Paper ID #22043

MAKER: Facial Feature Detection Library for Teaching Algorithm Basics in Python

Mr. Mehmet Ucar

M.S. in Computer Engineering, University of Houston- Clear Lake(2016) B.S. in Electrical and Electronics Engineering, Erciyes University (2008)

Dr. Sheng-Jen ”Tony” Hsieh, Texas A&M University

Dr. Sheng-Jen (”Tony”) Hsieh is a Professor in the Dwight Look College of Engineering at Texas A&M University. He holds a joint appointment with the Department of Engineering Technology and the Department of Mechanical Engineering. His research interests include engineering education, cognitive task analysis, automation, robotics and control, intelligent manufacturing system design, and micro/nano manufacturing. He is also the Director of the Rockwell Automation laboratory at Texas A&M University, a state-of-the-art facility for education and research in the areas of automation, control, and automated system integration.

MAKER: Face Detection Library to Teach Algorithm Basics in Python

Abstract

This paper describes an approach to teach face detection algorithms to beginner level programming learners using a face detection tool built in Python. Learners are expected to understand and practice their Python coding skills and algorithm knowledge in a facial feature detection and image processing application. This project is a Science Technology Engineering and Math (STEM) extensive module and includes research analysis and activity components. Existing work focuses on teaching algorithms to students using interactive tools and games [3, 4, 7]. This work lets students learn face detection algorithms using a library. Furthermore, students learn how to use the Dlib, Yattag, Cv, Numpy, Tkinter classes in Python along with the algorithmic details. This application enriches school curriculum by adding machine/software interactions and digital image processing. After completing the module activities, students are evaluated on basic understanding of face detection applications, computer vision, and the ability to complete the programming tasks. This classroom model has been implemented in a high school computer science class to provide an interesting way to help students learn programming and to motivate students to learn more about computer science.

Keywords

Face detection algorithm, Computer Vision, Image processing, Python

Background

For high school teachers, it is challenging to teach first-time programming students the concepts of programming and algorithm. Therefore it may cause difficulty to understand the course concepts for some students. Such problem causes students to have high failure rates and failure to complete the programming classes successfully[5].

An algorithm is a procedure for solving a computational problem with specified actions in computer. It provides a solution to a given problem. Understanding the algorithm basics is a key to boot up the programming experience for programming class takers. Students practice their problem solving skills and design a step by step procedures to solve a problem.

The initial purpose of introduction level computer programming classes is to teach students the basics of algorithm. Students are usually expected to practice and learn the intended programming language on their own throughout the course. In addition, algorithm courses are usually considered as the core courses for the Computer Science undergraduate. So, it is very important that students understand the algorithm concepts in high school level CS classes.

Learning a programming language is similar to learning a new spoken language, so students should have interest learning it, and be motivated to practice. High school Career and Technology education (CTE) and computer science classes should have extra hands on activities to gather more student attention. The best way to learn programming basics is to practice.

In high school setting, most school curriculum has enough coding implementation . However, students may have some difficulty to understand the practices if they have lack of algorithmic and procedural understanding of computer science. I decided to add an extra hands on learning tool to my CS courses to show students to strength and power of computer science.

Motivation

Computer science has become a more demanding area in today's world. High schools offer more CS classes to students. However, students like learning programming with visual tools. Some visual programming learning tools are Karel the Dog and MIT Scratch. It would be a great idea to teach students the face recognition basics as a visual learning tool.

During this project, students have a hands-on classroom activities on a face detection implementation and experience how the face detector tool works. Students learn the programming logic behind the face recognition tool. In addition students understand the use of digital image processing in the digital age.

Digital image processing is the use of computer algorithms to perform image processing on digital images. In digital image processing, we are able to use very wide variety of algorithms. Dlib is a modern machine learning toolkit which can be imported to Python. Our learning tool uses Dlib frontal recognition algorithm to analyze given image and detect faces.

During this project, students learn the use of image processing and face recognition tools. This project allows students to practice their understanding of face detection algorithms and learn the basics of image processing usage.

Advantages of Using this Library to Teach Algorithm to students

We use a wide variety of digital image applications such as digital cameras, cell phones, and security cameras. So, students have a great attention on image tools such as Instagram and Facebook. We propose to use a classroom activity set to teach programming and algorithm concepts of frontal face recognition in a high school class setting. This learning library tool is designed to detect human faces and save cropped images into a local directory using Python programming language. This learning tool is built as a simple image processing library tool. This classroom activity set is also used as a teaching tool which uses image detection and allowed beginning level CS students to understand the algorithm concepts in frontal face detection.

Methodology

This project is implemented in a high school AP Computer Science class to teach algorithm using the face detection algorithms. This project covers four activities.

Each activity takes about two hours of class-time. The learning module has following four classroom activities,

1. Fundamental knowledge of image processing - In this activity students are asked to read the activity materials and understand the term *image processing*. They are expected to

learn image processing operations such as blurring, zooming, edge detection, face recognition, and cropping.

2. Face detection in image processing

3. DLIB frontal face recognition

4. Testing the Face Recognition Library - Students use the library tool to detect faces and facial landmarks on each face. Python source code is given to students. They are required to successfully compile the code.

Activity 1: Fundamentals of Image Processing

Students are expected to gain basic knowledge of image processing and computer vision in this activity.

Digital image processing (DIP) is processing digital images using a computer or computer system. It consists of signals and systems as related to images. The system input should be a digital image. System will use a compatible algorithm to generate an output image. Image editing tools such as Photoshop are examples of digital image processing.

The steps in image processing are as follows,

(a) We have analog 3D surroundings around us. A camera is used to capture the surroundings.
(b) Analog input is converted into digital form.
(c) The digital image file is used as digital image processing input.
(d) A computer processes the input image as a two dimensional array.
(e) A Digital Image Processing unit uses algorithm(s) to process the image.
(f) A processed image is generated.

There are many methods and procedures for digital image processing. Some common image processing operations include blurring, zooming, edge detection, face recognition, cropping, and mirroring.

Digital image processing starts with how all humans visualize the world. We can categorize image processing in two parts, analog image processing and digital image processing.

Analog Image Processing. Analog image processing deals with analog signals in a varying electrical signal. It takes place in a two dimensional analog signal. Traditional television image is an example of an analog image processing.

Digital Image Processing. It is based on building a computational system to analyze and process digital images. Digital image processing has more advantages over analog image processing, so analog image processing has very limited usage in today’s world.

What is an Image?

An image is a combination of a bunch of two dimensional data. We can call this data mathematically f(x,y) where x stands for horizontal data and y for vertical data.

A pixel location can be located by this mathematical value in f(x,y) form.

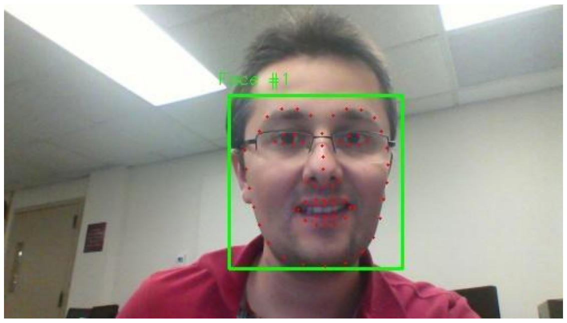


Figure 1: A sample digital image

This digital image is a combination of a data from a two dimensional array of number from 0 to 255. Then this image is an example of what we see in computer screen.

Table 1: Sample Digital Image in Two-Dimensional Array Form

128	30	123
232	123	321
123	77	89
80	255	255

Students are expected to answer the following questions after completing this module,

- What is Image Processing?
- Give some practical examples of image processing and computer vision in everyday life.
- What is the main difference between analog image processing and digital image processing?
- What is an image? How do we get digital images?
- How does the human eye recognize so many things?
- How does the brain interpret images?
- What is a pixel?

Activity 2: Face detection in image processing

Students learn the different type of face detection algorithms including
- Model based face tracking
- Weak classifier cascades
- Histogram of Oriented Gradients (HOGs) and Deep learning

The purpose of this activity is to teach students different ways to detect faces from images. Our face detection learning tool uses Dlib, which is a HOG-based (Histogram of Oriented Gradients) face detector.

Students are expected to answer the following questions after this module,

- List common face detection applications in the real world.
- What are the advantages of HOG based face detection tools?

Deep Learning (using multi-layered Neural Networks), especially for face recognition more than for face finding, and HOGs (Histogram of Oriented Gradients) are the current state of the art (2017) for a complete facial recognition process. [9].

Activity 3: Dlib and Frontal Face Recognition Algorithm

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.

The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset. [1] [6]. The landmarks are used as face part predictors.

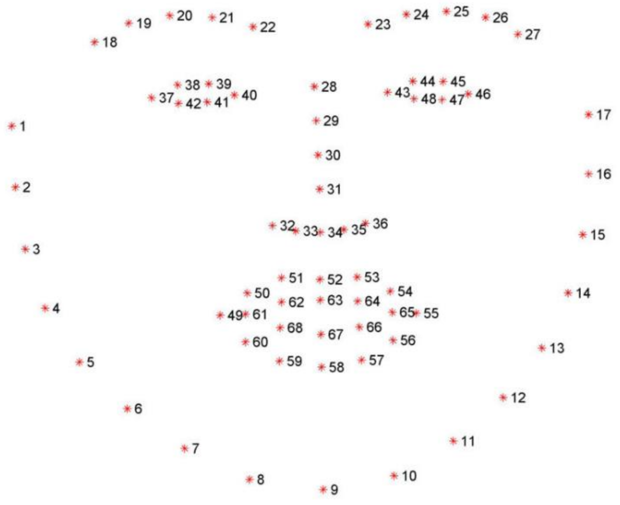


Figure 2. Visualizing each of the 68 facial coordinate points from the iBUG 300-W dataset [6].

- The mouth can be accessed through points [48, 68].
- The right eyebrow through points [17, 22].
- The left eyebrow through points [22, 27].
- The right eye using [36, 42].
- The left eye with [42, 48].
- The nose using [27, 35].
- And the jaw via [0, 17].

All these data are already in the FACIAL_LANDMARKS_IDXS dictionary inside face_utils of the imutils library.

dlib.get_frontal_face_detector() is Dlib's HOG-based (Histogram of Oriented Gradients) face detector and loading the facial landmark predictor.

Activity 4. Testing the Face Recognition Library

Project source files are given to students and students are asked to compile, run and test the frontal face recognition tool on their own.

This face detection library includes following Python libraries,

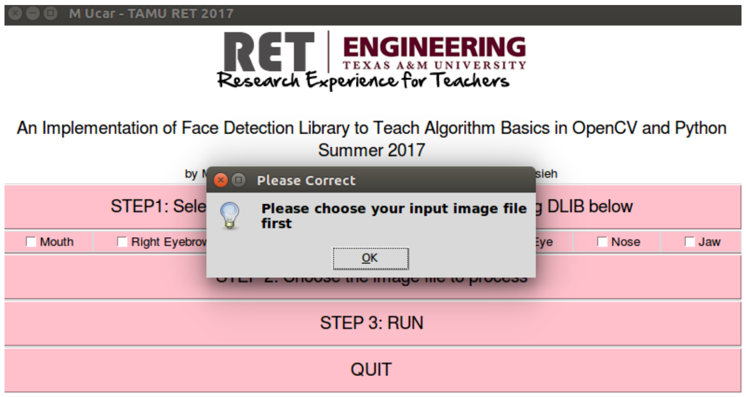
- Tkinter is used as a GUI Builder
- Os used to use file system and generate files
- Yattag used to generate html pages
- Dlib used as a face detector
- OpenCV , imutils,and numpy are used to process the images.

Step 1: Choose the parts of the faces to be extracted using DLIB face Detector

Chose what parts of the face they want to detect and extract using Dlib.forntal_face_detector. Options include mouth, eyes, eyebrows, nose and jaw

Step2: Choose the image file to process

If no image file is selected, the output will give a message box error as follows,


M Ucar - TAMU RET 2017

RET | ENGINEERING TEXAS A&M UNIVERSITY Research Experience for Teachers

An Implementation of Face Detection Library to Teach Algorithm Basics in OpenCV and Python
Summer 2017

by M sieh

Please Correct

Please choose your input image file first

OK

STEP1: Sele g DLIB below

Mouth | Right Eyebrow | Eye | Nose | Jaw

STEP 3: RUN

QUIT

Figure 3: Error message if no image input is selected

Once the user clicks on the STEP 2 button, we will see the image browse screen.

Step 3: Run program

Once clicked, the code generates the following screen and gives a link to see the generated html Report file

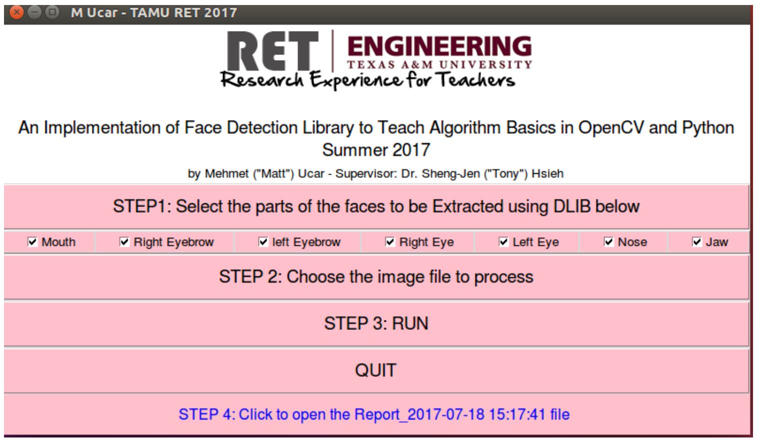

M Ucar - TAMU RET 2017
RET
ENGINEERING
TEXAS A&M UNIVERSITY
Research Experience for Teachers
An Implementation of Face Detection Library to Teach Algorithm Basics in OpenCV and Python
Summer 2017
by Mehmet ("Matt") Ucar - Supervisor: Dr. Sheng-Jen ("Tony") Hsieh
STEP1: Select the parts of the faces to be Extracted using DLIB below
Mouth
Right Eyebrow
left Eyebrow
Right Eye
Left Eye
Nose
Jaw
STEP 2: Choose the image file to process
STEP 3: RUN
QUIT
STEP 4: Click to open the Report_2017-07-18 15:17:41 file

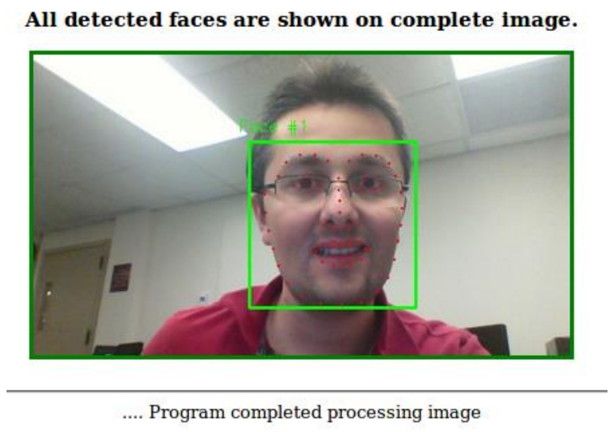
Figure 4: Screen when the tool completes processing the image.

Once this screen is visible, command output gives all the details to user. Following is an example output of the command output:

```
DLIB Frontal face recognition algorithm is selected.
....    Processing /home/user/Desktop/project/Faceparts/sample_5.jpg    ....
Number of faces detected in the image: 1
STARTED processing FACE #1
  1_MOUTH                 Detected, circled and saved.
  1_MOUTH                 Cropped and saved.
  1_RIGHT_EYEBROW         Detected, circled and saved.
  1_RIGHT_EYEBROW         Cropped and saved.
  1_LEFT_EYEBROW          Detected, circled and saved.
  1_LEFT_EYEBROW          Cropped and saved.
  1_RIGHT_EYE             Detected, circled and saved.
  1_RIGHT_EYE             Cropped and saved.
  1_LEFT_EYE              Detected, circled and saved.
  1_LEFT_EYE              Cropped and saved.
  1_NOSE                  Detected, circled and saved.
  1_NOSE                  Cropped and saved.
  1_JAW                   Detected, circled and saved.
  1_JAW                   Cropped and saved.
Face #1 is cropped and saved.
COMPLETED processing FACE #1.
All detected faces are shown on complete image.
```

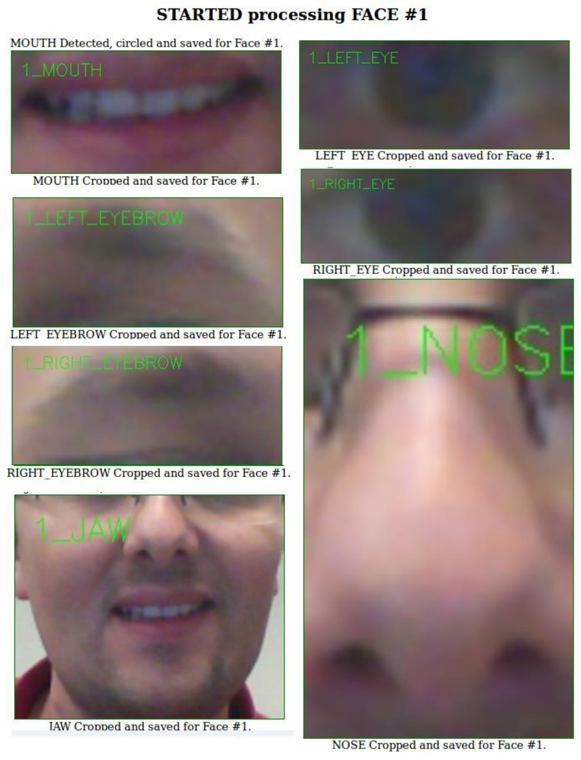
Step 4: Click to Open the Report file

Once the report file is generated with Yattag module in Python programming language. We will be able to see the following output:


All detected faces are shown on complete image.

.... Program completed processing image

Figure 5: All face parts are found and a rectangle is drawn around the face.


STARTED processing FACE #1

MOUTH Detected, circled and saved for Face #1.

MOUTH Cropped and saved for Face #1.

LEFT EYE Cropped and saved for Face #1.

RIGHT_EYE Cropped and saved for Face #1.

LEFT EYEBROW Cropped and saved for Face #1.

RIGHT_EYEBROW Cropped and saved for Face #1.

JAW Cropped and saved for Face #1.

NOSE Cropped and saved for Face #1.

Figure 6: Cropped face parts from the HTML report file.

This activity allows students to complete development of an image processing tool after gaining theoretical knowledge. This teaching method is expected to broaden students' knowledge and help encourage them to become computer scientists.

Implementation

The activities described above were implemented in an AP Computer Science class consisting of students in grades 10-12. Students were asked to try at least 10 images and test how many of those images were completely processed and all faces/face parts are detected. Students also recorded their findings in spreadsheet. Students noted that:

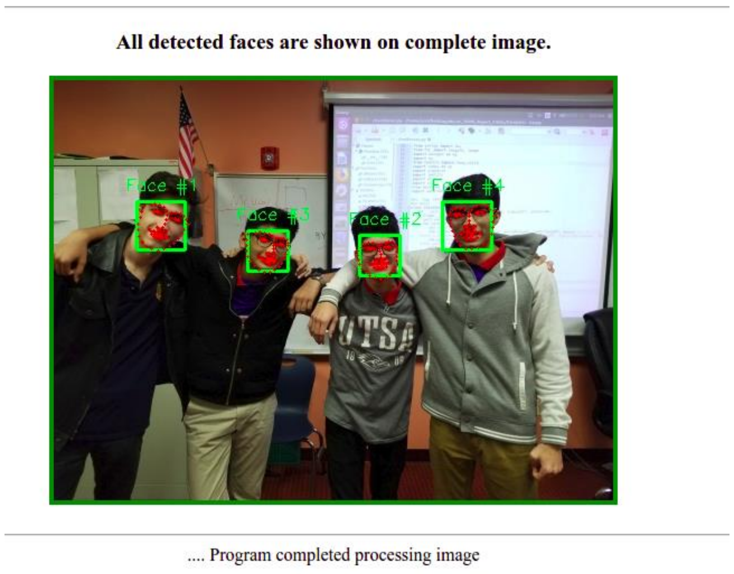
- The better the image quality, the better the library works.
- Light should be equally distributed over the face.
- The library doesn't work if images are very dark or very bright.
- If hair covers the eyes, the library will not be able to detect some parts of the image.
- The library may not work if the image is upside down.

Software needs

- Python programming tools
- Dlib machine learning libraries (for frontal face recognition algorithm)
- 68 point face predictor data file (for facial parts recognition)
- OpenCV for image processing purposes
- Notepad++ software for code editing
- Microsoft Excel for data analysis

Conclusion and Future Directions

Lesson module and learning activities are implemented in an AP Computer Science class in high school setting. Students' knowledge of image processing was assessed using a pre-test before implementing the modules. The average student score on the pre-test was 42%.



All detected faces are shown on complete image.

.... Program completed processing image

Figure 7. Programming students testing face detection tool in CS Class.

Students completed all activities and practiced using the facial detection tool in Python. Students knowledge was assessed again with a post-test. The average post-test score was 88%, suggesting that this learning module clearly caught the attention of the students.

As a future direction, this module will be implemented in more computer science classes. In addition, more image processing tools, such as motion tracking and human-computer interaction, will be added to the module.

Acknowledgement

This work was supported by a grant from the National Science Foundation’s Research Experience for Teachers (RET) Program (Award No. 1300779). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] http://docs.opencv.org/trunk/d9/df8/tutorial_root.html

[2] YeeHui Oh, ChengYew Tan, Vishnu Monn Baskaran, "Active participant identification and tracking using depth sensing technology for video conferencing", *2013 IEEE Conference on Open Systems (ICOS)*, pp. 7-12, 2013.

[3] Tussanai Parthornratt, Natchaphon Burapanonte, Wisarute Gunjarueg, "People identification and counting system using raspberry Pi (AU-PiCC: Raspberry Pi customer counter)", *2016 International Conference on Electronics Information and Communications (ICEIC)*, pp. 1-5, 2016.

[4] Min Zuo, Guangping Zeng and Xuyan Tu, "Research and improvement of face detection algorithm based on the OpenCV," *The 2nd International Conference on Information Science and Engineering, Hangzhou, China, 2010*, pp. 1413-1416.doi: 10.1109/ICISE.2010.5691414

[5] Begosso, Luiz Carlos, et al. "An approach for teaching algorithms and computer programming using Greenfoot and Python." *Frontiers in Education Conference (FIE), 2012*. IEEE, 2012.

[6] https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

[7] Al-Bow, M., Austin, D., Edgington, J., Fajardo, R., Fishburn, J., Lara, C., ... & Meyer, S. (2009). Using game creation for teaching computer programming to high school students and teachers. *ACM SIGCSE Bulletin*, *41*(3), 104-108.

[8] Kalelioglu, F., & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: a discussion from learners' perspective. *Informatics in Education*, *13*(1), 33.

[9] http://www.face-rec.org/databases/