Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 11 (2016) pp. 021-030

Facial Sketch-to-Photo Matching and Face Recognition using Local Invariant Features

Alaa Tharwat[1,2], Hani Mahdi[2], Adel El Hennawy[2]

[1]Faculty of Engineering, Suez Canal University, Ismailia, Egypt
[2]Faculty of Engineering, Ain Shams University, Cairo, Egypt

Abstract: **In this paper, a proposed model is used to identify face sketch images based on local invariant features. Our system has three phases: (1) Feature extraction from the photos and sketches; (2) Reducing the dimension of feature vectors; (3) Matching the features of the unknown sketches with the features of the original photos. In this model, four local invariant feature extraction methods are used to extract local features from photos and sketches namely; Scale Invariant Feature Transform (SIFT), Speed Up Robust Features (SURF), Local Binary Patterns (LBP), and Weber Local Descriptor (WLD). Due to high dimensional features of all feature extraction methods, Direct Linear Discriminant Analysis (Direct-LDA) is used. Moreover, in the classification phase, AdaBoost classifier is used. CHUK face sketch database images were used in our experiments. The experimental results proved that local invariant features achieved high accuracy and SIFT method achieved the best accuracy. Moreover, the effect of the different parameters were discussed and tuned to extract robust and discriminative features.**

Keywords: Face Sketch, Scale Invariant Feature Transform (SIFT), Local Binary Patterns (LBP), Speed Up Robust Features (SURF), Linear Discriminant Analysis (LDA), Direct-LDA, Weber Local Descriptor (WLD), AdaBoost Classifier.

I. Introduction

Biometrics method is one of the methods that are used identify individuals by measuring and analyzing physiological characteristics, e.g. face, fingerprint, and iris, and behavioral characteristics, e.g. signature, gait, and keystrokes. Face recognition is one of the widely used biometrics. However, changing expressions, mode, and health have a great impact on the accuracy of face recognition. Moreover, face images may be occluded by glasses or clothes [**?**, **?**, **?**, **?**, **?**].

Face recognition is used to assist law enforcement by retrieving the photo of the suspects using sketch images which are drawn based on the recollection of an eyewitness. Hence, matching a sketch drawing with the photos becomes important and helps the police to locate a group of potential suspects. In the past, manual face sketch recognition used manual measurements, but it achieves low accuracy and needs more time. On the other hand, automatic face sketch recognition systems save processing time and increase the accuracy. In addition to the challenges of face recognition application, face sketch depends mainly on the skills of the artists. However, there is a great difference between sketches and photos; hence, face sketch recognition is much harder than normal face recognition [**?**, **?**, **?**].

The process of face sketch recognition system starts from manually drawing a sketch by artists and then matches the sketch with different photos in the database to determine the nearest photo to the sketch and hence identify the person. Different factors affect the accuracy of face sketch recognition such as the skills of the artists and the matching algorithm [**?**, **?**]. Another important factor is that the sketch is drawn while viewing a photograph of the person or the person himself or the sketch is drawn by asking a witness about the descriptions of the suspect [**?**, **?**, **?**].

A. Related Work

Many studies used face sketch to recognize face photo by matching sketches directly with photos. For example, Yong *et al.* [**?**], used hand-drawn face sketches which were collected from five different artists. They used *Principal Component Analysis* (PCA) to extract features and Mahalanobis distance was used as a classifier. Moreover, they used score level fusion technique and achieved 90% recognition rate. Alaa Tharwat *et al.* used *Scale Invariant Feature Transform* SIFT and *Local Binary Patterns* LBP to extract local features from sketches and photos. They used *Support Vector Machines* (SVM) and minimum distance classifiers to match the features of the unknown sketches with the photos and they achieved 99.25% and 96.4% using SIFT and LBP methods, respectively [**?**]. Due to the difference between sketches and photos, many other studies synthesized sketches from photos to solve this problem. Zhong *et al.* synthesized a photo from a sketch and then they used photo-photo recognition to achieve better accuracy [**?**]. They used 56 color photo-sketch pairs and *Embedded Hidden Markov Model* (EHMM) to map between photos and sketches and used eigenspace to identify sketch images. Their proposed model achieved recognition rate 17.6% when they directly applied photo-sketch recognition while the recognition rate increased to 88.2% when they transform photos-to-sketch and sketch-to-photo transformation. Liu *et al.* generate pseudo-sketch images using *Local Linear Preserving* (LLP). Moreover, they used *Kernel Linear Discriminant Analysis* (KNDA) to recognize face photos

from the pseudo-sketch images. They used CHUK dataset and polynomial kernel for KNDA and their proposed model achieved accuracy ranged from 87.67% to 99% [**?**]. Xiaoou *et al.* used PCA technique to transform the photo face images into pseudo-sketches. In addition, they used PCA and Bayesian classifier to recognize face photos from pseudo-sketches and they used CHUK dataset and their proposed accuracy ranged from 81.3% to 97% [**?**]. Xinbo *et al.* used Embedded Hidden Markov Model (E-HMM) to model the nonlinear relationship between a photo-sketch patch pair to generate pseudo-sketch images. Moreover, they used PCA to extract features and they used CHUK dataset and their recognition rate was 95.24% [**?**]. Alaa Tharwat *et al.* used Gabor filters to extract features from three different scales. The results of the three scales were combined at classification level fusion. They also used linear regression to transforms photos to sketches, i.e. pseudo-sketches. They found that matching sketches to pseudo-sketches achieved better results than matching sketches with original photos [**?**].

B. Our Model

In this paper, a face sketch model based local features was proposed. There are two methods to extract the local invariant features, namely, sparse descriptor and dense descriptor. In sparse descriptor method, the interest points were extracted to represent the features such as *Scale Invariant Feature Transformation* (SIFT) [**?**] and *Speeded Up Robust Features* (SURF) [**?**]. While in the dense descriptor methods, local features were extracted from every pixel over the input image such as *Local Binary Patterns* (LBP) [**?**, **?**] and *Weber's Local Descriptor* (WLD) [**?**]. In this model, two sparse methods (SIFT and SURF) and two dense methods (LBP and WLD) were used to extract local features. Each method has different parameters that control the accuracy of the model and the CPU time. Due to high dimensional features of the feature extraction methods that were used in our proposed model, *Direct Linear Discriminant Analysis* (Direct-LDA) method was used for two purposes. First, Direct-LDA reduces the dimension of feature vectors. Second, it increases the separability between different classes. Adaptive Boosting (AdaBoost) classifier was used to match the features of an unknown sketch with the photo images.

The rest of this paper is organized as follows: The preliminaries of the proposed model are introduced in Section II. The proposed model is explained in Section III. Experimental results and discussion are introduced in Section IV. Finally, concluding remarks and future work are introduced in Section V.

II. Preliminaries

A. Scale Invariant Features Transform (SIFT)

SIFT is widely used due to its robustness against many challenges such as scaling and rotation [**?**]. SIFT algorithm consists of four steps as follows:

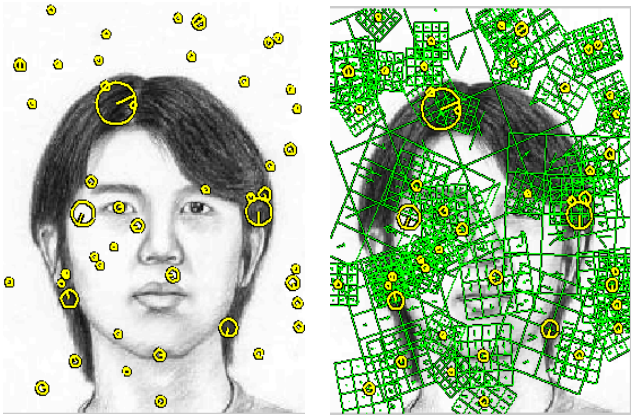
1. *Creating the Difference of Gaussian (DoG):* The goal of creating DoG or scale-space is to decompose the input image into different scales to extract features from different image's scales. At the first level, the images will be in the original size. At the second level, the images are half-sized, quarter-sized at the next level, and so on [**?**, **?**, **?**, **?**]. The images are decomposed by filtering the original images with Gaussian functions of many different scales. The DoG represents the difference between nearby scales separated by a constant multiplicative factor k as follows:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (1)$$

where, $D(x, y, \sigma)$ is the DoG, $L(x, y, \sigma)$ and $L(x, y, k\sigma)$, are two images that are produced from the convolution of Gaussian functions with an input image $I(x, y)$ using σ and $k\sigma$, respectively, as follows:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2)$$

where $G(x, y, \sigma)$ represents the Gaussian function and it is calculated as follows, $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} exp[-\frac{x^2+y^2}{\sigma^2}]$ [**?**].



(a) (b)

Figure. 1: A sample of keypoints and its orientation, (a) the keypoints without orientation, (b) the keypoints after applying assignment orientation step.

2. *Key-point (Extrema) Detection:* The Interest points in DoG pyramids, called key-points, are detected by comparing each point with its 26 neighbours, i.e. eight neighbours in the current level and nine neighbours in the above and below levels, to compute the extrema points. If the value of the current pixel represents local minimum or maximum, then this point is known as an extrema [**?**, **?**]. Some key-points that are sensitive to noise, or are poorly localized on an edge are removed.

3. *Orientation Assignment:* In this step, one or more orientations are assigned to the key-points based on local image properties as shown in Figure (1). An orientation histogram is first formed from the gradient magnitude and orientations of the sample points within a region around the key-point. Gradient orientation and orientation are calculated as in Equations (3 and 4) [**?**].

$$m(x,y) = ((L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2)^{\frac{1}{2}} \quad (3)$$

$$\theta(x,y) = \arctan \frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-1,y))} \quad (4)$$

Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak in the histogram and any other local peaks are used to create a key-point with that orientation. Some points will be assigned to many different orientations if there are multiple peaks of similar magnitude [**?**, **?**].

4. *Descriptor Computation:* The image gradient magnitudes and orientations are sampled around the key-point location and illustrated with small arrows at each sample location [**?**]. The coordinates of the descriptor and the gradient orientations are rotated relative to the key-point orientation to increase the robustness against orientation invariance [**?**].

SIFT algorithm has many parameters that have a great impact on the robustness of SIFT. The first parameter is the *Peak Threshold* parameter. This parameter is used to determine the amount of contrast to accept a key-point. The minimum value of Peak Threshold parameter is 0, which all key-points are accepted, i.e. there are no eliminated key-points as shown in Figure (2). Increasing the value of peak threshold parameter decreases the number of key-points; hence, the robustness of feature matching may be decreased. The second parameter is the *Patch Size* parameter, which controls the number of patches and the number of key-points. Hence, increasing the size of patches decreases the number of key-points and SIFT will be considered as global features. On the other hand, decreasing patch size parameter increases the number of key-points; hence, increases the CPU time [**?**, **?**]. The *Number of Angles* is the third parameter. Increasing number of angles collects features in different angles, i.e. orientations, and increase the number of features; hence, improve the accuracy. On the other hand, decreasing number of angles leads to a small number of features, low matching rate, and the features are more sensitive to rotation [**?**].

B. Speed Up Robust Features (SURF)

SURF algorithm is one of the recent feature extraction methods. It consists of the following steps.

1. In the first step, the integral of an image is generated. An Integral image $I(x)$ is an image where each point $X = (x, y)$ stores the sum of all pixels in a rectangular area between origin and X (See Equation (5)) [**?**].

$$I(X) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x,y) \quad (5)$$

The calculations of the integral image are independent of the size of the image.

2. In the second step, the interest points are extracted. In SURF algorithm, Hessian matrix is used to extract the interest points as follows:

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \quad (6)$$

where $X = (x, y)$ is a point in an image I, σ is the scale, and $L_{xx}(X, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2 g(\sigma))}{\partial x^2}$ with the image I in point X as follows, $L_{xx}(X,\sigma) = I(X) * \frac{\partial^2 g(\sigma))}{\partial x^2}$. Similarly, L_{xy} and L_{yy} are calculated as follows, $L_{xy}(X,\sigma) = I(X) * \frac{\partial^2 g(\sigma))}{\partial xy}$ and $L_{yy}(X,\sigma) = I(X) * \frac{\partial^2 g(\sigma))}{\partial yy}$ [**?**, **?**]. Hessian matrices calculates the interest points as follows, $Det(H_{approx}) = D_{xx}Dxy - (wD_{xy})^2$, where D_{xx}, D_{xy}, D_{yy} are the approximation of second order Gaussian partial derivative in x, xy, and y directions, respectively, and w is the weight of the filter responses. A negative determinant means that eigenvalues with different signs, therefore, the analyzed point neither a local maximum nor minimum; hence, it is not interest point. On the other hand, positive determinant indicates that both eigenvalues are either positive or negative, thus an extreme point (maximum or minimum) is detected. For more details, we refer to [**?**, **?**].

3. The third step is the orientation assignment. In this step, Haar wavelets responses are calculated in x and y directions within a circular neighbourhood of radius $6s$ around the interest point, where s represents the scale at which the interest point is detected. The orientation of each interest point is then calculated by all responses within a sliding orientation window of size $(\pi/3)$ [**?**, **?**].

4. In the fourth and last step in SURF algorithm, a SURF descriptor for each interest point is generated. In this step, the descriptor is calculated from square interest point's neighbourhood with size $20s$. The square is then divided into 16 equal sub-squares with edge size $5s$. Inside each square, Haar Wavelet filters are used to calculate responses. Moreover, the neighbourhood of the interest point is rotated to match the orientation of the interest point [**?**, **?**].

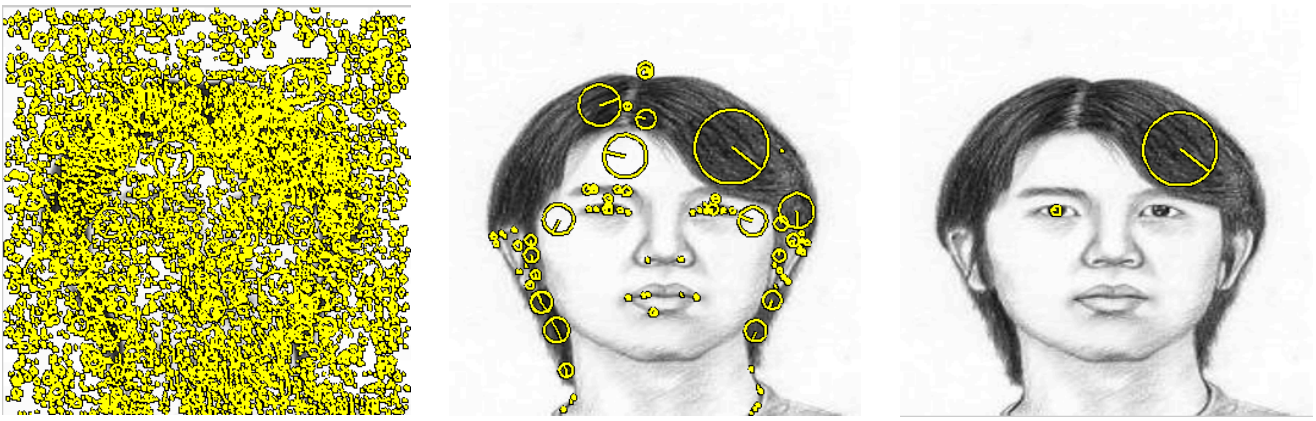
Detector or contrast threshold is one of the most important parameters of SURF. The goal of this parameter is to eliminate weak interest points as peak threshold parameter in SIFT.

C. Weber Local Descriptor (WLD)

The goal of WLD method is to describe an image as a histogram of gradient orientations and differential excitations [**?**]. WLD is originally inspired by Weber's Law where Ernst Weber, in the 19^{th} century, observed that the ratio between an increment threshold and the background intensity is constant as follows:

$$\frac{\Delta I}{I} = k \quad (7)$$

where ΔI represents the increment threshold, I refers to the initial intensity or an image background, and k denotes the



(a) Peak Threshold =0 (b) Peak Threshold=0.10 (c) Peak Threshold=0.4

Figure. 2: The effect of the value of peak threshold parameter on the number of features.

constant value even if I is changing. The fraction $\frac{\Delta I}{I}$ is known as *Weber law or Weber fraction* [**?**].
WLD algorithm consists of three steps, namely, calculating differential excitations, gradient orientations, and building the histogram. The details of the WLD steps are summarized below.

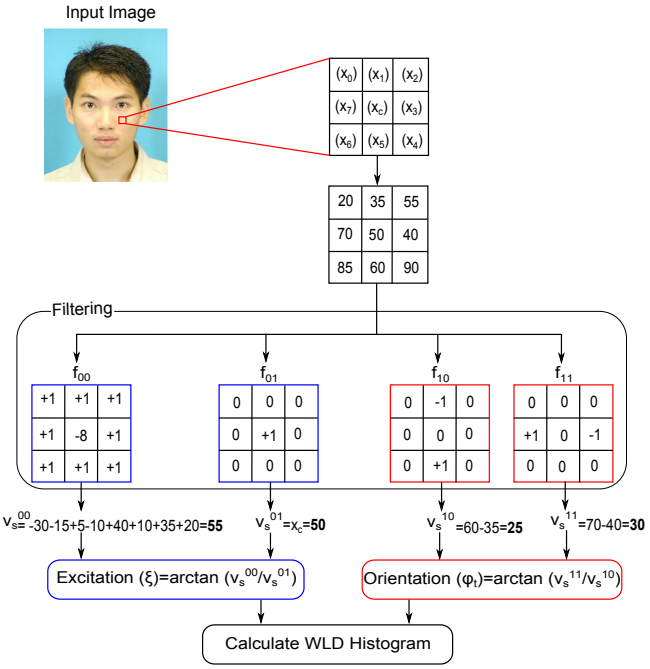


Figure. 3: Illustration of the computation of the WLD algorithm.

1. *Differential Excitation (ξ):* In this step, the difference between the pixel x_c (the center pixel) and its neighbours is calculated using Equation (8) [**?**].

$$\nu_s^{00} = \sum_{i=0}^{p-1}(\Delta x_i) = \sum_{i=0}^{p-1}(x_i - x_c) \tag{8}$$

where $x_i (i = 0, 1, \ldots, p-1)$ represents the intensity of the i^{th} neighbours of x_c and p refers to the number of neighbours. For example, as shown in Figure (3), there are eight neighbours to x_c, i.e. $p = 8$. To calculate the differential excitation and the orientation, four filters, f_{00}, f_{01}, f_{10}, and f_{11} are used to calculate $\nu_s^{00}, \nu_s^{01}, \nu_s^{10}$, and ν_s^{11}, respectively, where, ν_s^{00} represents the difference between x_c and its neighbours as denoted in Equation (8), $\nu_s^{01} = x_c$, $\nu_s^{10} = x_5 - x_1$, and $\nu_s^{11} = x_7 - x_3$. The ratio between the differences, ν_s^{00}, and the intensity of the current pixel, $\nu_s^{01} = x_c$ is then computed as in Equation (9). Finally, the arc-tangent function is applied on $G_{ratio}(.)$ to get the differential excitation of (x_c), as denoted in Equation (10) [**?**].

$$G_{ratio}(x_c) = \nu_s^{00}/\nu_s^{01} \tag{9}$$

$$\begin{aligned}\xi(x_c) &= G_{arctan}[G_{ratio}(x_c)] = \arctan\left[\nu_s^{00}/\nu_s^{01}\right] \\ &= \arctan\left[\sum_{i=0}^{p-1}\left(\frac{x_i - x_c}{x_c}\right)\right]\end{aligned} \tag{10}$$

2. *Orientation (ϕ_t):* This step is starting by computing the gradient orientation of the current pixel, x_c, by calculating the changes in the horizontal and vertical directions as follows:

$$\theta(x_c) = \arctan\left(\frac{\nu_s^{11}}{\nu_s^{10}}\right) = \arctan\left(\frac{x_7 - x_3}{x_5 - x_1}\right) \tag{11}$$

The gradient orientation is quantizing by transforming it into T dominant orientation. This is achieved by first mapping θ to $\acute{\theta}$ as follows:

$$\acute{\theta} = \text{arctan2}(\nu_s^{11}, \nu_s^{10}) + \pi \tag{12}$$

where

$$\text{arctan2}(\nu_s^{11}, \nu_s^{10}) = \begin{cases} \theta, & \nu_s^{11} > 0 \text{ and } \nu_s^{10} > 0 \\ \pi - \theta, & \nu_s^{11} > 0 \text{ and } \nu_s^{10} < 0 \\ \theta - \pi, & \nu_s^{11} < 0 \text{ and; } \nu_s^{10} < 0 \\ -\theta, & \nu_s^{11} < 0 \text{ and } \nu_s^{10} > 0 \end{cases} \tag{13}$$

where $\theta \in [-\pi/2, \pi/2]$ and $\acute{\theta} \in [0, 2\pi]$. Finally, the quantization function is calculated as in Equation (14) [**?**].

$$\phi_t = f_q(\acute{\theta}) = \frac{2t}{T}\pi \ , \tag{14}$$

where

$$t = mod\left(\left\lfloor \frac{\acute{\theta}}{2\pi/T} + 0.5 \right\rfloor, T\right) \tag{15}$$

3. *WLD Histogram:* In this step, the WLD histogram is computed using the values of both the differential excitation (ξ_j) and orientation (ϕ_t) at each pixel. Hence, this histogram consists of (ξ_j, ϕ_t), $j = 0, 1, \ldots, N-1$ and $t = 0, 1, \ldots, T-1$, where N represents the dimensionality of an image and T denotes the number of the dominant orientation [**?**].

The *patch size* is a very important parameter affecting the accuracy and CPU time of the WLD algorithm. Changing the *patch size* changes the number of extracted features; hence, classification time and recognition rate are affected.

D. Local Binary Patterns (LBP)

LBP feature extraction method is used to extract local features from grayscale images. In LBP, the LBP code is calculated for each pixel by comparing each pixel with its neighbors, thus LBP method is relatively robust to rotation and poor quality images [**?**, **?**]. The LBP code is represented by binary number and it is calculated as follows:

$$LBP_{P,R} = \sum_{i=0}^{P-1} s(g_i - g_c)2^i, \text{ where } s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{16}$$

where g_c is the gray level of the center pixel, $g_i, i = 0, 1, \ldots, P-1$ represents the gray levels of the P equally spaced pixels around the center pixel (g_c), $LBP_{P,R}$ represents the LBP code when the number of neighbors pixels is P, $R(R > 0)$ is the radius or distance from the center to the neighboring pixels, and s is the threshold function of x as shown in Figure (4) [**?**, **?**].
To increase the robustness of LBP against rotation, LBP code is rotated until reach to its minimum as denoted in Equation (17).

$$LBP_{P,R} = \min\{\text{ROR}(LBP_{P,R}, i)\}, \ i = 0, 1, \ldots, P-1 \tag{17}$$

where $ROR(f, i)$ represents the circular bit-wise right shift on the f value i times.
There are two types LBP codes, namely, *uniform* and *non-uniform* patterns. In a uniform pattern, the bit-wise transition from zero to one or from one to zero is maximum one. For example, the binary values, 11111111 and 00000000 have zero transition, are uniform patterns. On the other hand, the values 01010101 and 11001100 have seven and three transitions, respectively, are non-uniform patterns [**?**, **?**].

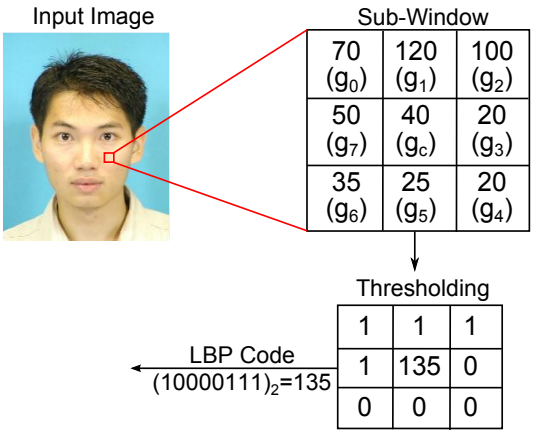


Figure. 4: Illustration of the computation of the LBP algorithm.

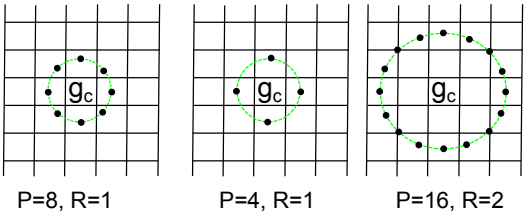


Figure. 5: Positions of the neighboring pixels according to different values of P and R.

After calculating the LBP code for each pixel of the gray image, a histogram is built to represent the texture of the image as follows:

$$H(k) = \sum_{i=1}^{I}\sum_{j=1}^{J} f(LBP_{P,R}, k), k \in [0, K] \tag{18}$$

where

$$f(x, y) = \begin{cases} 1 & , x = y \\ 0 & , \text{otherwise} \end{cases} \tag{19}$$

and K represents the maximum LBP code.
LBP method has two main parameters that affect the number features and robustness of LBP. The first parameter is the *radius (R)*. Due to a high correlation between neighborhood pixels, more information can be extracted from local neighbors, thus the radius R is usually small [**?**, **?**]. The *number of neighbors* P represents the second parameter. Changing the value of P will change the number of neighbors as shown in Figure (5). The neighbors that do not fall exactly on pixels are estimated using interpolation. Increasing the value of P increases the robustness of the LBP code against image scaling, translation, or noise [**?**, **?**].

E. Direct-Linear Discriminant Analysis (DLDA)

LDA is one of the common feature extraction and dimensionality reduction methods. The main objective of LDA is to search for LDA space that maximizes the Fishers' formula ($J(W) = \frac{W^T S_B W}{W^T S_W W}$) by: (1) Increasing the between-class variance, S_B, by increasing the distance between means/-centroids of different classes, (2) Decreasing the within-class

Algorithm 1 : Direct-LDA

1: Compute the mean of each class μ_i and total mean of all classes μ.
2: Compute S_B as follows:

$$S_B = \sum_{j=1}^{c} n_i(\mu_j - \mu)(\mu_j - \mu)^T \tag{20}$$

where μ_j is the mean of j^{th} class, μ is the total mean, n_i is the number of samples in each class, and c represents the number of classes.
3: Diagonalize S_B as follows:

$$V^T S_B V = \lambda \tag{21}$$

where V and λ represent the eigenvectors and eigenvalues of S_B, respectively.
4: Sort eigenvectors according to eigenvalues and neglect the eigenvectors that have eigenvalues less or equal to zero. The selected eigenvectors and eigenvalues are denoted by Y and D_b, respectively.
5: Let $Z = Y D_b^{-\frac{1}{2}}$, where Z used to reduce the dimension of S_B from n to m.
6: Use Z to diagonalize $Z^T S_W Z$, as follows:

$$U^T Z^T S_W Z U = Q \tag{22}$$

where U and Q represent the eigenvectors and eigenvalues of $Z^T S_W Z$, respectively.
7: Sort eigenvectors U according to eigenvalues Q and neglect eigenvectors that have high eigenvalues to increase the ratio of Fishers' formula. The selected eigenvectors and eigenvalues are denoted by R and D_w, respectively.
8: The final projection space (A) is calculated as follows, $A = R^T Z^T$,

variance, S_W, by decreasing the distance between the samples and mean of each class, where W is the transformation matrix that is used to project the original data on the LDA space [**?**, **?**].
Small Sample Size (SSS) is one of the main problems of L-DA. This problem occurs when the number of samples in classes is lower than the dimension of the feature vectors of those samples. Due to (1) high dimensionality in face sketch images and photos in this research, (2) the number of samples in each class is small; therefore, Direct-LDA method is used. In this method, the null space of S_B, which contains no useful information for classification, is removed. This step is achieved by diagonalizing S_B first then diagonalizes S_W [**?**]. Algorithm (1) summarizes the steps of Direct-LDA.

F. AdaBoost Classifier

AdaBoost (Adaptive Boosting) is a classifier ensemble algorithm and it consists of a number of weak learners. A weak learner/classifier is a simple and easy to implement classifier such as single level decision tree or simple neural networks [**?**]. In ensemble classifiers, the weak learners are first trained and then their decisions are combined to determine the final decision. AdaBoost classifier consists of two phases, namely, training and testing.
In the training phase, the parameters of AdaBoost classifier such as weights of all samples, and maximum iteration (T) are first initialized. The weights of all samples (w) are initialized to be equal and the weights will be adjusted for each iteration. For each iteration (t), the training samples are selected according to their weights (w), and these samples are used to build the weak learner (C_t). The difference between the predicted values and the true labels of the training samples, i.e. the training error rate of the current weak learner (ϵ_t) is then calculated. If the error rate is more than or equal to 0.5, i.e. 50% of the samples, the weights (w) are reinitialized and the error rate is recalculated again. The weight of the current weak learner, ($\alpha_t \in (0, 1)$), is then calculated as follows:

$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t} \tag{23}$$

As denoted in Equation (23), increasing the error rate increases the weight of the weak learner (α_t). The weights of the training samples are then updated at the end of each iteration to be used in the next iteration as denoted in Equation (24). In each iteration, the AdaBoost will focus on the misclassified patterns and the procedure is repeated for many iterations until the performance is satisfied [**?**].

$$w_j^{t+1} = \frac{w_j^t \alpha_t^{(1-l_j^t)}}{\sum_{i=1}^{N} w_i^t \alpha_t^{(1-l_i^t)}} \ , \ j = 1, 2, \ldots, N \tag{24}$$

In the testing phase, to classify an unknown sample (x_{test}), all weak learners of the AdaBoost classifier are used as denoted in Equation (25). The score of each class is calculated and then assigns the class that has a maximum score to the unknown sample.

$$\mu_t = \sum_{C_t(x_{test})=\omega_t} ln(\frac{1}{\alpha_t}) \ , \forall \ t = 1, 2, \ldots, T \tag{25}$$

where T represents the maximum number of iterations and it ranges from a few dozen to a few thousand, $C_t(x_{test})$ denotes the t^{th} weak learner, μ_t represents the score of a class ω_t, and α_t refers to the weight of the t^{th} weak learner. The number of iterations parameter, i.e. the number of weak learners, controls the size of the ensemble and it controls the accuracy of AdaBoost classifier.

III. Proposed Face-Sketch Recognition System

This section describes the proposed approach in detail. The proposed model depends on using the different local invariant features to extract robust features and then using the AdaBoost classifier to identify an unknown sketch. The approach, as illustrated in Figure (6), generally consists of three phases: feature extraction, feature reduction, and classification. These phases are explained below.

A. Feature Extraction Phase

In this phase, four local invariant methods were adapted to extract features. As shown in Figure (6), the photos were used to train the model in the training phase. In other words,

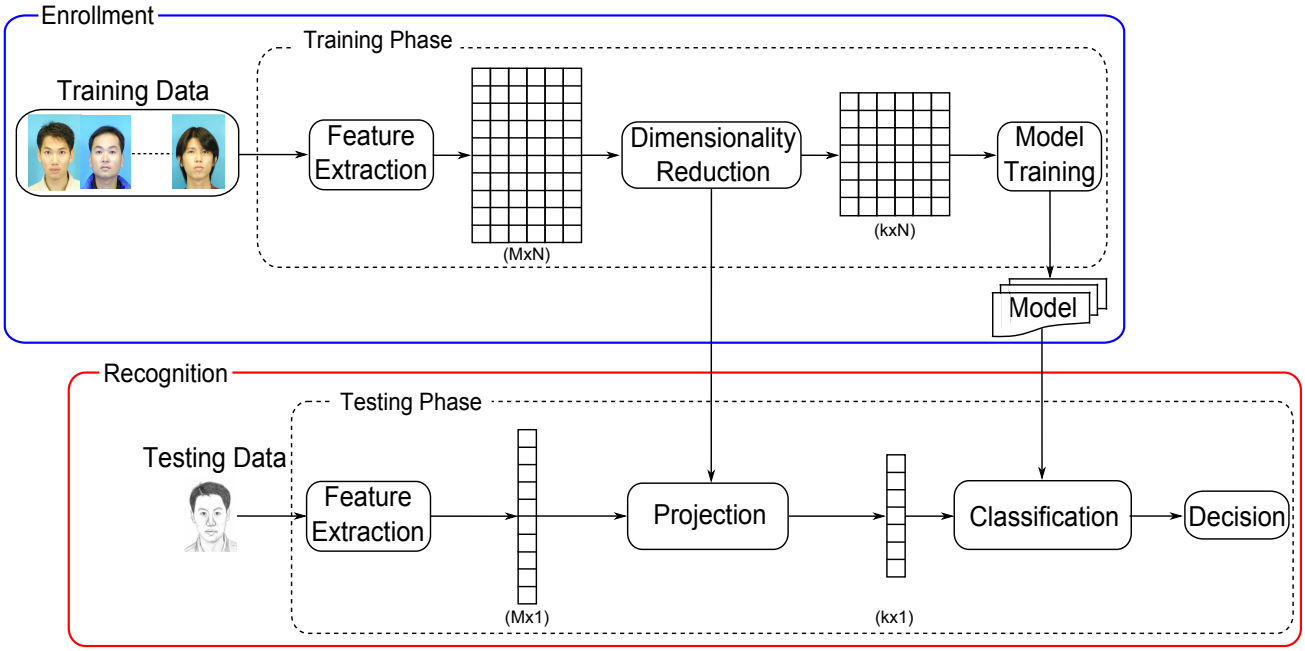


Figure. 6: A block diagram of the proposed model.

photos represent the training data, while the sketches were used as testing data in the testing phase. Each sample (x_i) of the training data was represented by a vector; hence, all the training samples were used to build the feature matrix ($X = \{x_1, x_2, \ldots, x_N\}$), where N is the number of photos, while the unknown sketch was represented by one feature vector.

B. Feature Reduction Phase

The output of the *feature extraction phase* is usually a high dimension feature vector because dense methods extract features from each pixel and spares method have a high number of interest points. To use these feature vectors in the classification phase, there will be a high computational cost and time-consuming process. That is to say that the high dimensionality will in turns affect the performance of the proposed approach. To address these issues, Direct-LDA (DLDA) algorithm, described in Section II-E, was applied on the output of the feature extraction phase. In other words, the DLDA method was applied to the feature matrix ($X \in \mathcal{R}^M$) which computed in the training phase to find the DLDA space, W, that reduces the dimension of the training data to $\mathcal{R}^k$ and separate different classes, i.e. photos. The feature vector of an unknown sketch image ($x_{test} \in \mathcal{R}^M$) was then projected on the DLDA space to reduce its dimension to $\mathcal{R}^k$ before starting the classification phase.

C. Classification Phase

Finally, in the classification phase, the proposed model classifies an input, a sketch image of unknown person, to the nearest photo. In this paper, a supervised learning classifier (AdaBoost) was used. As shown in Figure (6), the feature matrix, after projection onto the DLDA space, and the labels of the training samples represent the input to the AdaBoost classifier. The AdaBoost classifier model was then built by training one weak learner in each iteration and calculating the weight of that weak learner.

To automatically identify an unknown sketch image, all weak learners were used to classify that sketch image. The weighted voting method was then used to calculate the score of each class, and assign the class with the maximum score to the unknown image.

Figure. 7: Samples of face photos (top row) and corresponding sketches (bottom row) for different three individuals.

IV. Experimental Results and Discussion

The proposed approach was evaluated using CHUK face sketch dataset which consists of 606 individuals. Each person has a frontal face photo image and a sketch image drawn by an artist. The dataset was divided into two sets, namely, training and testing sets. The training set consists of 306 individuals, while the other images were used as a testing set.

Table 1: Accuracy (in %) and CPU time (secs) of the proposed model using SIFT method according to different angles.

Number of Angles	Length of Feature Vector	CPU Time	Accuracy
4	64	0.44	94.5
8	128	0.64	99.33
16	256	1.05	99.67

The size of all photos and sketches were 250×200; Figure (7) shows samples of the used dataset. In this section, four experiments were conducted to investigate the robustness of the feature extraction methods that were used to extract features from photo and sketch images. In each experiment, one of the local invariant feature extraction methods that were mentioned in Section II was used.
In the first experiment, SIFT method was used. This experiment has three sub-experiments to tune up parameters of SIFT method. The second experiment was conducted to investigate the robustness of SURF feature extraction method. In the third experiment, WLD was used as a feature extraction method. The fourth experiment was conducted to test the robustness of LBP feature extraction method. In all the four experiments, AdaBoost classifier was used to match the unknown sketch with the training photos. The size of the AdaBoost classifier was 15. One more experiment was conducted to evaluate the influence of the size of AdaBoost classifier on the accuracy and CPU time of the proposed model. Herein, the accuracy of our method is given as a ratio of the number of correctly classified samples to the total number of samples. It is computed as follows:

$$Accuracy = \frac{\#\text{Correct Classification}}{\#\text{Total Samples}} \tag{26}$$

The experiments in this paper were conducted using a PC with Intel(R) Core(TM) i5-2400 CPU @ 3.10 GHz, and 4.00 GB RAM. The Matlab platform was used and it was run under windows 32-bit operating system.

A. SIFT Experiments

The aim of this experiment was to evaluate the robustness of SIFT method. In this experiment, three sub-experiments were conducted to test the influence of the parameters of SIFT method (peak threshold, patch size, and the number of angles) on the accuracy and CPU time of the proposed model. In the first sub-experiment, the value of peak threshold parameter was 0, 0.05, 0.1, 0.15, and 0.25. Figure (8) shows the results of this experiment. In the second sub-experiment, a different number of angles was used to extract features from different orientations. The number of angles was 4, 8, and 16. Table (1) summarizes the result of this sub-experiment. In the third sub-experiment, different values of patch size parameter were used to evaluate the accuracy of the proposed model. The patch size was 64×64, 32×32, 16×16, 8×8, 4×4, and 2×2. Table (2) summarizes the results of this sub-experiment.
From Figure (8), it can be noticed that the accuracy inversely proportional to the value of peak threshold parameter. Moreover, the number of key-points decreased when the peak

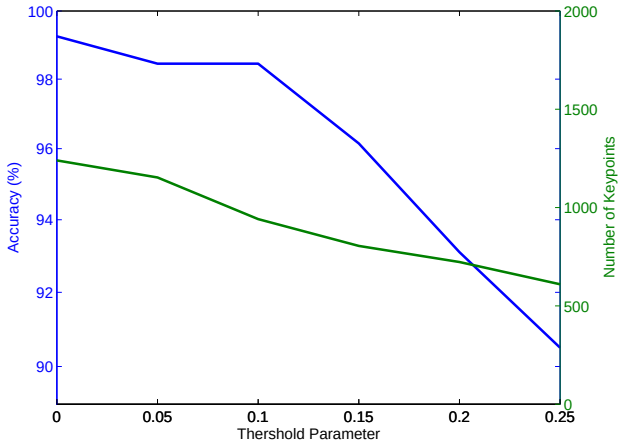


Figure. 8: Accuracy and number of key-points of the proposed model using SIFT method with different values of the peak threshold parameter.

Table 2: Accuracy (in %) and CPU time (secs) of SIFT method using different values of patch size parameter.

Patch Size	No of Patches	CPU Time	Accuracy
64×64	$3 \times 3 = 9$	5.5	91.56
32×32	$7 \times 6 = 42$	1.18	95.61
16×16	$15 \times 12 = 180$	0.64	99.33
8×8	$31 \times 25 = 775$	0.48	97.54
4×4	$62 \times 50 = 3100$	0.44	93.44
2×2	$125 \times 100 = 12500$	0.43	83.6

threshold increases. Thus, classification time decreases. Table (1) shows the accuracy and CPU time when the number of angles was, 4, 8, and 16. From the table, it can be noticed that the accuracy was increased when the number of angles increases from four to eight angles. The reason for that is increasing the number of angles extracts features from different orientations, thus, the features are more robust against rotation. However, increasing the number of angles increases the length of the feature vector; hence, needs more classification time. Generally, using eight angles will save more time and extract discriminative features. According to the patch size parameter, as shown in Table (2), increasing the patch size decreases the number of patches, decreases the number of key-points, thus SIFT will be considered as global and extracting features from each patch needs more CPU time. On the other hand, small patch size increases the number of key-points, but it takes less CPU time to extract features. Further analysis to this parameter showed that the accuracy of SIFT using different values of patch sizes fluctuated and the best accuracy achieved when the patch size was 16×16.

B. SURF Experiment

The aim of this experiment was to evaluate the contrast threshold of SURF method on the accuracy of the proposed model. In this experiment, different values of the contrast threshold parameter were used to show how this parameter affects the accuracy and CPU time. Figure (9) shows the results of this experiment.
From Figure (9), it can be noticed that the accuracy inversely proportional to the value of the contrast threshold parameter.

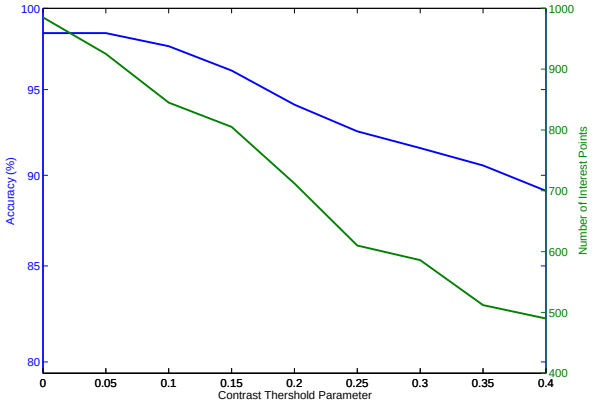
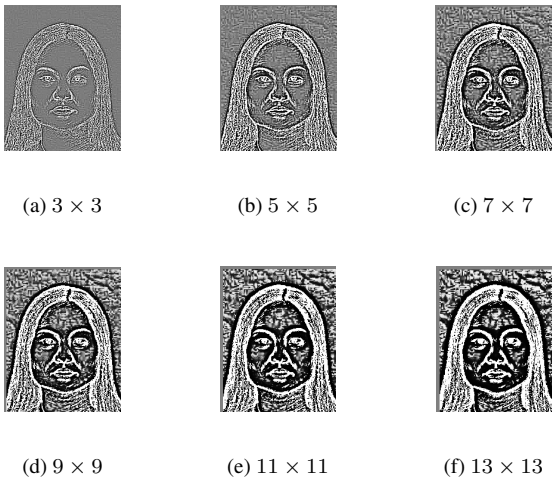


Figure. 9: Accuracy and number of interest points of the proposed model using SURF method with different values of the contrast threshold parameter.

In terms of the number of interest points, the number of interest points decreased when the contrast threshold increases. Hence, increasing the contrast threshold reduces the required classification time.

C. WLD Experiment

In this experiment, different patch sizes were conducted to investigate the impact of the WLD patch size parameter on the accuracy and CPU time of the proposed model. Changing the value of patch size parameter changes the robustness of WLD features. Figure (10) shows WLD features using different values of the patch size parameter. Table (3) illustrates the results of this experiment.



(a) 3×3 (b) 5×5 (c) 7×7

(d) 9×9 (e) 11×11 (f) 13×13

Figure. 10: WLD features using different values of patch size parameters.

From Table (3), it was found that the most suitable size for the patch parameter was 7×7. This is because it allowed our model to achieve an accuracy rate significantly better than the other sizes. Moreover, it can be noticed that increasing the patch size led to decreasing the length of the feature vectors, consequently decreasing the CPU time for classification. Thus, the 7×7 patch size did not take more CPU time comparing with the other patch sizes.

Table 3: Accuracy (in %) and CPU time (secs) of the WLD method using different values of patch size parameter.

Patch Size	CPU Time	Accuracy
3×3	1.64	91.56
5×5	0.95	94.65
7×7	0.54	96.67
9×9	0.41	95.33
11×11	0.32	94.65
13×13	0.24	90.6

Table 4: Accuracy (in %) and CPU time (secs) for LBP method using different values of P and R.

P	R	CPU Time	Accuracy
4	1	0.232	85.5
4	2	0.241	83.6
4	3	0.245	81.4
8	1	0.248	91.5
8	2	0.256	93.5
8	3	0.262	95.2
16	2	0.38	96.33
16	3	0.426	94.6

D. LBP Experiment

As mentioned in Section II-D, LBP method has only two parameters namely, P which represents the number of neighbors and R which represents the radius. In this experiment, different values of P and R were used to show how these two parameters affect the accuracy and CPU time. Table (4) summarizes the results of this experiment.

Table (4) shows that increasing the values of P and R increases the CPU time slowly. The reason is that the calculations for each pixel were increased when the P and R increased. For example, when $P = 4$, the LBP code is calculated by comparing each pixel with its four neighbors, while when $P = 16$ we need to compare the current pixel with 16 neighbors which needs more time and memory space. Moreover, increasing P increases the length of the feature vector. According to the accuracy, we note that the best accuracy achieved when $P = 16$ and $R = 2$.

E. Ensemble Size Experiment

The aim of this experiment was to investigate the accuracy rate of our face sketch recognition model when changing of the ensemble size, L, of the AdaBoost classifier. In this experiment, the size of the AdaBoost ensemble was 5, 15, 25, and 35. Table (5) summarizes the experimental results of this experiment.

From Table (5) two main remarks can be seen. Firstly, the CPU time of the proposed model was increased when the size of the ensemble increases. In other words, increasing the ensemble size increases the number of weak learners; hence, training the model needs more time. Secondly, the accuracy also increased when the size of the ensemble increases. The analysis of this point is that increasing the ensemble size, i.e. weak learners, accordingly, more classifiers were used to predict the class of the unknown sample.

Table 5: Accuracy (in %) and CPU time (secs) of the proposed model using SIFT, SURF, WLD, and LBP methods with different ensemble sizes.

Ensemble Size	SIFT		SURF		WLD		LBP	
	Accuracy	CPU Time	Accuracy	CPU Time	Accuracy	CPU Time	Accuracy	CPU Time
5	99.33	0.64	98.33	0.51	96.67	0.54	96	0.38
15	99.33	1.21	98.33	1.12	96.67	0.98	96.33	0.46
25	99.67	3.25	98.67	2.94	96.67	1.74	96.67	0.74
35	99.67	7.45	98.67	4.56	97	3.05	96.67	0.82

F. Further Analysis

From the experimental results obtained from the five experiments, the following remarks can be noticed. First, for SIFT method, the most suitable value of the peak threshold parameter was zero because it allows more key-points than other values. Moreover, using eight angles achieved high accuracy and low CPU time. In addition, 16×16 patch size parameter achieved the highest accuracy. Generally, SIFT method achieved high accuracy (99.67%) compared with the three other methods. Second, SURF algorithm achieved high accuracy (98.33%) when the contrast threshold parameter was low and the highest accuracy achieved when the contrast threshold was zero. Third, WLD method achieved high accuracy (97%) when the patch size was 7×7. Fourth, LBP achieved the lowest accuracy (96.67%) among the other three methods. The most suitable values of P and R parameters were 16 and 2, respectively. Fifth, sparse, i.e. SIFT and SURF, methods achieved accuracy better than dense, i.e. WLD and LBP, methods. This is because sparse methods extract features from different levels, i.e. scales, of the original image while in dense methods the features are extracted from only the original scale. Sixth, increasing the size of AdaBoost ensemble increases the accuracy, but needs more CPU time.

To further prove that our model is better than other related work, as illustrated in Table (6), a comparison with the most related work [**?**, **?**, **?**, **?**] was conducted. From this table, it can be remarked that although our approach used the same dataset (CHUK dataset), local invariant features (both sparse and dense) methods achieved high accuracy results and sparse methods achieved results better than dense methods.

1) Sparse vs. Dense Methods

As mentioned in Section I, there are two main methods to extract local invariant features: dense and sparse methods. To justify why sparse methods achieved accuracy better than dense methods in this work, a comparison between the two methods that were used in this work is made.

There are two main differences between sparse and dense methods. Firstly, the computation time of LBP and WLD depend mainly on the size of the image. On the other hand, in SIFT and SURF methods, the scale space step needs more time. Hence, the complexity of sparse methods is much higher than dense methods [**?**]. Secondly, in terms of accuracy, sparse methods achieve high accuracy compared with dense methods as proved in our experiments. This is because sparse methods are applied on different scales while in dense methods the features are only extracted from the original scale which reduces the accuracy of the dense methods than the sparse ones.

Table 6: A comparison between some state-of-the-art models and the proposed model.

Author	Method	Accuracy (in %)
X. Tang *et al.* [**?**]	Eigentransform+PCA	75
X. Tang *et al.* [**?**]	Eigentransform+Bays	81.3
X. Tang *et al.* [**?**]	PCA	81.3-97
Qingshan Liu *et al.* [**?**]	LDA	85
Qingshan Liu *et al.* [**?**]	PCA	64.33
Xinbo *et al.* [**?**]	E-HMM+PCA	95.24
Wang *et al.* [**?**]	Multiscale−MRF−SP	96.3
Proposed Model	SIFT	99.67
	SURF	98.33
	WLD	97
	LBP	96.67

Markov Random Field (MRF), Sketch/ Photo (SP)

V. Conclusions and Future Work

In this paper, a face sketch recognition model was proposed. In this model, two types of local invariant feature extraction methods, namely, dense and sparse methods were used. SIFT and SURF methods were used as sparse methods, while LBP and WLD methods were used as dense methods. Moreover, AdaBoost classifier was used to match the features of the unknown sketch with the features of the photos. Four different experiments were performed to test the robustness of the four feature extraction methods. Another experiment was conducted to evaluate the impact of the ensemble size on the accuracy and CPU time. Due to high dimensionality in both sparse and dense methods, Direct-LDA method was used to reduce the number of features.

In terms of accuracy rate and CPU time, the best performance of SIFT algorithm achieved when peak threshold=0, the number of angles=8, and patch size=16×16. According to SURF method, the best accuracy achieved when the contrast threshold was zero. For WLD method, the best accuracy achieved when the patch size was 7×7. For LBP method, the best accuracy achieved when P=16 and R=2. In general, SIFT method achieved the best accuracy (99.67%) when the ensemble size was more than 25. Thus, it could be concluded that the proposed model has achieved an excellent accuracy against many state-of-the-art methods.

In the future work, bio-inspired optimization algorithms are used to select the most discriminative features. Moreover, further analysis is performed to increase the accuracy of the future model.