**CS3330 REPORT**

**1. Background**

What problem (e.g. face detection, illumination normalization) are you working on and how• does it contribute to facial recognition?

Eyes detection:

Eyes detection in human face images plays an important part in facial recognition. Accurately extracting the eyes region help us performing normalisation of the human face (i.e., facial features alignment), which is a crucial pre-processing step of various face recognition techniques.

Why is your chosen problem challenging and how can these challenges be addressed with image processing techniques?

color based method doesn't work on well on ethical people.

my algorithm fast

What are the existing approaches for your chosen problem? Briefly outline THREE approaches to your application, found through your research and discuss the way in which the aims and performances of these approaches differ in academic literature.

There are several existing approaches for extracting the location of the eyes.

Most of modern face detection algorithms are developed based Viola-Jones object detection framework [1], which based on series of Haar like features. Haar cascades are the combination of multiple weak classifiers. The downside of this algorithm is the slow learning process and large number of features.

Rajpathak [2] was able to detect the face's skin region by performing six-sigma operation on mixture of NTSC, HSV and RGB spaces. When an eye is well illuminated, it has a very sharp reflection point .Because of this fact; morphological operations had been used to detect the spot between the two eyes. Having to use three color space models, a high complexity is the main limitation of this algorithm.

The third algorithm that I have looked into was proposed by T.Kawaguchi [3] which detects eyes with region growing and circular Hough transformation.. Slow and expensive computationally was the major disadvantage of this algorithm.

**2. Algorithm**

Motivate why it would be suitable for solving your chosen problem, making reference to the literature.

Discuss any drawbacks to the algorithm, again, making reference to the literature:

- Algorithm doesn't work well when the image is not well illuminated

- Only works with passport like photo

- Detect wrong features like nose trills or eyes brow to be eyes

Describe how the method works in your own words (e.g. using high-level pseudocode).

**3. EXPERIMENTAL RESULTS**

I tested my algorithm on PICS [6] facial images database. 90 random facial photos were selected including.

The proposed idea was implemented in Matlab using image processing toolbox. The proposed algorithm was tested on PICS [6] facial images database. 80 photos of 50 individuals including both males and females of different ages and ethnicity were randomly selected. Most of them were of frontal faces. However, some faces were tilted to left or right. These are coloured images with Widths vary from 360 to 480 pixels while their heights vary from 480 to 540 pixels. On Intel core duo 1.60 GHz Processor, the average processing time of our method was one second. Table 1 shows the results and percentage accuracy of our algorithm. The table shows that our algorithm is very accurate in initial blobs extraction stage. From the table it is also clear that accuracy is more when the input images are good illumination images.

Blob extract : 4 failed

Eyes extract: 14 times eyes not detected, 12 times wrong features,

**4. Conclusions**

- Summarize the findings from your report and attempt to draw general conclusions or make a general hypothesis based on your results:

- Discuss any limitations of your research:

I was not able to implement other algorithms to compare with mine.

Kawaguchi and Rizon proposed an algorithm in which eyes were extracted by the intensity valley from the face region using edge detections, morphology, region growing algorithm, circular Hough transformation and feature template approaches [7]. The limitation of this algorithm was computationally slow.

The illumination spot between eyes had been detected by morphological operations and this was based on the fact that an eye had a sharp point of reflection whenever it was being well illuminated [11]. The major drawback of this algorithm was increased complexity due to many color space models.

Kawaguchi and Rizon proposed an algorithm in which eyes were extracted by the intensity valley from the face region using edge detections, morphology, region growing algorithm, circular Hough transformation and feature template approaches [7]. The limitation of this algorithm was computationally slow.

The location of the eyes is the most commonly used features to perform face normalisation (i.e., alignment of facial features), which is an essential pre-processing stage of many face recognition systems.

slow learning method and large number of features were major drawback of this algorithm.

Viola and Jones used integrated image, Adaboost learning algorithm with Haar like features, and combining the classifier in a cascade for face detection [8]. Weak classifier, slow learning method and large number of features were major drawback of this algorithm.

Various researchers have been working on face detection and eyes extraction in human face image for two decades due to its numerous applications in various fields. The extraction of eyes from face region was achieved by many researchers while certain researchers extract face region with the help of eyes points. The related work of different face detection and eyes extraction methods are discussed here.

in computer vision applications such as face recognition, face verification, gaze estimation and human computer interaction. The position of the eyes can be extracted from an image using different eye features such as colour, illumination, edge, shape and geometrical information. Most colour variations occur in eye regions in facial images due to colour difference between eyelids and skin, skin and sclera, sclera and iris, and iris and pupil (see figure 1). The colours of the rest of the face are more uniform when compared to the eyes. Therefore, if edge detection was applied to facial images then the eye regions would have maximum edge densities. The proposed method uses this edge density information to extract the eyes from facial images

proposed : delete small component one more time before apply rules. (Area>100). Reduce the chances of mistaken small are to eyes. (nose strills..)

RULES:

Proposed: distance between two eyes shouldn't be too large or too small compare to their bounding box size.

Increase eyes orientation to 90.0

eyes slope decrease to 30.0

eyes size ratio to 3.0

function [ eyes\_detected\_img ] = eyes\_detect( face\_img )  
 %EYES\_DETECT Detect eyes region then draw box around them  
  
 %convert image to gray image  
 gray\_img = rgb2gray(face\_img);  
 %Edge detection with sobel  
 edge\_img = edge(gray\_img,'sobel');  
 % Dilate the image twice  
 dilated\_img = dilate(edge\_img);  
 %Inverse the image then fill the small holes  
 negative\_dilated\_img = imcomplement(dilated\_img);  
 filled\_img = imcomplement(rm\_small\_cmp(negative\_dilated\_img, 300));  
 %Erode the image three times  
 eroded\_img = erode(filled\_img);  
  
 %Aspect ratio rule  
 aspect\_img = aspect\_ratio\_rule(eroded\_img);  
  
 %The orientation angle of eyes is not greater than 45 degrees.  
 angle\_img = rm\_large\_orient(aspect\_img,100.0);  
  
 %Remove small components  
 rm\_img = rm\_small\_cmp(angle\_img, 120);  
  
 %Apply rules that compare two eyes  
 hpadding = 40.0;  
 vpadding = 80.0;  
 eyes\_ratio = 3.0;  
 max\_orient\_diff = 30.0;  
 eyes\_slope\_angle = 15.0;  
 eyes\_binary\_img = two\_eyes\_rule(rm\_img,hpadding,vpadding,eyes\_ratio, max\_orient\_diff, eyes\_slope\_angle);  
 %Draw box around the eyes  
 eyes\_detected\_img = draw\_box(eyes\_binary\_img, face\_img );  
% eyes\_detected\_img = dilated\_img;  
  
end

function [eyes\_binary\_img] = two\_eyes\_rule(img,hpadding,vpadding,eyes\_ratio, max\_orient\_diff, eyes\_slope\_angle)  
 CC = bwconncomp(img,4);  
 stats = regionprops(CC,'all');  
  
 idx\_size = [];  
 [y,x] = size(img);  
 for i1 = 1 : length(stats)  
 comp1 = stats(i1);  
 for i2 = 1:length(stats)  
 if i1 == i2  
 continue;  
 end  
 comp2 = stats(i2);  
  
 far\_from\_border = far\_from\_border\_rule( comp1,comp2, hpadding,vpadding,x,y);  
 ratio\_match = ratio\_rule (comp1, comp2, eyes\_ratio);  
 orient\_match = orient\_rule (comp1, comp2, max\_orient\_diff);  
 slope\_angle\_match = slope\_angle\_rule (comp1, comp2 ,eyes\_slope\_angle);  
 if far\_from\_border && ratio\_match && orient\_match && slope\_angle\_match  
 idx\_size = [idx\_size i1 i2];  
 break;  
 end  
 end  
 end  
 eyes\_binary\_img = ismember(labelmatrix(CC), idx\_size);  
end

function [ far\_from\_border ] = far\_from\_border\_rule( comp1,comp2 ,hpadding,vpadding,x,y)  
 bb1 = comp1.BoundingBox;  
 far\_from\_border1 = hpadding<bb1(1) && bb1(1)+bb1(3)<x-hpadding && vpadding<bb1(2)&& bb1(2)+bb1(4)<y-vpadding;  
 bb2 = comp2.BoundingBox;  
 far\_from\_border2 = hpadding<bb2(1) && bb2(1)+bb2(3)<x-hpadding && vpadding<bb2(2)&& bb2(2)+bb2(4)<y-vpadding;  
 far\_from\_border = far\_from\_border1 && far\_from\_border2;  
end

function [ratio\_match] = ratio\_rule (comp1, comp2, eyes\_ratio)  
 ratio = comp1.Area/comp2.Area;  
 ratio\_match = ratio > 1/eyes\_ratio && ratio < eyes\_ratio;  
end

function [orient\_match] = orient\_rule (comp1, comp2, max\_orient\_diff)  
 orient\_match = comp1.Orientation - comp2.Orientation < max\_orient\_diff;  
end

function [slope\_angle\_match] = slope\_angle\_rule (comp1, comp2, eyes\_slope\_angle)  
 bb1 = comp1.BoundingBox;  
 bb2 = comp2.BoundingBox;  
 center1 = [bb1(1)+bb1(3)/2,bb1(2)+bb1(4)/2 ];  
 center2 = [bb2(1)+bb2(3)/2, bb2(2)+bb2(4)/2];  
 slope\_angle = atan2(center2(2)-center1(2),center2(1)-center1(1))\* 180/pi;  
 slope\_angle = abs(slope\_angle);  
 if slope\_angle > 90.0  
 slope\_angle = 180.0 - slope\_angle;  
 end  
 slope\_angle\_match = slope\_angle < eyes\_slope\_angle;  
end

function [ aspect\_ratio\_img ] = aspect\_ratio\_rule( img )  
% aspect\_ratio: Keep components that have 0.8 < w/h < 4.0  
  
 CC = bwconncomp(img,4);  
 stats = regionprops(CC,'all');  
 %Aspect ratio rule  
 idx\_boundingbox = [];  
 for k = 1 : length(stats)  
 BB = stats(k).BoundingBox;  
 aspect\_ratio = BB(3)/BB(4);  
 if aspect\_ratio > 0.8 && aspect\_ratio < 4.0  
 idx\_boundingbox = [idx\_boundingbox k];  
 end  
 end  
  
 aspect\_ratio\_img = ismember(labelmatrix(CC), idx\_boundingbox);  
  
end

function [ dilated\_img ] = dilate( edge\_img )  
 %dilate Dilate img twice  
 SE = strel('disk', 3);  
 dilated\_img = imdilate(edge\_img,SE);  
 dilated\_img = imdilate(dilated\_img,SE);  
  
end

function [ eroded\_img ] = erode( filled\_img )  
 %erode Erode image three times  
  
 SE = strel('disk', 3);  
 eroded\_img = imerode(filled\_img,SE);  
 eroded\_img = imerode(eroded\_img,SE);  
 eroded\_img = imerode(eroded\_img,SE);  
  
end

function [ angle\_img ] = rm\_large\_orient( img, max\_orientation )  
%rm\_large\_orient: Remove component with orientation larger than a choosen  
%angle  
  
CC = bwconncomp(img,4);  
stats = regionprops(CC,'Orientation');  
idx = find([stats.Orientation] <= max\_orientation);  
angle\_img = ismember(labelmatrix(CC), idx);  
% figure(),imshow(angle\_img),title('angle');  
  
end

function [ rm\_img ] = rm\_small\_cmp( img, min\_area )  
%rm\_small\_cmp remove components that are small than min\_area  
% Detailed explanation goes here min\_are rm2 = 120  
 CC = bwconncomp(img,4);  
 stats = regionprops(CC,'Area');  
 idx = find([stats.Area] > min\_area);  
 rm\_img = ismember(labelmatrix(CC), idx);  
end

function [ eyes\_detected\_img ] = draw\_box( final\_binary\_img, original\_img )  
%draw\_box: Draw box around the eyes region  
% Detailed explanation goes here  
  
CC = bwconncomp(final\_binary\_img,4); %final image  
stats = regionprops(CC,'BoundingBox');  
eyes\_detected\_img = original\_img;  
 for k = 1 : length(stats)  
 BB = stats(k).BoundingBox;  
 f = @() rectangle('Position', [BB(1),BB(2),BB(3),BB(4)]);  
 params = {{'EdgeColor','r','LineWidth',2}};  
 eyes\_detected\_img = insertInImage(eyes\_detected\_img,f,params);  
% rectangle('Position', [BB(1),BB(2),BB(3),BB(4)], 'EdgeColor','r','LineWidth',2 );  
 end  
  
end

**References**

[1] Kawaguchi, T. and Rizon, M. (2003). Iris detection using intensity and edge information. *Pattern Recognition*, 36(2), pp.549-562.

[2] Rajpathak, T., Kumar, R. and Schwartz, E. (2009). Eye detection using morphological and color image processing. *Proceeding of Florida Conference on Recent Advances in Robotics*, pp.1-6.

[3] Shafi, M. and Chung, P. (2008). Eyes extraction from facial images using edge density”. *7th IEEE International Conference on Cybernetic Intelligent Systems (CIS)*, pp.1-6.

[4] Viola, P. and Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2), pp.137-154.

[5] Singh, A., Singh, M. and Singh, B. (2016). Face detection and Eyes extraction using Sobel Edge Detection and Morphological Operations. *2016 Conference on Advances in Signal Processing (CASP)*

[6] PICS, 2003. Psychological Image Collection at Stirling (PICS image data base). Available from , University of Stirling Psychology Department.