Evaluation of Face Detection and Recognition Algorithms on Avatar Face Datasets

Cybersecurity Lab
University of Louisville

Presented by
Gyuchoon Cho
Face Detection

- Face detection is a computer technology which determines the location and size of human faces in arbitrary images. (Wikipedia)

- Nowadays, face detection technology is ubiquitous in the world.

  - Surveillance cameras - with suspect detection.
  
  - Fuji film's face-focusing, Sony's smile shutter, Apple iPhoto.
  
  - System log on with face recognition
Examples of Face Detection

HOW 2D FACIAL SCANNERS RECORD IDENTITIES

1. Detecting....

2. Matching with Database
   Name: Ali Reza
   Date: 25 May 2007
   Place: Main corridor

3. Recording
   Name: Unknown
   Date: 25 May 2007
   Place: Main corridor

Scanner starts reading features on a grid
Points are transformed to a database as an algorithm of numbers
Comparison can be made quickly by a computer program
Once a match is found an identity can be verified

Report

University of Louisville
Computer Engineering and Computer Science
Cybersecurity Lab
[Face Detection Techniques]
Face Detection tasks

face detection can be regarded as a specific case of

- Object - Class detection
  The task is to find the locations and size of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.

- Face Localization
  The task is to find the locations and size of a known number of faces.
**Face Localization**

![Image of a group of people with face localization overlay]

**Probable Face Pixels**

Lighter pixels mean higher probability of being a face pixel.
Face Localization

Color Segmented Mask
Mask produced from thresholding the filtered probability image.
Model fitting by scaling, translation and rotation.

ex) Hausdorff Distance is a specific case of Object Class Detection
Software Development Kits

- **OpenCV**
  OpenCV is a computer vision library originally developed by Intel. The library is cross-platform, and runs on Windows, Mac OS X, Linux, VCRT (Real-Time OS on Smart camera) and other embedded devices. It focuses mainly on real-time image processing.

- **VeriLook**
  VeriLook SDK is based on the VeriLook PC-based face recognition technology and is intended for biometric systems developers and integrators. It allows rapid development of biometric applications using functions from the VeriLook library, which ensure high reliability of the face identification. VeriLook facial recognition software is now available for Mac OS X.

- **Luxand Face SDK**
  - Detection of 40 facial feature points such as eyes, mouth, nose, and face.
  - Detection time: 0.9 seconds.
  - Up to 5,000 faces per second.
System Environment

- MATLAB R2009a
- Codegear RAD studio. (Delphi 2009)
- Rapid Application Development Programming Tool
- Verilook 4.0
- SQL lite
System Work Flow
Algorithm

- Advanced face localization
  - The algorithm implements advanced face localization, enrollment and matching using robust digital image processing algorithms.

- Fast identification speed - 100,000 faces per second
  - Tested on a PC with Intel Core 2 processor running at 2.66 GHz and 640 x 480 pixels of image
Roll, Pitch and Yaw #1

- Head roll (tilt) - ±180 degrees (configurable); ±15 degrees recommended
- Head pitch (nod) - ±15 degrees from frontal position
- Head yaw (bobble) - ±15 degrees from frontal position
## Roll, Pitch and Yaw #2

<table>
<thead>
<tr>
<th></th>
<th>Maximized Template Size</th>
<th>Medium Template Size</th>
<th>Minimum Template Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection time for all faces in a frame (±15° head roll tolerance)</td>
<td></td>
<td></td>
<td>10 milliseconds</td>
</tr>
<tr>
<td>Detection time for all faces in a frame (±180° head roll tolerance)</td>
<td></td>
<td></td>
<td>135 milliseconds</td>
</tr>
<tr>
<td>Single face template extraction time (1) (milliseconds)</td>
<td>111</td>
<td>62</td>
<td>31</td>
</tr>
<tr>
<td>Matching speed (2) (face records per second)</td>
<td>24,000</td>
<td>44,000</td>
<td>200,000</td>
</tr>
<tr>
<td>Template size in database (3) (bytes)</td>
<td>20,440</td>
<td>11,368</td>
<td>2,256</td>
</tr>
</tbody>
</table>
# Enrollment

![Progress Window](image)

<table>
<thead>
<tr>
<th>SL-100g.tif</th>
<th>YES</th>
<th>2.284</th>
<th>60.58001</th>
<th>50</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Detected</td>
<td>559</td>
<td>Percentag</td>
<td>79.85%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average of Confidence</td>
<td>56.48021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Process Time</td>
<td>1253.339</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Database #1

features

...
if (not database.TableExists('Templates')) then
begin
    createTable := 'CREATE TABLE Templates (Id INTEGER PRIMARY KEY, ' +
                    'TemplateName TEXT NOT NULL, ' +
                    'Template BLOB NOT NULL, ' +
                    'Thumbnail BLOB)';

database.ExecSQL(createTable);
end;
if (not database.TableExists('Images')) then
begin
    createTable := 'CREATE TABLE Images (Id INTEGER PRIMARY KEY, ' +
                    'TemplateName INTEGER NOT NULL, ' +
                    'BiometricType INTEGER NOT NULL, ' +
                    'RecordIndex INTEGER NOT NULL, ' +
                    'ConIndex INTEGER NOT NULL, ' +
                    'FrameIndex INTEGER NOT NULL, ' +
                    'Image BLOB NOT NULL)';

database.ExecSQL(createTable);
end;
Result #1

Total Image found: 331
[Result #2]

Total Image found: 317
[Color Structure Descriptor]

- Main functionality of this descriptor is **image to image** matching
  - it expresses local color structure in an image by use of a structuring element that is comprised of several images sample.

- The CSD is a generalization of the color histogram that captures some spatial characteristics of the color distribution in an image.
  - It is defined in the HMMD color space using non-uniform color quantization to between 32-256 colors.
max value = maximum value from input r, g, b values;
min value = minimum value from input r, g, b values;
diff = max value - min value;
sum = (max + min)/2; and

hue = (g - b) / (max - min) * 60, if (r = max ∧ (g - b) ≥ 0)
hue = (g - b) / (max - min) * 60 + 360, if (r = max ∧ (g - b) < 0)
hue = (2.0 + (b - r)) / (max - min) * 60, if (g = max)
hue = (4.0 + (r - g)) / (max - min) * 60, if (g = max)
hue = 0, if max = min.
CSD result

Match results:
- Face1: SL-C7C6
  Similarity: 180
- Face2: SL-C7C6
  Similarity: 180
- Face3: SL-C7C6
  Similarity: 180
- Face4: SL-C7C6
  Similarity: 180
- Face5: SL-C7C6
  Similarity: 180
- Face6: SL-C7C6
  Similarity: 180
- Face7: SL-C7C6
  Similarity: 180
- Face8: SL-C7C6
  Similarity: 180
- Face9: SL-C7C6
  Similarity: 180
- Face10: SL-C7C6
  Similarity: 180
- Face11: SL-C7C6
  Similarity: 180
- Face12: SL-C7C6
  Similarity: 180
- Face13: SL-C7C6
  Similarity: 180
- Face14: SL-C7C6
  Similarity: 180
- Face15: SL-C7C6
  Similarity: 180
- Face16: SL-C7C6
  Similarity: 180
- Face17: SL-C7C6
  Similarity: 180
- Face18: SL-C7C6
  Similarity: 180
- Face19: SL-C7C6
  Similarity: 180
- Face20: SL-C7C6
  Similarity: 180
- Face21: SL-C7C6
  Similarity: 180
- Face22: SL-C7C6
  Similarity: 180
- Face23: SL-C7C6
  Similarity: 180
- Face24: SL-C7C6
  Similarity: 180
- Face25: SL-C7C6
  Similarity: 180
- Face26: SL-C7C6
  Similarity: 180
- Face27: SL-C7C6
  Similarity: 180
- Face28: SL-C7C6
  Similarity: 180
- Face29: SL-C7C6
  Similarity: 180
- Face30: SL-C7C6
  Similarity: 180
- Face31: SL-C7C6
  Similarity: 180
- Face32: SL-C7C6
  Similarity: 180
- Face33: SL-C7C6
  Similarity: 180
- Face34: SL-C7C6
  Similarity: 180
- Face35: SL-C7C6
  Similarity: 180
- Face36: SL-C7C6
  Similarity: 180
- Face37: SL-C7C6
  Similarity: 180
- Face38: SL-C7C6
  Similarity: 180
- Face39: SL-C7C6
  Similarity: 180
- Face40: SL-C7C6
  Similarity: 180
- Face41: SL-C7C6
  Similarity: 180
- Face42: SL-C7C6
  Similarity: 180
- Face43: SL-C7C6
  Similarity: 180
- Face44: SL-C7C6
  Similarity: 180
- Face45: SL-C7C6
  Similarity: 180
- Face46: SL-C7C6
  Similarity: 180
- Face47: SL-C7C6
  Similarity: 180
- Face48: SL-C7C6
  Similarity: 180
- Face49: SL-C7C6
  Similarity: 180
- Face50: SL-C7C6
  Similarity: 180
- Face51: SL-C7C6
  Similarity: 180
- Face52: SL-C7C6
  Similarity: 180
- Face53: SL-C7C6
  Similarity: 180
- Face54: SL-C7C6
  Similarity: 180
- Face55: SL-C7C6
  Similarity: 180
- Face56: SL-C7C6
  Similarity: 180
- Face57: SL-C7C6
  Similarity: 180
- Face58: SL-C7C6
  Similarity: 180
- Face59: SL-C7C6
  Similarity: 180
- Face60: SL-C7C6
  Similarity: 180
- Face61: SL-C7C6
  Similarity: 180
- Face62: SL-C7C6
  Similarity: 180
- Face63: SL-C7C6
  Similarity: 180
- Face64: SL-C7C6
  Similarity: 180

CSD result using 64 bins
Edge Histogram Descriptor

- The image array is divided into 4x4 subimages.
- The edges in each image-block is categorized into one of the following six types: vertical, horizontal, 45° diagonal, 135° diagonal, nondirectional edge and no-edge.
- A 5-bin edge histogram of each subimage can be obtained.
- Each bin value is normalized by the total number of image-blocks in the subimage.
- The normalized bin values are non-linearly quantized.
**Fig. 1. Definition of sub-image and image-block.**
Fig. 2. Five types of edges.

- a) vertical edge
- b) horizontal edge
- c) 45-degree edge
- d) 135-degree edge
- e) non-directional edge
Fig. 3. Five types of edge bins for each sub-image.
Fig. 4. 1-D array of 80 bins of EHD.
Fig. 6. Filter coefficients for edge detection.
Result: EHD using 80 bins
My idea is...