John Randazzo

Andrew Rozniakowski

4/6/2016

Proposal - Facial Recognition

For this assignment, our group has decided to create facial recognition software that implements Harr cascading to identify facial features in a given picture. This application can be used to determine the presence of a face in an image as well as to identify specific facial features, such as the eyes and mouth. Through the use of trial pictures, the computer learns the basic features of a human face and how to identify these features in a given image. Similar systems are used in most cameras to identify and focus in on faces, rather than sharpening the background scenery. A more advanced version of this software has a myriad of possible applications. For instance, it can be highly beneficial in the application and final outcome of CGI (Computer-generated imagery) characters in movies, television shows and video games since the facial recognition software would be able to identify and reduplicate individual and unique facial features that can then be converted into CGI. It also has multiple military uses, such as in security and weaponry technologies.

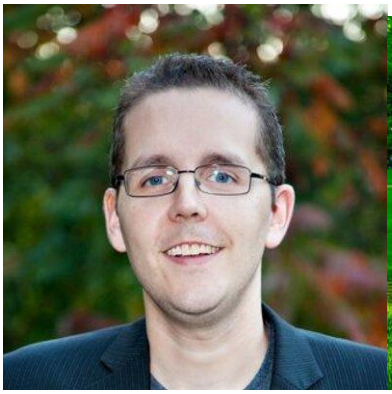
Likely users of this application include, but are not limited to, companies that install cameras in their products, such as cell phone and computer manufacturers. The entertainment industry can implement this software in a variety of ways, from movie production to video game character development. The military is also a highly plausible user of this facial recognition software. While our project is a simple model of this application, there are numerous examples of advanced facial recognition systems currently available. For example, these advanced systems can be used for security purposes, such that a person can unlock their doors by simply having a camera scan their face and the facial recognition software reads their facial features. This is just one of many other instances. This application could even be able to identify faces and people from far distances for surveillance or identification purposes. The military could also use it to determine how many people are in an area before they attack or fire a weapon, which could protect the safety of civilians and our own soldiers. Most people use of this software everyday in their phones and laptops; most cell phones and computers have a similar system in the camera application that allows it to focus on people’s faces rather than on other objects. Most computers also come equipped with applications that can adjust digital pictures to fix issues, such as red eye and brightness. These programs must be able to identify, not only facial features, but also any object to fix these issues.

To the average person, our system solves a very simple problem: taking better pictures and adjusting these pictures to a person’s liking with the least amount of effort. However, industries, such as the entertainment industry, it allows for capturing the most detailed visual effects and images anyone has seen to date and bringing the into the homes of their customers, whether it be through video games, movies or more.

We will use OpenCV's facial recognition software through Haar cascade training to complete this project. We will have to use two different datasets to acquire our expected results. The first is a set of Haar cascade xml files, which will be implemented to teach our program how to identify basis facial features in juxtaposition to other objects. There are specific Haar files that can teach the computer different features, such as an eye, a mouth, a full body or even other objects, like a car. Specific files are also needed to teach variations of these features, like a smile or eyes with glasses. Some of the Haar cascade files we will be using for our project can be found at: https://github.com/Itseez/opencv/tree/master/data/haarcascades.

Using this technique, our computer will then be able to identify the features specified in the Haar files by looking through positive and negative pictures–positive pictures representing things that we want the computer to recognize (faces, eyes, nose) and while negative pictures representing random objects. This will allow the computer to learn how compare the images with its knowledge of faces and non-faces and recognize the differences. Thus, our second dataset is a set of positive and negative pictures we will build through Google images. Examples of positive and negative pictures are below:

Positive: Negative:



Our project begins by obtaining our datasets. First, we organized the Haar cascade files we would like to use for our first dataset. We used multiple websites (including the one given above) to find the various Haar files we wanted to use. Currently, we have over 20 files that we can use to teach the computer to recognize the facial features we want it to. Next, we compiled our dataset of images by searching for various non-facial objects through Google images. We will use a dataset of 40 positive pictures and 100 negative pictures. Since we are personally picking the pictures, we do not have to worry about cleaning the dataset for outliers. After obtaining our complete set of images, we will normalize them by having all positive pictures equal one and all negative pictures equal 0.

After manually gathering 40 positive and 100 negative pictures, we must train the system to know what it is looking for. In order to do this, we will create what is called a *classifier* using a specific Harr file. We will accomplish this by using Python's CascadeClassifier command with our specific Harr cascade xml file. We will then train this classifier by giving it a set of positive and negative pictures using Python's command train cascade, which takes in images as an input. Once a classifier is trained, it can then be applied to a bigger region of interest. For example, we

can train a classifier to locate eyes, then apply that classifier to other images. When applying it to other images for testing, the classifier uses a window that moves across the image searching for the feature. The classifier returns a 1 if the object is likely in the classifiers window or a 0 if its not. The classifiers window can easily be resized in order to find objects in different sized images. Finally, we would test our application. To do this, we will use a series of 10 different images. Seven of the 10 pictures will be positive pictures, in which we expect our program to identify faces and facial features in some sense (scoring a 1). These images will test our program's success rate by testing its ability to identify faces and facial features in a variety of different images. For example, some pictures could have multiple people, a person smiling or crying, a person with glasses on and so forth. We also expect our program to view the other 3 pictures out of 10 trial pictures as negatives (scoring a 0).