Android Application to Assist Visually Impaired with Outfit Coordination

Noha Kareem
British University in Egypt
El Sherouk City,
Cairo, Egypt

Abeer Hamdy
Electronics Research Institute,
British University in Egypt
Cairo, Egypt

Khaled Nagaty
Ain Shams University,
British University in Egypt
Cairo, Egyp

ABSTRACT
The paper presents an android based application to assist visually impaired with outfit coordination process. The application helps them to be independent in their decisions while shopping or getting ready for the day. The application matches apparel image input with user's previously saved closet items, then provides the user with the possible matching item suggestions. The matching process is done based on the item outline and dominating colours.

In order to develop this application five main components have been developed: 1. Two Region of interest (ROI) extraction components facilitating feature and colour extraction; 2. An outline detection component to determine whether an item is a top, skirt or a pair of trousers; 3. A colour recognition component to extract dominant colours of an apparel image; 4. A descriptive colour verbal feedback; 5. A matching component based on item outline and colours to coordinate with others in the user's closet. The output to the user includes the descriptive colours, communicated via audio, indicating the 3 dominant colours using their descriptive names - selected from a pool of 581 different colour shades. Other output is the outline details of the matching items.

The application was developed and tested on Android 2.2 (Froyo) and the results showed that the ROI extraction and outline detection components perform well. The colour recognition and descriptive name generation modules outperforms the current ones.

Keywords
Colour Blindness, Visual Impairment, Android, Computer Vision, Outfit Coordination, Clothes Matching, Apparel.

1. INTRODUCTION
There exist 285 million individuals globally who suffer from visual impairment, as estimated by the World Health Organisation (WHO); around 13.7% of which are blind [1]. Visually impaired, including colour blind, individuals suffer from lack of autonomy in their selection of clothing where they must rely on other opinions. Such a simple task, outfit coordination, that is rather intuitive for many non-visually impaired, is non-autonomous, and may cause stress, to the ill-sighted persons [2].

Additionally, even some visually healthy people struggle with outfit coordination and appreciate fashion advice. A survey, of 22 participants – 9 of which had visual impairments (impairments were not further specified), revealed that 100% of those visually impaired, and around 77% of normally-sighted people, welcome help in outfit coordination [2]. Moreover, fashion retail stores, especially through their ecommerce landmarks, are highly concerned with returning customised results to their shoppers, and thus need automated outfit coordination to encourage a shopper to purchase a full outfit, recommended by the store [3].

To furnish such needs of an automated solution, this paper proposes an Android-based outfit coordination application that: 1. Classifies an item in terms of its dominant colours and its outline (top or bottom, the latter further classified into skirt or a pair of trousers). 2. Provides means for descriptive feedback regarding the item's dominant colours. 3. Matches the item based on outline and colour to other items in the user's closet, which is a database of apparel images on the application-hosting device. Outline matching is held by returning results with items of the opposite outline to the input; for instance given a skirt, a top is returned. Colour matching is offered through analogous and complementary matching, as opposed to merely returning items with similar colours.

The paper is organized as follows: Section 2 introduces literature review; section 3 introduces the proposed application. Section 4 presents the development environment, while sections 5-9 discuss in detail the development and testing of the proposed system. Finally, section 10 concludes the paper.

2. RELATED WORK
Fashion is a highly subjective matter with innumerable tastes and scarce solid rules. However, outfit coordination systems have been experimented with. This section presents the recent research in the field of automated outfit coordination.

Iwata et al [4] maintained an outfit database with magazine and online stylist opinions as references, modelled on humans. To discern tops from bottoms, they segmented images according to assumed ratios with respect to the model's face utilising a face detection algorithm. Their system matches the user top (or bottom) input against the outfit images in the database yielding a matching bottom (or top) as a suggestion. This process seems advantageous in providing the user with real life suggestions, possibly recognising fashion trends.

Rose's [5] helped the visually impaired to allocate clothing items in their closets via RFID (Radio Frequency Identification) technology. The system followed a client-server architecture offering three interfaces to SQL-based queries: One designed for the fashion consultant; another for administrator use and then an interface for the closet users. Selective "get" operation retrieved apparel from the user's closet, purging inappropriate items such as stained and weather-inappropriate clothing. The returned set is then matched together via cross product to form potential outfit suggestions that are then filtered by fashion rules fed by experts.

Paisios et al [6] used a standard neural network and a Siamese neural network to determine whether a pair composed of a shirt and a (patterned) tie produces a match to assist the visually impaired with the matching problem.

Hsu et al [3] developed a system that provides similar matches to a given input item to offer custom-recommended shopping options. The system receives an input image of an item and returns similar items in terms of outline, colour and texture,

each with different significance–based weight. They first separated the item from its background through a binarisation algorithm (further mentioned in the *ROI method 2* discussion, section 5); then categorised the pixels into prioritised classes that form the image's colour "palette": "primary" (over 45% occurrence frequency); "secondary" (10-45%) and "decorative" colours (2-10%). In addition to colour comparison of an item with database images for matching, the following has been used: Fourier descriptors for outline-specific features; Gabor filters for texture feature extraction and SIFT (Scale Invariant Feature Transform) to extract feature vectors. Matching "similarity scores" are then computed, yielding finest matches as the concluded results.

Yuan et al [7] designed a system, embellished with audio feedback and voice-input controls, that detects an item's colour(s) to be one of 8 options, other than white, black and grey (total of 11 options), by employing HSI values. The system furthermore provides an algorithm to decide if a pair of input items shares the same pattern and/or colour where Haar wavelet, and Radon, transforms assisted in pattern comparison.

Yang et al [8] offered a system with visually-impaired user-friendly interface that categorised textures into *stripes; lattice; special* patterns and no patterns. Additionally, the system recognised 11 clothing colours. For texture categorisation, they merged structural and statistical descriptors (SIFT and STA respectively), using a confidence-based measure approach instead of mere concatenation of extracted features to a single vector.

The objective of this paper is to provide the visually impaired user a means for autonomous clothes matching, as well as rich colour identification. A method for detailed outline classification is pursued by offering classification of top/bottom items, further identifying bottom item types into skirts or pair of trousers. In past research it was found that colour identification is narrowed down to only a handful of basic shades, such as in the works of [7, 8]; the motivation in this paper however is to provide an accurate colour extraction method that offers hundreds of colour options for rich feedback. Once tops and bottoms are discerned; the outline alongside extracted colours are used in matching to provide the intended outfit matching system. This paper also aspires to provide users with fashionable suggestions; hence the employment of complementary and analogous colour matching for aesthetic variety of possible outfits, as opposed to returning items with similar colour categories which was noticed in previous projects such as [3, 7] where their colour matching focused on returning matches with similar colours to a given item.

3. PROPOSED OUTFIT COORDINATION APPLICATION

This paper offers a mobile application for outfit coordination, to match an input item image against the user's (pre-populated) closet items in order to generate outfit suggestions.

Figure 1 shows a flow chart to the proposed application. The application consists of 6 phases: 1. Pre-processing and ROI extraction phase; 2. Feature extraction and outline detection phase; 3. Colour Quantisation and Extraction phase; 4. Colour name matching phase. 5. Item Matching phase, 6. Text-to-speech phase to communicate feedback to the application user.

Sections 4-9 discuss the development and testing of these phases in detail. Section 4 presents the development environment. Section 5 discusses the pre-processing and the two ROI extraction methods. Section 6 is dedicated for feature extraction and item's outline detection and classification. Section 7 is dedicated for colour quantisation and extraction. Section 8 is dedicated for extracting the quantised colours' categories and descriptive names. Section 9 discusses the last phase of the system which is the process of item matching, where the item's outline and colours are used to match it with other items in the user's closet database and return the user matching suggestions.

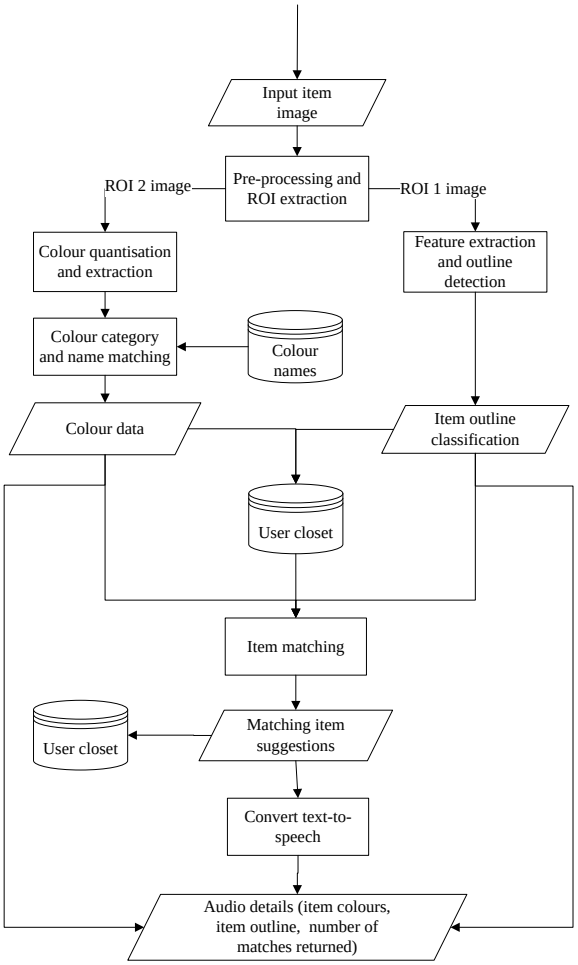

Figure 1: Proposed outfit coordination application

4. DEVELOPMENT ENVIRONMENT

The application was developed on Android mobile phone with specifications provided in Table 1.

OpenCV library [9] was chosen for the purpose of computer vision and machine-learning algorithms used in the application.

Regarding the image dataset used for training and testing purposes throughout the implementation, a collection of apparel items have been selected from a multitude of fashion retail stores' websites such as: Coldwater Creek; Dorothy Perkins; Esprit; Gap; H&M; Miss Selfridge; Net-A-Porter; Ralph Lauren and Topshop. The images utilised have the following properties: 1. Minimal details in the background (mostly plain with a unified colour; at times however there exists a shadow effect in the item's background), 2. A single image in the foreground; the apparel item image of interest, 3. The images are supplied with convenient lighting to support optimal colour reflection; 4. The apparel item is centred and most are not folded.

An HTML colour dataset prepared by Kevin Walsh [10] was adopted as a reference for colour categories and shade names. The HTML-parsing java library, jsoup [11] was used to parse the HTML colour dataset. For audio output, the built-in Android text-to-speech library has been used to provide audio feedback, upon user selection, regarding the evaluated item's outline; colours; and number of item matches found.

Table 1. Testing device specifications

Criterion	Specification
Hardware specifications	
Processor	600MHz
RAM[1]	189-279MB
Internal memory	181MB
External memory[2]	10.5MB
Software specifications	
Operating System	Android 2.2 (API Level 8), known as Android Froyo
Application prerequisites	OpenCV manager (downloaded through Google's Application store, Google Play [12].)

5. PRE-PROCESSING AND ROI EXTRACTION

Images are pre-processed by passing through a median filtering and Gaussian blurring with filter sizes 11 x 11. Such processes are held on each colour channel (Red; Green and Blue) which are split for individual processing and re-merged once pre-processing is done.

Two Region Of Interest (ROI) extraction approaches are applied to the pre-processed images. ROI method 1 precedes the feature extraction phase, while ROI method 2 precedes the colour extraction phase.

***ROI method 1*:** It utilises adaptive Gaussian thresholding with a 3x3 mask over a pre-processed grey-scaled version of the image followed by finding the contours in the binarised image using the algorithm of Suzuki et al. [13, 14, 15]. Upon finding contours, they are then filled [16, 15] to produce the binary mask, whose set pixels are then used as a guide to selecting pixels from the original photo to form an ROI image [15]. Figure 2 illustrates the steps of generating a binary mask using this approach. In Figure 2 two example images are shown to highlight the need for calling the finding and contour filling methods, where in some instances it yields better ROI results. An example on a skirt item is seen in the rightmost flow of Figure 2 showing a fuller ROI than the left example. Even though the finding and filling contours steps were added there were many cases where not the entire ROI is extracted, such as in the T-shirt flow example in Fig. 2 above. However it was found that the ROIs extracted sufficed for the purpose of feature extraction for SVM classificaiton (consider section 6 for further details). For colour extraction on the other hand, the entire item's ROI is pursued to gather as much colour data as possible. Hence, the implementation of ROI method 2.

***ROI method 2*:** It encompasses an item's interior details, suitable for colour extraction. This binarisation method hosts some similarities with that of Hsu et al, summarised as "Grayscale → Canny edge detection → Dilation → Fill holes → Erase small regions → Erode" [3]. A contrast in this paper's method to that of Hsu et al is the use of an xor operation, which was found to be vital to produce a filled ROI with reduced pixel data loss, as soon explained. Another difference is that in this paper small regions have not been erased nor the resultant mask eroded. Although it would have produced a finer result in reducing false positive pixels, the result produced deemed satisfactory when teamed with the colour quantisation and extraction modules selecting the most frequent colours from an image (further discussed in the Colour Quantisation and Extraction section, 7). The algorithm's details are further elaborated in the lines below.

The ROI2 algorithm proceeds as follows:

1. The image is pre-processed using earlier mentioned median and Gaussian filters, and also adding histogram equalisation.

2. The pre-processed image is converted to grey scale by utilising the saturation channel of an HSV version of the image.

3. An edge map of the greyed image is produced using canny edge detection [3].

4. Contours are then found using algorithm of Suzuki et al. [13] from the edge map

5. Douglas-Peucker algorithm [17, 18, 19] is applied over the contours to help approximate them to a more regularised shape through attempting to map the polygons to a straighter figure. The polygons are then filled [20].

6. Dilation is then performed using a 5 x 5 elliptical structuring element to fill edge gaps in pursuit of a connected contour for the apparel item [3].

7. Following the dilation process, the edge map is then converted back to RGB to flood-fill (with a shade other than pure black or white, in this work an orange shade) at its four corners and then its converted to grey scale to be xored with a grey scale version of the dilated binary image. Thus, a (grey scale) version before the flood-fill is xored with a (grey scale) version after the flood fill. The xor operation sets all the ROI pixels to zero, since both xor operand images have the same ROI pixels – just varying background (non-ROI) colours, and the xor operation produces 0 upon xoring two similar values. The xored image is then flood-filled with white at the corners before the image is negated. Upon negation, the image's background is set to black, while the item's pixels are set to white, representing the binary mask.

Figure 3 illustrates the steps of ROI extraction method 2, alongside an example. It is shown that upon the xor step, all similar pixels between the two versions of the dilated binary image are set to zero and then upon inversion, to one, signifying the ROI pixel locations. This binary image is then used as a reference to indicate which pixels in the item image represent ROI pixels, to produce the ROI image.

[1] The testing device has 279MB worth of RAM, 189MB of which are usually free in testing phases (the remaining approximate 90MB tend to be consumed by system operations).
[2] The external memory requires approximately 10MB total, the number is increased for additional assurance.

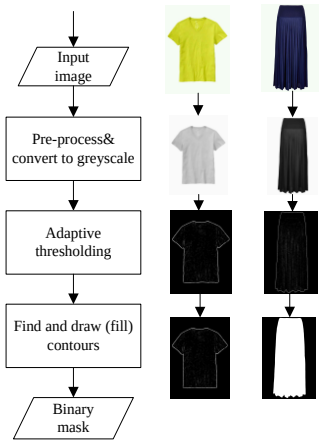

Figure 2: ROI1 binarisation flowchart with examples

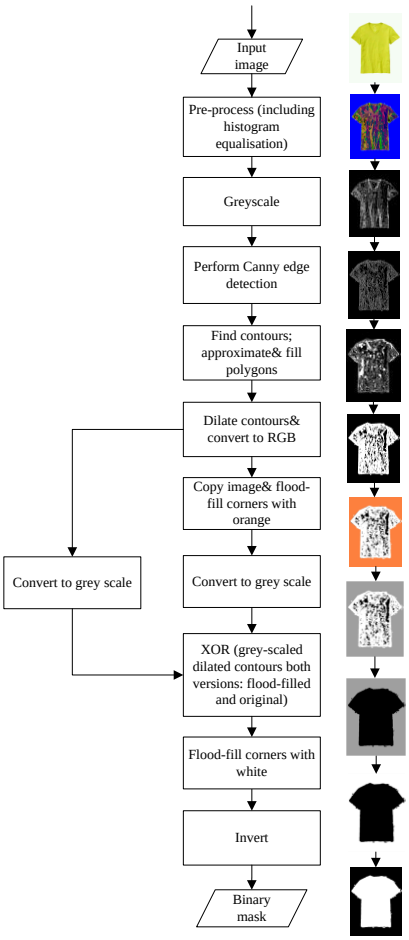

Figure 3: ROI2 binarisation flowchart with example

6. FEATURE EXTRACTION AND OUTLINE DETECTION

This component includes: A feature extraction stage then a two-phase classifier. The feature extraction stage is developed using an extractor called Oriented FAST and Rotated BRIEF (ORB) [21]. One of ORB's prominent features is its remarkable efficiency in comparison to SIFT [22] and SURF (Speeded Up Robust Features) [23] . ORB requires approximately 0.3% and 7%, of the time needed by each extractor respectively, per image. Furthermore, ORB has the advantage of being a royalty-free alternative to SIFT and SURF [21].

The input to the ORB is the ROI1-image. The output of the ORB is a set of features that are fed to a linear, binary, Support Vector Machine (SVM1). SVM1 is the first phase of what is referred to as the two-phase classifier. SVM1 classifies an item into either a top or a bottom. If the item is classified as a bottom, it is fed into a second linear, binary SVM (SVM2) for further classification, into either a skirt or a pair of trousers.

For training images, a total of 182 training images' features has been used (80 tops – including knee-length dress and coats; 55 pairs of trousers and 47 skirts). SVM1 training dataset considered all items while SVM2's training set considered only trouser and skirt items.

The two-phase classifier has been tested using the features extracted from each of ROI1 and ROI2, which provides the SVM features input. A test set of 10 item images has been assembled to include 4 tops; 3 pairs of trousers and 3 skirts. For SVM phase 1 all 10 images were tested; for SVM 2 only the 6 bottom items were tested. The test images for the two-phase classifier are shown in Figure 4, with the top test items in the first row; trousers in the second and skirts in the third.

Figure 4: Two-phase classification test items

After running different ROI variations on the test set to precede feature extraction (all using ORB), it was found that the method yielding highest accuracy rates required ROI1. Furthermore, the ROI1 images were resized to the mode image size of the training dataset which enhanced accuracy results (compared to no scaling). Test images were also resized to the mode size. Training set image mode size was thus saved as preference on the test device.

To test the different ROI variations, after feature vectors were produced (from the scaled/non-scaled ROI1/ROI2 images, using ORB), vectors were padded with zeros to reach the maximum size of all training feature vectors. In cases where a test image's features exceeded such maximum size, the feature matrix was resized to the saved maximum size (also saved via Android preferences).

The remainder of this section illustrates the experimental data supporting the decision of using scaled ROI1 images as the preferred source to extract features from. Table 2 tabulates the ROI variation test cases. 'Scaled' in Table 2 refers to scaling the image to the mode size of training images.

Performance metrics consulted were the accuracy, sensitivity and specificity for each of the two SVM classifiers.

Table 3 and Table 4 provide the classes with respect to which specificity and sensitivity were measured in first and second phase SVMs respectively. The formulae used for performance metrics are indicated in Table 5. They are applied for each of SVM 1 and SVM 2 using their designated positive and negative classes identified in
Table 3 and Table 4 respectively. Note that *true positives* refer to items correctly classified as positives; *total positives* refer to all items originally categorised as positives. The same naming convention is applied to *true negatives* and *total negatives* with respect to the *negative* category.

Table 2. Test cases for SVM effectiveness measurement

Test case	Preparation for image preceding feature detection (for training and test images)
1	Not scaled, ROI method 1
2	Not scaled, ROI method 2
3	Scaled, ROI method 1
4	Scaled, ROI method 2

Table 3. SVM phase 1 class key for effectiveness analysis

Value	Classification
1 (positive class)	Tops
-1 (negative class)	Bottoms

Table 4. SVM phase 2 class key for effectiveness analysis

Value	Classification
1 (positive class)	Skirts
-1 (negative class)	Pairs of trousers

Table 5. Performance metrics

Metric	Formula (Percentage)
Accuracy	$\frac{100 * \ correctly\ classified\ items}{total\ test\ items}$
Sensitivity	$\frac{100 * \ true\ positives}{total\ positives}$
Specificity	$\frac{100 * \ true\ negatives}{total\ negatives}$

Figure 5 and Figure 6 illustrate the effectiveness results for SVM1 phase and SVM2 classifiers respectively. As shown in Figure 5 and Figure 6; case 3, which used scaled ROI1 images as the feature extraction source, signified highest accuracy in both SVM phases. Test case 3 hosted 70% accuracy in SVM 1 discerning tops from bottoms with most erroneous classifications being false negatives (misclassifying tops as bottoms), hence the 50% sensitivity (and approximate specificity of 83.3%). More significantly, test case 3 yielded a 100% score accuracy, sensitivity and specificity with bottom items (SVM 2) in the test cases.

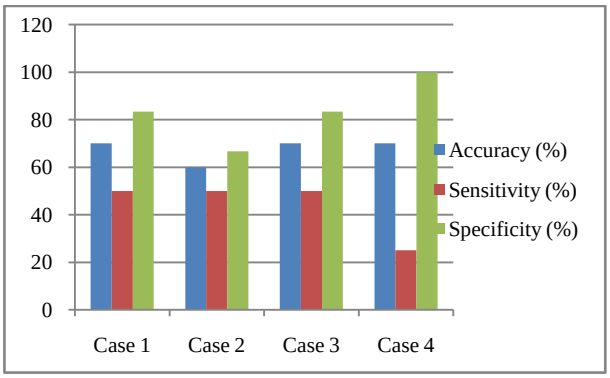

Figure 5: SVM phase 1 effectiveness graph

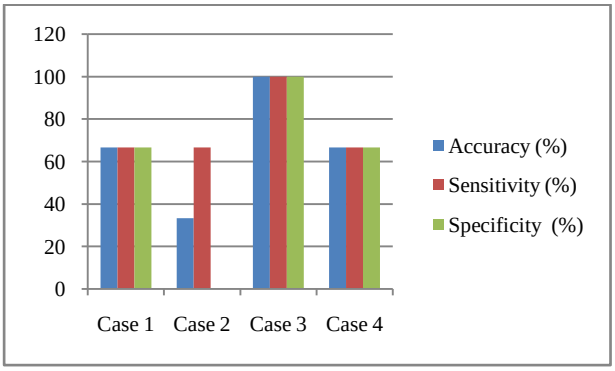

Figure 6: SVM phase 2 effectiveness graph

7. COLOUR QUANTISATION AND EXTRACTION

The objective of this component is to extract the major colours of ROI2 images. Normally, an image's histogram encompasses a broad array of bins, in each of the Red, Green and Blue channels, and thus colours utilised in the image, the image's colour palette, are required to be narrowed down, through a colour quantisation algorithm, for more focused recognition of the various colours within the image. This paper utilises *k*-means clustering of an image's pixels into colour clusters [24], to perform the needed colour palette reduction.

The colour quantisation and extraction process proceeds as follows:

1. The original image is downsized to its half , before it's fed to ROI method2, to reduce the pixels considered for colour quantisation.

2. The pixels of the ROI2 image are then iterated over to determine which are to be sent for quantisation, since the background pixels of the ROI2 do not represent the item's colours.

3. The image is then vectorised to a single column and then quantised using *k*-means.

4. Once the image's colours are quantised, the colours are sorted by frequency. The quantised colours are then mapped to the nearest colour *id*s (as discussed in Section 8.2) and saved to the database and can be used in item matching and colour name extraction.

The clustering process is repeated three times in efforts to yield better results [25]. The initial centres are selected randomly.

Different values to the number of quantisation bins, or equivalently the *k* value for the *k*-means clustering, were experimented, as *k* is one of the parameters that impact the time complexity of the k-means algorithms. Figure 7, illustrates graphically the time taken to colour-quantise a single image given an assortment of bin values. As shown by the figure, to quantise an image; approximately 29 minutes are required for 15 bins, compared to approximately 0.3 minutes for the three bins. So *k* is set to 3 to speed up the mobile application. Moreover, three colours tend to offer a satisfactory metric to describe a multicoloured clothing item's dominant resident shades.

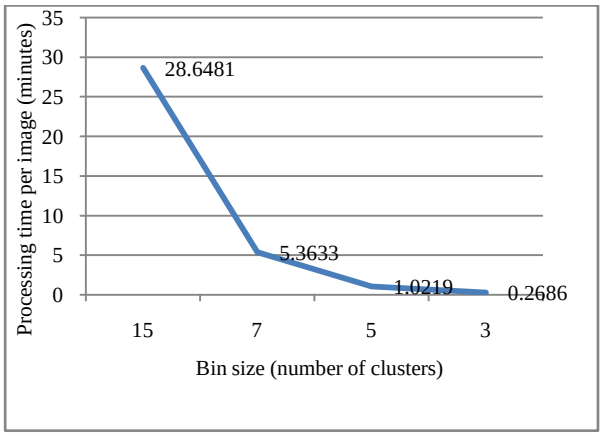

Figure 7: Quantisation processing times of an image for various bin sizes

8. COLOUR CATEGORY AND NAME MATCHING

To offer the application user a detailed verbal feedback about the input apparel's colour, a database of colours and descriptive names was built and a colour name matching process was performed with the aid of this database. The following two subsections discuss the database of colour categories and the colour name matching.

8.1 Database of Colour Categories and Names

To build a database of colours, a dataset of colour values mapped to respective descriptive names is required. For such purpose, the HTML-based colour directory, compiled by Kevin Walsh [10] was chosen. The HTML tables were parsed using jsoup [11] and saved in a SQLite database in the Android application.

The colour-related tables in the SQLite database are represented by the Entity Relationship Diagram (ERD) seen in Figure 8.

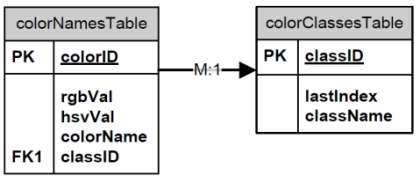

Figure 8: ERD from perspective of colour names

As shown in the ERD, the *colorNamesTable* offers colour details: RGB and HSV values, name, as well as the foreign key of colour category, *classID*. The second table, *colorClassesTable*, represents the colour classes (categories) hosts the *lastIndex* column which designates the last *colorID* (from *colorNamesTable*) within the current colour class. For completeness, the table preserves the colour class's name (such as *Blue*, *Brown* and *Green*) in the *className* column. There are 10 colour classes, representing 581 colours in total. This vast amount of colours in the database is expected to assist with plausible colour name outputs even in cases where the actual colour category classification is not optimal, for example if a colour is misclassified as white instead of grey, there are still some colour shades in the white category that represent greyish hues.

Colour classes are helpful in having nearest neighbour matching performed against it (first step in finding a matching colour name, as will be soon elaborated upon, section 8.2). Colour classes are also helpful in matching apparel based on complementary and homogenous colours. Colour classes were parsed from table headers [10] where each colour category had an exclusive table (example colour classes are *red*, *blue* and *yellow*, where each hosts a myriad of various "shades" within its own table).

The colour records are read sequentially, one at a time. The parsed colour names were run through string operations to yield readable results; numbers, if existed, were purged from colour names and capitalised letters were preceded by spaces to separate words (such as *SlateGray1* and *SlateGray2* both became *Slate Gray*) for easier user comprehension upon the text-to-speech feature. Also, colour records with a colour value but no name were not added to the colour database.

8.2 Colour Name Matching

In this stage the quantised colours, section 7, will be matched against the colour names database. The first step to find the colour name was finding the colour class; k-nearest neighbours (KNN) algorithm was used for this purpose. To train the KNN object, the parsed colour data (stored in the SQLite database) was saved in a matrix in HSV format and then fed into the KNN object as a training set. Each of the three colours extracted from image colour quantisation was converted to HSV and saved in a matrix. Then the nearest colour category was found using KNN with k set to 7. Only the hue channel has been used in both training dataset and quantised colours.

The colour category classification comparisons are based on hues since similar RGB values can be found across various colour categories, thus hue values help discern one category from another. It was observed that comparisons based on hue (H) only tends to yield higher accuracy in category detection than using saturation and/or value (S,V) as well, as depicted in Figure 9 (with other RGB varieties compared for experimentation). This result was expected since hue is commonly used for matching, given its intuitive measure for a colour's category [26]. Test results were based on a sample set of 10 colours, half of which did not exist in the colour database, and thus their categories were set by visual assessment. The non-existing colours tested the nearest neighbour mapping, especially regarding colour names' plausibility in cases of no exact matches in the database.

Hue-based colour category classification also has the added advantage of a higher efficiency in terms of nanoseconds, as illustrated in Figure 10, where classification using Hue only saves around 44% (0.26 seconds) of training dataset preparation time when compared to classification using all HSV channels. This is considerable because KNN is a lazy learner, and thus retrains the training dataset (which can grow with larger colour databases) per classification; unlike with SVMs for instance

where the trained model was saved as an external xml file (which is facilitated by the OpenCV library).

The legend '*prepare training dataset*' in figure 10 refers to the addition of the category labels of all colour records - retrieved from the colour database – into a matrix object, as well as the colour values themselves (for the channel(s) in question) in a matrix object representing training attributes. This is variable to colour channels used where less channels are expected require less time to save. Provided HSV colour space was used for classification, the colour would also need to be converted to HSV. The '*instantiate KNN object*' step, in this project using the OpenCV library, is also the step that trains the dataset. The '*prepare test dataset*' step adds the training dataset's colours themselves (for the channel(s) in question) to a matrix object converting to HSV upon need.

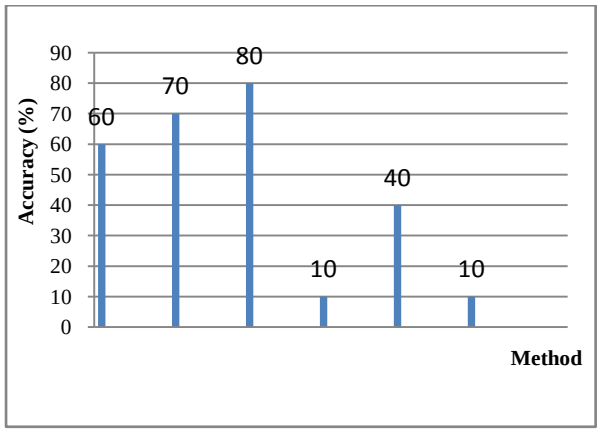

Figure 9: Channel-based comparison of KNN category colour matching accuracy

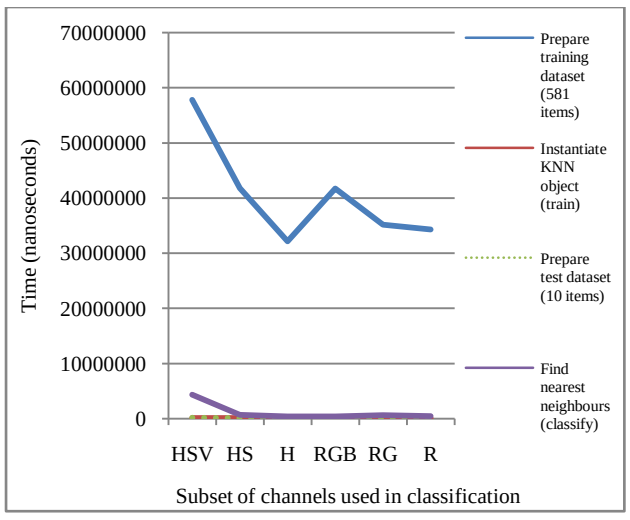

Figure 10: Channel-based comparison of processing time for KNN classification

Once the colour category is found, a linear search then follows to discover the nearest colour shade - also referred to here as nearest neighbour - in the deduced colour class from the colour database (instead of searching the entire colour database). To estimate the nearest neighbour - which help matches the input colour to a descriptive name from the colour dataset - Manhattan distance was used to compare RGB distances (for all three channels). The nearest RGB value was chosen as the metric to find the nearest colour name to yield the nearest shade sought after.

For an example of colour name results, the image below in Figure 11, has *Dark O*range, *Light Salmon* and *honeydew* describing the top three dominant colours. Note how the *honeydew* colour could be attributed to the top's interior's brand tag as well as the light cream embellishments on the top's front, offering visually impaired users richly illustrated apparel colour data. Such colour detail has a wealth of potential in future applications detecting various materials' colours.

Figure 11: Sample top with colour name results Dark Orange, Light Salmon and Honeydew

9. ITEM MATCHING

To provide the user with outfit coordination results, the system is to return compatible matches, to the input item image, in terms of outline and colour.

In this paper the item's most frequent colour is used to find complementary and analogous matches. Medium frequent and least frequent colours could also be considered, however for a reduced pool of more related results, only the most dominant colour has been chosen. Thus, provided a patterned item is in question, only the most dominant colour in that apparel's material is considered in the outfit matching. The query results are also purged by the item outline by querying items in the user's closet with the opposite outline. Thus for trousers and skirts, tops results are returned, and vice versa.

The matching process ensues as follows: (1) matching colours - two analogous and one complementary, elaborated upon later - are computed using the most dominant colour in the apparel; (2) every computed matching colour is classified into a colour category using the colour database – this calls for KNN classification (method discussed in Colour Name Matching section 8.2). Then (3) - still for every matching colour - the system searches for matches through querying the closet for items with the needed matching colour category and outline. Regarding colour, the query first starts by searching for the matching colour category using the most frequent colour; provided the user's closet hosts no items with such colour as the most frequent one, then the search is repeated for the second most frequent colour, then if still no results, the least frequent colour, before determining that no matching results are found. The topmost result could then be highlighted, or even sort the matching results in order of closest matches by sorting the matching items ordered by ascending (Manhattan) distance from the original matching colours, depending on application needs. The matching details are then saved in the SQLite database. Figure 12 offers an example matching result for a pair of trousers given a sample user closet of 10 items in the application prototype.

Regarding computing colour matches; a shade's complementary colour hosts a hue value (hence the need for HSV space) 180° away from the original one - on the opposite end of the hue spectrum. After shifting the hue by adding 180°, the new colour needs to be normalised to the maximum allowable hue: 360°.

While complementary colours tend to offer more colourful outfit options for users, analogous colours are inclined to be a colour-conservative choice where matching colours are more similar to one another, offering additional fashion options for the user. For every colour, two analogous colours are extracted: the neighbouring colours on the right and left banks on the colour wheel – each 30° apart in hue value. To obtain the analogous colours, simply the hue is shifted leftwards (by subtraction) and rightwards (by addition) to result in two different analogous shades (and normalise to the maximum 360°). Colour wheels have a continuous range of possible shades, 256^3 possible shades to be numerically precise [27]; however, it is common to divide the colour wheel into 12 slices, as shown in the colour wheel in Figure 13. Thus the distance from a colour of $\frac{360°}{12} = 30°$ was adopted as a sound heuristic to obtain an analogous shade. Figure 13 illustrates the concept of a colour wheel, divided into 12 sections; it also demonstrates an example for a complementary colour (green) and analogous colours (pink and orange) with respect to the colour red.

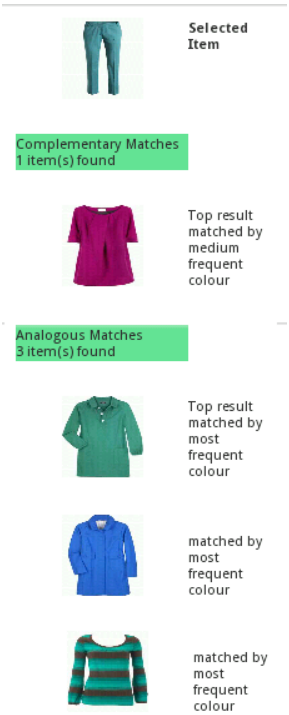

Figure 12: Sample matching result

Figure 13: Example of complementary and analogous matches with red

10. CONCLUSION AND FUTURE WORK

This paper discussed the development of a mobile application for outfit coordination. The application could be used in both shopping and home environments to help visually impaired individual to decide what items to purchase independently; as well as to assist him to independently decide what to wear on a daily basis.

For this purpose two ROI extraction methods have been implemented each of which caters an exclusive purpose: The first ROI method, using adaptive thresholding, was used for feature extraction; while the second ROI method, using canny edge detection and an xor operation, offered binary masks fit for colour extraction. A two-phase SVM has been implemented to classify items to top or bottoms then classify bottoms to either skirts or pants. A rich colour description feedback has been developed to provide the application user with highly descriptive colour feedback, selected from an array of 581 colours. Furthermore, an intuitive approach to outfit matching is developed to offer users with a variety of fashion options. Complementary and analogous matching are used as opposed to only matching similar colours together, as seen in previous applications. Computational and storage efficiency were concerned in developing the application to ensure its being responsive.

Further work to consider is extending the two-phase SVM to multi-phase SVM for further classification to items such as classifying the tops to shirts and jackets.

11. REFERENCES

[1] World Health Organization, "Visual impairment and blindness," October 2013. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs282/en/. [Accessed 29 May 2014].

[2] M. A. Burton, E. Brady, R. Brewer, C. Neylan, J. P. Bigham and A. Hurst, "Crowdsourcing Subjective Fashion Advice Using VizWiz: Challenges and Opportunities," *ASSETS*, pp. 135-142, 22-24 October 2012.

[3] E. Hsu, C. Paz and S. Shen, "Clothing Image Retrieval for Smarter Shopping," California, 2011.

[4] T. Iwata, S. Watanabe and H. Sawada, "Fashion Coordinates Recommender System using Photographs from Fashion Magazines," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan, 2011.

[5] J. Rose, "Closet Buddy: Dressing the Visually Impaired," *ACM SE*, pp. 611-615, 10-12 March 2006.

[6] N. Paisios, L. Subramanian and A. Rubinsteyn, "Choosing which Clothes to Wear Confidently: A Tool for Pattern Matching," New York, 2012.

[7] S. Yuan, Y. Tian and A. Arditi, "Clothing Matching for Visually Impaired Persons," IOS press, New York, 2011.

[8] OpenCV, [Online]. Available: http://opencv.org/.

[9] K. J. Walsh, "rgb," Copyright © 2010 Kevin J. Walsh, 2010. [Online]. Available: http://web.njit.edu/~kevin/rgb.txt.html. [Accessed 20 November 2012].

[10] jsoup, "jsoup: Java HTML Parser," [Online]. Available: http://jsoup.org/.

[11] OpenCV manager, "OpenCV Manager," [Online]. Available:

https://play.google.com/store/apps/details?id=org.opencv.engine&hl=en.

[12] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," *CVGIP*, vol. 30, no. 1, pp. 32-46, 1985.

[13] Find contours, [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#findcontours.

[14] P. Wagner, "Extracting Contours with OpenCV," 26 May 2012. [Online]. Available: http://www.bytefish.de/blog/extracting_contours_with_opencv/. [Accessed 30 May 2014].

[15] Draw contours, [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#drawcontours.

[16] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer* , vol. 10, no. 2, p. 112–122 , 1973.

[17] Approximate polygonal curves, 2014. [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#approxpolydp. [Accessed 31 May 2014].

[18] Abid Rahman, "Contours-2: brotherhood," 2012. [Online]. Available: http://opencvpython.blogspot.com/2012/06/contours-2-brotherhood.html. [Accessed 3 June 2013].

[19] D. G. Lowe, "Object recognition from local scale-invariant features," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, Kerkyra, 1999.

[20] B. Herbert, T. Tuytelaars and L. V. Gool, ""Surf: Speeded up robust features," *Computer Vision–ECCV* , pp. 404-417, 2006. Athi, "Color Quantization," [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/31687-color-quantization.

[21] Athi, "Color Quantization," [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/31687-color-quantization.

[22] K-menas, [Online]. Available: http://docs.opencv.org/modules/core/doc/clustering.html#kmeans.

[23] D. Cardani, "Adventures in HSV Space". M. Specht, "Using perceptually uniform color spaces for imagesteganography steganography: An enhancement of the Least Significant Bit method".

[24] Dataset images from coldwatercreek.com, dorothyperkins.com, esprit.com, gap.com, hm.com, missselfridge.com, net-a-porter.com, ralphlauren.com and topshop.com.