

CREU 2016-2017 Final Report:
Towards an Ear Classification Scheme for Improved Biometric Recognition

CREU Dr. Gloria Washington, Howard University
CREU Morgan Williams, Howard University
CREU Errol Grannum, Howard University

I) **Goals and Purpose**

Ears have been used to identify people since the early 1900s. The first example of ears being used to recognize humans was when Iannarelli, an American police officer, used ear measurements to keep track of prisoners. His work led to the creation of the discipline called biometrics. Biometric systems can be used to confirm an individual's identity by comparing a sample physical feature to a previously examined template. The process of developing a biometric system consists of presenting and analyzing a physical feature, extracting the most distinguishing parts of the presented feature, then lastly creating and storing a template of the feature. In our project we focused solely on collecting biometrics relating to the structures that comprise the ear.

Ear biometrics is the measurement and analysis of the physical attributes of the ear to identify an individual. In order to improve ear recognition and identification in ear biometrics, the research conducted intends to determine the efficiency of an ear classification scheme in improving the biometric recognition of ears. Questions that we aimed to address in our study were:

1. Can ears be categorized to their structures such as the anti-helix, tragus, and lobule?
2. Can a Haar-based classifier be developed using OpenCV's computer vision library that is able to categorize the anti-helix, tragus, and lobule from pictures?
3. Can this classifier be used on an existing open-source ear database to categorize the ears in the database?
4. Can the rank-1 recognition rate or accuracy of existing biometric algorithms be improved by through this categorization step?

The research was conducted on the helix and lobule found on the outer structure of the ear.

II) **Related Work**

We learned that the structure of the ear is unique and permanent. In addition we discovered that the appearance of the ear does not change much over the course of a person's life. The contributions within this paper further reinforced our reasoning behind focusing on building classifiers for ear structures.

http://www.cosy.sbg.ac.at/~uhl/ear_survey.pdf

Pflug, Christoph B. "Ear Biometrics: A Survey of Detection, Feature Extraction and Recognition Methods" (2012).

From the study conducted in this paper we were able to get a concrete idea of how to



approach building an ear classifier. We also learned that there is a high demand for biometrics and improved schemes allow for better storage.

Ugbaga, Ghazali S. "Human Ear Classification Scheme Based On Lobule Chain Code and Helix" (2013).

The information within this paper introduced us to biometrics as a process: feature extraction, existing recognition systems, examples of algorithms used to detect edges, reduce noise and normalize image samples.

Washington, Gloria "Ear Feature Analysis, Extraction Tools, and Classification for Improved Recognition." (2013).

Key concepts we took from this paper were parametric vs non-parametric statistics and models and how bioinformatics can be taken from a person's effect on the environment.

<http://www.csse.monash.edu.au/~smarkham/resources/param.htm>

Ashok Veeraraghavan, Amit Roy Chowdhury and Rama Chellappa. "Matching Shape Sequences in Video with an application to Human Movement Analysis."

The following are references to datasets containing ear samples from the University of Notre Dame.

ND-Collection E

Data Type: Visible Light Ear

Approximate Download Size: 487 MB

[License Agreement](#)

464 visible light profile (ear) images from 114 human subjects captured in 2002.

ND-Collection F

Data Type: 3D + 2D Ear Images

Approximate Download Size: 2.5 GB

[License Agreement](#)

942 3D (and corresponding 2D) profile (ear) images from 302 human subjects captured in 2003 and 2004.

III) Process

The implementation of ear biometrics in a software application began with the installation of the Open Source Computer Vision Software (opencv), an application with a python interface designed for computational efficiency in real-time. The application was installed on a Mac OS prior to the installation of Python.

In order to create the classification scheme, the helix and lobule were defined. The helix was defined as a curved structure of cartilage on the visible ear. The helix was divided into two categories, a wide helix and a narrow helix. The lobule was defined as the small lobe of the ear. The lobule was divided into two categories, an attached lobule and a detached lobule. Positive and negative haar cascade classifier training images were collected for the helix and the lobule. 25 positive haar cascade classifier training images and 25 negative haar cascade training images were collected for the attached lobule and

detached lobule.

A visual definition of the outer structure of the ear reduces accuracy biometric recognition in contrast with a mathematical definition of the outer structure of the ear. In order to generate a mathematical definition of the outer structure of the ear, an edge detection algorithm was implemented in an object marker. The object marker adjusted the positive haar cascade classifier training images to display the structures of the ear that contained the helix and the lobule. The object marker produced a description file that specified coordinates in correspondence to the location of the helix and the lobule. A description file was then generated for the negative haar cascade classifier training images with images displaying structures of the ear that did not contain the helix or the lobule.

After creating description files for the helix and lobule, implementation of the haar cascade classifier training was conducted. The implementation consisted of the development of a detection program in python. When executed, the program detected the location of the structure of the ear based on the classifier and positive haar cascade classifier training image specified in the program. The results of the program execution determined the accuracy of the haar cascade classifier training by defining accurate and inaccurate location detection of the specified structure of the ear. The results were evaluated to generate an accuracy report and conclude a portion of the research.

IV) Results and Discussion

We were able to build a haar-cascade classifier for the the helix portion of the ear. In addition we wrote a python script to test how well the classifier detects narrow helixes. Ideally the program will highlight only areas where there is a narrow helix, but the accuracy of the classifier was sub-par to our expectations. The following are the results from conducting 5 trials.

Narrow helix accuracy reports:

trial 1: 1/4

trial 2: 3/7

trial 3: 1/4

trial 4: 1/5

trial 5: 2/5

Overall Accuracy: $8/25 = 32\%$

The script is below (figure 1).

```
import cv2
imagePath = "/test_samples/04558d168.jpg"
cascPath = "/classifier/cascade.xml"

image = cv2.imread(imagePath)
nHelixCascade = cv2.CascadeClassifier(cascPath)
g_scale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect narrow helixes in the image, return a list of rectangles of the detected helix
nHelix = nHelixCascade.detectMultiScale(
    ..g_scale,
    ..scaleFactor=1.1,
    ..minNeighbors=5,
    ..minSize=(30, 30),
    ..flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)

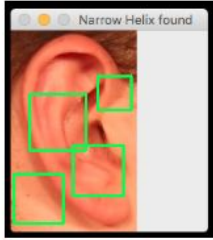
print "Found {0} narrow helixes".format(len(nHelix))
for (x,y,w,h) in nHelix:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow("Narrow Helix found", image)
cv2.waitKey(0)
```

Figure 1. Source code for detect.py

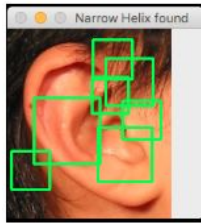
Screenshots of the trials are below (Figure 2)



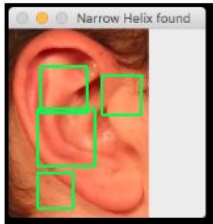
Trial 1 - 4 Narrow Helixes Detected



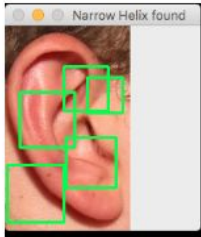
Trial 2 - 7 Narrow Helixes Detected



Trial 3 - 4 Narrow Helixes Detected



Trial 4 - 5 Narrow Helixes Detected



Trial 5 - 5 Narrow Helixes Detected

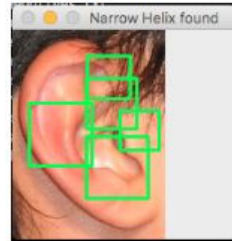


Figure 2. Trials for detecting narrow helixes

The implementation of the haar cascade classifier training for the lobule was conducted in a windows environment as a result of a consistently produced error in Mac OS when generating description files for positive and negative images. The positive and negative haar cascade classifier training images for the lobule were adjusted to black and white in order to improve accuracy. A description file was created for haar cascade classifier training and resulted in a collection of coordinates for the lobule structure of the ear (Figure 3).

```

Stage training time: 0.11
Number of used features: 1
Parent node: 5
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
-----
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
-----
0--1--2--3--4--5--6

Parent node: 6
*** 1 Cluster ***
pop: 10 20 0.100000
NFI: 10 0.000000
BACKGROUND PROCESSING TIME: 0.20
Practical time times: 0.20
-----
| N | NCOMP | ST_THR | NR | FA | EXP_ERR |
-----
| 1 | 11000 | -1.00000 | 1.00000 | 0.00000 | 0.00000 |
-----
Stage training time: 0.16
Number of used features: 1
Parent node: 6
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
-----
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
-----
0--1--2--3--4--5--6--7

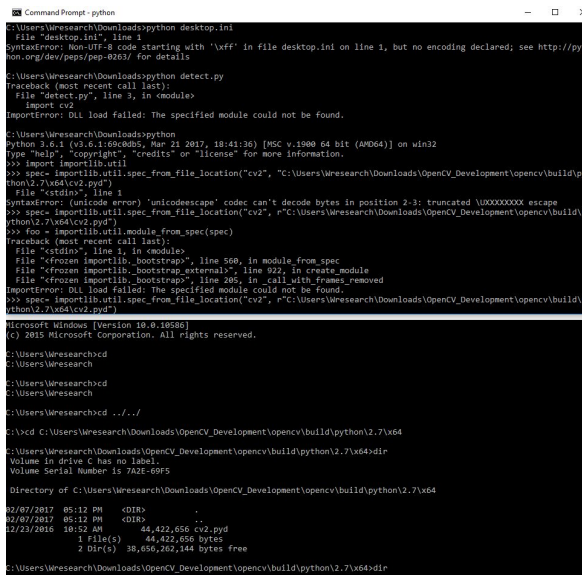
Parent node: 7
*** 1 Cluster ***
pop: 10 20 0.100000
NFI: 10 0.000000
BACKGROUND PROCESSING TIME: 0.45
Practical time times: 0.23
-----
| N | NCOMP | ST_THR | NR | FA | EXP_ERR |
-----
| 1 | 11000 | -1.00000 | 1.00000 | 0.00000 | 0.00000 |
-----
Stage training time: 0.13
Number of used features: 1
Parent node: 7
Chosen number of splits: 0
Total number of splits: 0
Tree Classifier
Stage
-----
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-----
0--1--2--3--4--5--6--7--8

Parent node: 8
*** 1 Cluster ***

```

Figure 3. Haar Cascade Classifier Training: Description File Creation

After the creation of a description file, implementation was attempted in order to define the accuracy of the haar cascade classifier. The implementation was done with a detection program written in python for a Mac OS and adjusted to run in a Windows environment. When attempting to run the program an error was produced and remained unresolved. The error indicated a module was undetected therefore we weren't able to test the detection accuracy of the lobule classifier. (Figure 4).



```

C:\Users\Wresearch\Downloads>python desktop.ini
File "desktop.ini", line 1
SyntaxError: Non-UTF-8 code starting with '\xff' in file desktop.ini on line 1, but no encoding declared; see http://pyth
on.org/dev/peps/pep-0263/ for details

C:\Users\Wresearch\Downloads>python detect.py
Traceback (most recent call last):
  File "detect.py", line 2, in <module>
    import cv2
ImportError: DLL load failed: The specified module could not be found.

C:\Users\Wresearch\Downloads>python
Python 3.6.1 (v2.6.1:02c0095, Mar 21 2017, 18:41:30) [MSC v.1900 64 bit (AMD64)] on win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>> import importlib.util
>>> spec = importlib.util.spec_from_file_location("cv2", "C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64\cv2.pyd")
>>> foo = importlib.util.module_from_spec(spec)
>>> foo.__stdin__ = line 1
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \U00000000 escape
>>> spec = importlib.util.spec_from_file_location("cv2", r"C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64\cv2.pyd")
>>> foo = importlib.util.module_from_spec(spec)
Traceback (most recent call last):
  File "stdin", line 1, in <module>
  File "frozen importlib._bootstrap", line 569, in module_from_spec
  File "frozen importlib._bootstrap_external", line 822, in create_module
  File "frozen importlib._bootstrap", line 205, in _call_with_frames_removed
ImportError: DLL load failed: The specified module could not be found.
>>> spec = importlib.util.spec_from_file_location("cv2", r"C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64\cv2.pyd")

Microsoft Windows [Version 10.0.18886]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Wresearch>cd
C:\Users\Wresearch>cd
C:\Users\Wresearch>cd
C:\Users\Wresearch>cd ..\..\f
C:\>cd C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64
C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64>dir
Volume in drive C has no label.
Volume Serial Number is 7A2E-69F5

Directory of C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64

07/07/2017  05:12 PM    <DIR>          .
07/07/2017  05:12 PM    <DIR>          ..
11/23/2016  10:52 AM             44,422,656 cv2.pyd
               1 File(s)             44,422,656 bytes
               2 Dir(s)          38,656,262,144 bytes free

C:\Users\Wresearch\Downloads\OpenCV_Development\opencv\build\python2.7\x64>

```

Figure 4. Haar Cascade Classifier Training: Detection Program Execution

V) Future Work

In the future, we plan to improve the existing accuracy of our classifiers as well as create additional classifiers for other structures of the ear including the tragus and anti-helix. In order to improve accuracy we would collect a larger amount of haar cascade classifier training images and generate updated accuracy reports based on the larger collection of positive and negative images. Questions we aim to address in the future:

1. Can the accuracy report results improved with an increase in haar cascade classifier training images?
2. How many subcategories do the outer structures of the ears need to be divided into in order to maintain accuracy of classifiers?
3. Which of the outer structures of the ear produce the highest accuracy in terms of identification and detection of structure location?

VI) Web Links

Errol Grannum: <http://hucreu2016.blogspot.com/>

Morgan Williams: <http://creuhu.blogspot.com/>

VII) **Presentations and Publications**

Howard University Research Group Presentation Engineering Conference
Washington, DC Fall 2016