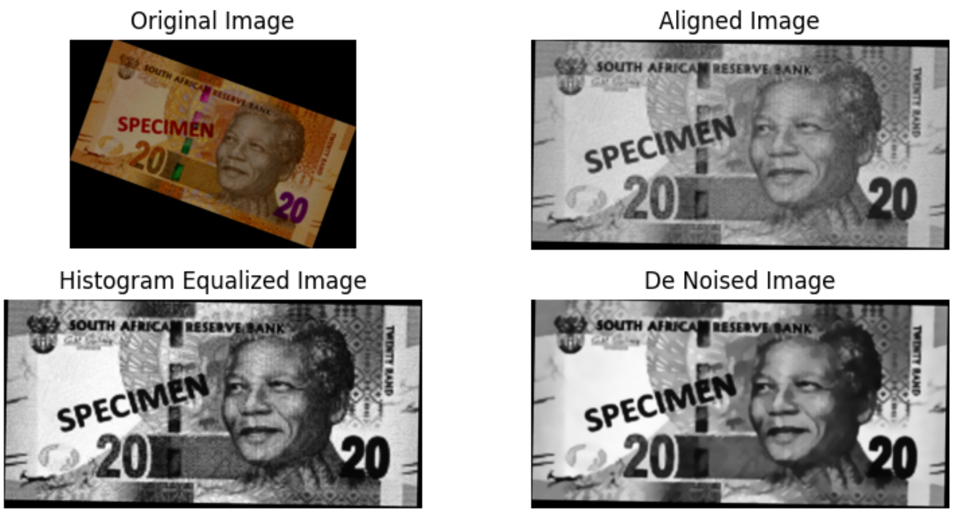

Figure 5: Comparison between Greyscale Processing and change in Whitelight Channel

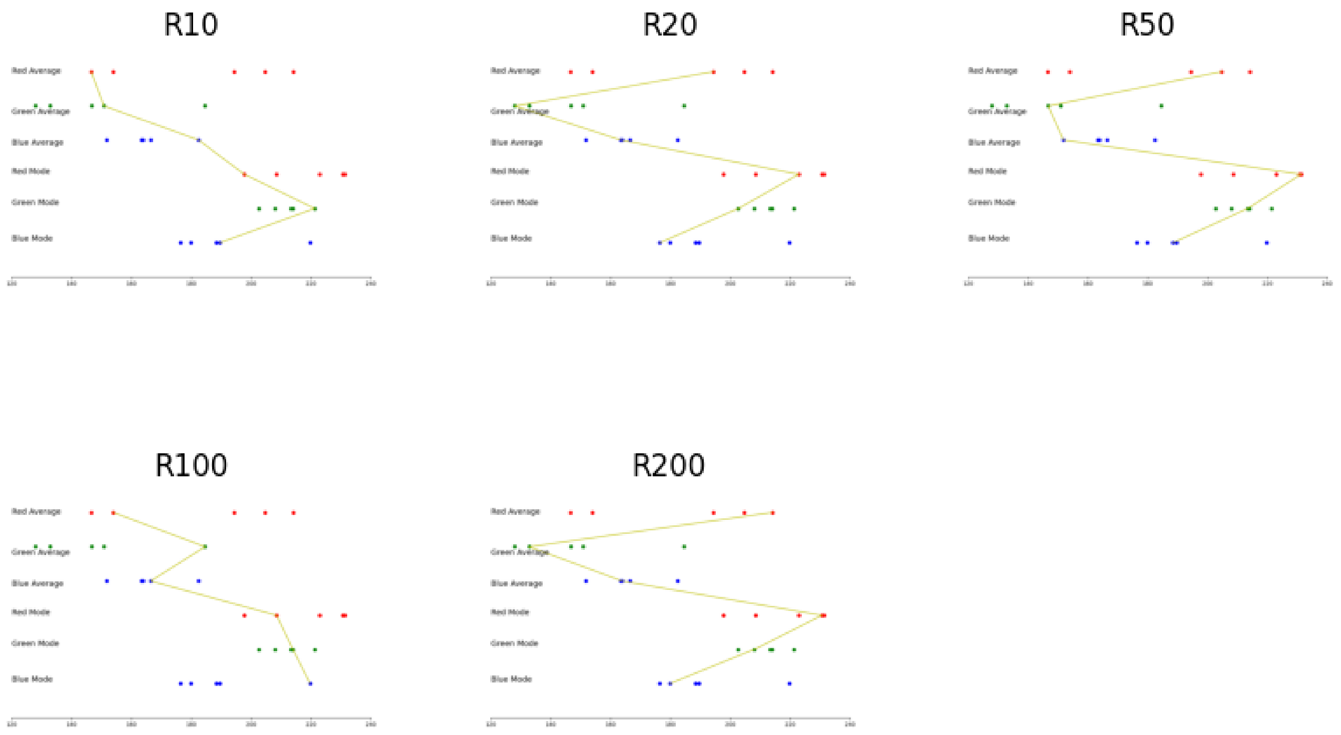

Figure 6: Connected Lines showing the Raw Colour Features for each bill

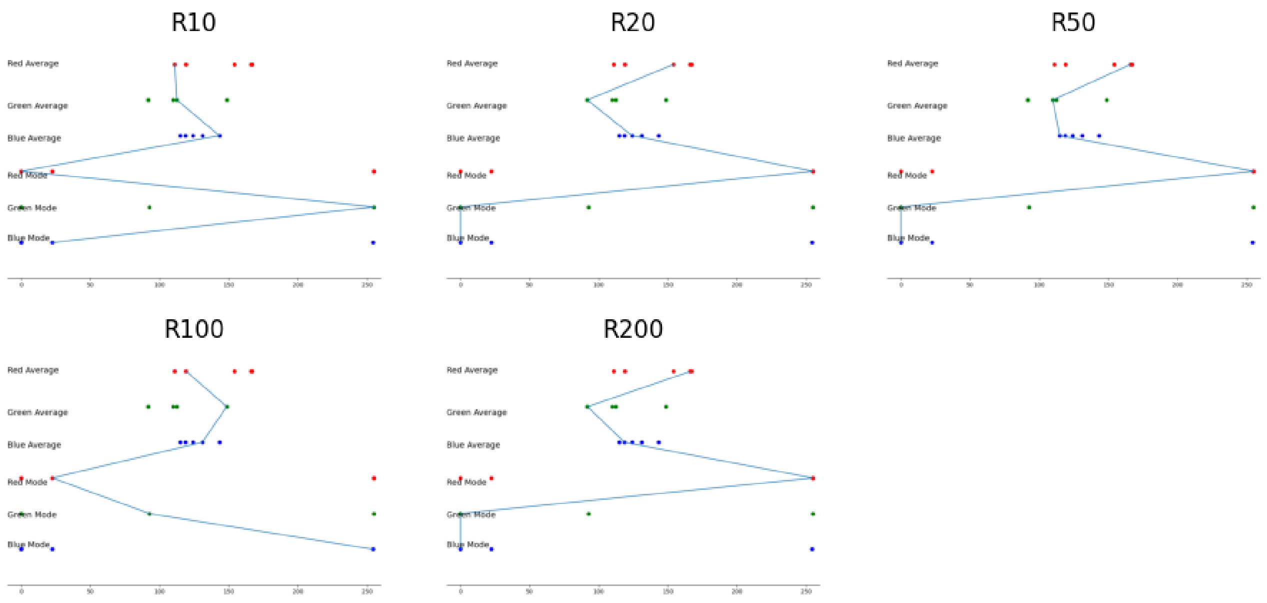

Figure 7: Connected Lines showing the Histogram Equalized Colour Features for each bill

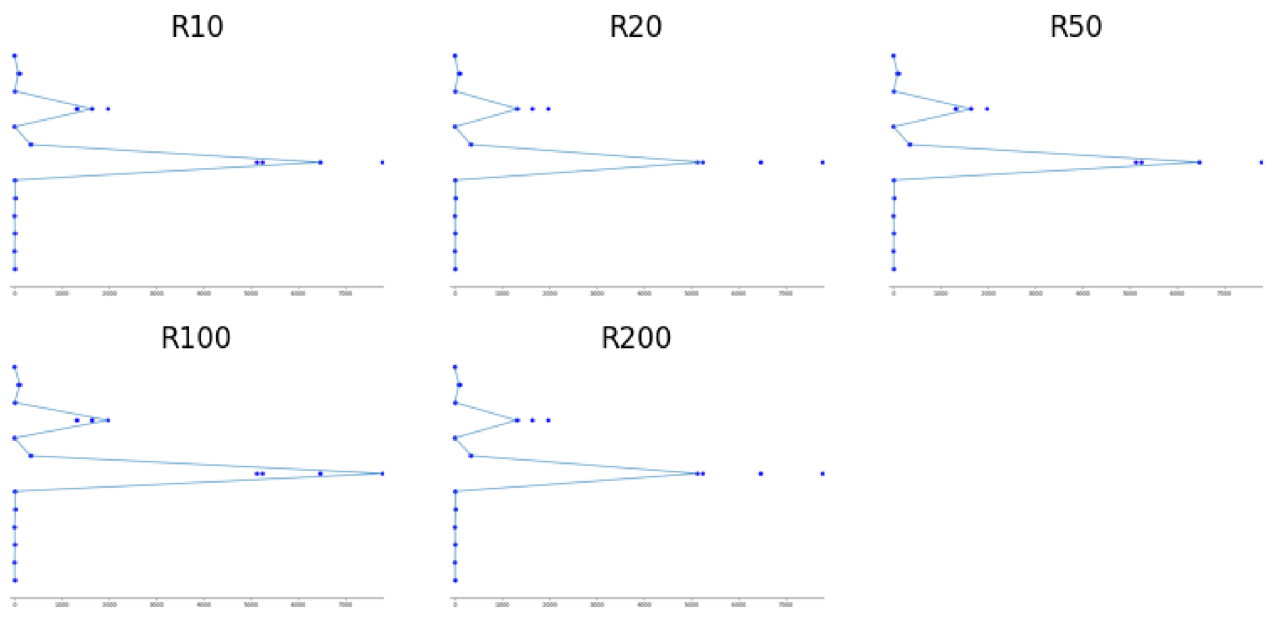

Figure 8: Connected Lines showing the Haralick Greyscale Features for each bill

	R10	R20	R50	R100	R200
R10	25	11	11	3	16
R20	29	4	17	14	2
R500	29	9	16	9	3
R100	18	10	17	8	13
R200	32	8	13	9	4

Table 4: Confusion matrix showing Bulk Simple Colour Hits (300 total)

	R10	R20	R50	R100	R200
R10	32	12	10	9	3
R20	32	9	0	22	3
R500	19	13	7	23	4
R100	23	13	15	13	2
R200	25	12	7	20	2

Table 5: Confusion matrix showing Bulk Histogram Equalisation Colour Hits (300 total)

	R10	R20	R50	R100	R200
R10	90	14	0	0	39
R20	106	1	1	0	35
R500	103	0	14	12	14
R100	97	2	1	0	43
R200	88	2	11	0	42

Table 6: Confusion matrix showing Bulk Simple Greyscale Haralick Hits (715 total)

	R10	R20	R50	R100	R200
R10	78	2	12	11	40
R20	106	1	1	0	35
R500	101	10	5	1	26
R100	98	0	2	0	43
R200	77	0	23	0	43

Table 7: Confusion matrix showing Bulk Histogram Equalization Haralick Hits (715 total)

	R10	R20	R50	R100	R200
R10	1	70	0	51	21
R20	8	65	10	31	30
R500	0	77	5	39	22
R100	10	66	5	25	37
R200	1	63	5	52	22

Table 8: Confusion matrix showing Bulk Histogram Equalization Contour Haralick Hits (715 total)

	R10	R20	R50	R100	R200
R10	58	30	1	53	1
R20	67	51	0	25	0
R500	72	39	0	32	0
R100	64	41	1	37	0
R200	70	35	6	32	0

Table 9: Confusion matrix showing Bulk Histogram Equalization Canny Negative Haralick Hits (715 total)

	R10	R20	R50	R100	R200
R10	122	0	0	14	7
R20	114	0	13	1	15
R500	123	0	16	4	0
R100	111	0	1	19	12
R200	113	7	0	15	8

Table 10: Confusion matrix showing Bulk Hist Equalization 5-Means Clustering Haralick Hits (715 total)

References

[1] Iyad Abu Doush and Sahar AL-Btoush. Currency recognition using a smartphone: Comparison between color sift and gray scale sift algorithms. *Journal of King Saud University - Computer and Information Sciences*, 29:484–492, 10 2017. doi:10.1016/j.jksuci.2016.06.003.

[2] Luke Daniel. Nelson mandela centenary: The story behind the new madiba money, 07 2018. URL: https://www.thesouthafrican.com/news/nelson-mandela-centenary-the-story-behind-the-new-madiba-money/.

[3] Gogul Ilango. Texture recognition using haralick texture and python, Dec 2016. URL: https://gogul.dev/software/texture-recognition.

[4] Gogul Ilango. Image classification using python and scikit-learn, Jan 2017. URL: https://gogul.dev/software/image-classification-python.

[5] Shaurav Lamsal. *Counterfiet Paper Banknote Identification based on Color and Texture*. PhD thesis, 11 2015.

[6] Ji Lee, Hyung Hong, Ki Kim, and Kang Park. A survey on banknote recognition methods by various sensors. *Sensors*, 17:313, 02 2017. doi:10.3390/s17020313.

[7] Tuyen Pham, Young Park, Seung Kwon, Kang Park, Dae Jeong, and Sungsoo Yoon. Efficient banknote recognition based on selection of discriminative regions with one-dimensional visible-light line sensor. *Sensors*, 16:328, 03 2016. doi:10.3390/s16030328.

[8] Peng Wang and Peng Liu. Invariant features extraction for banknote classification. *Proceedings of the 11th Joint Conference on Information Sciences (JCIS)*, 2008. doi:10.2991/jcis.2008.46.