John Randazzo

Andrew Rozinakowski

4/6/2016

**Proposal - Facial Recognition**

For this project our group has decided to create a facial recognition system that uses Harr cascading to identify faces in a given picture. This application can be used to determine when a face is present in an image as well as identify facial features such as eyes or a mouth. Through test pictures the computer learns what makes up a face and what to look for when identifying a face. A similar system is used by most cameras to identify faces in order to focus on the people rather than something in the background. A more advanced software has an abundant amount of possible applications. For instance, it can be highly important to the implementation of CGI (Computer-generated imagery) characters in movies, tv shows or video games since the facial recognition software can identify facial features that can then be converted to CGI. It also has multiple military uses such as security and weaponry.

Likely users of this application include any company that uses cameras in their products, such as cell phone and laptop makers. The entertainment industry can use it in a variety of ways from movies to video games. The military is also a major potential user of a this system. While our project is a simple version of this system, an advanced facial recognition system can be used for security purposes, such having locked doors open by facial features or even fingerprints, and many more possibilities. It could identify faces and people from a far distance 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 military. Most people use of this software everyday in their phones or laptops. Every person who owns a cell phone has a similar system in their phone’s camera that allows it to focus on people rather than 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 just facial features but any object to fix these issues.

To the average person our system solves a very simple problem, which is taking better pictures and adjusting pictures to a person’s liking. However, to the entertainment industry it allows them to bring the most detailed visual effects anyone has seen to date into people’s homes whether it be through video games, movies or more.

We will use OpenCV’s facial recognition using 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 are used to teach our program how to identifying facial features and other objects. There are specific haar files to teach the computer different objects such as an eye, a mouth, a full body or even other objects like a car. Specific files are also needed for variations of these objects, like a smile or an eye 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 learn how to identify the features specified in the haar files by looking through positive and negative pictures. Positive pictures being of things we want the computer to recognize like a face or an eye, while negative pictures are of random objects so the computer can compare them with its knowledge of faces and find 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 dataset. First we found our dataset of the haar cascade files we would like to use. We used multiple websites(including the one given above) to find various kinds of haar files we would like to use. Thus, we have over 20 files that can teach the computer to recognize objects we want it to. Next we found our dataset of images by searching with google images. We will use a dataset of 40 positive pictures and 100 negative ones. Since we are personally picking the pictures we will use we do not have to worry about cleaning the dataset for outliers. After obtaining our images we will normalize them by having all positive pictures equal one, while negative ones equal 0.

After manually gathering 40 positive and 100 negative images we must train the system to know what it is looking for. In order to do this we create what is called a classifier with a specific harr file. We accomplish this by using python’s CascadeClassifier comand with our specific har cascade xml file. We then train this classifier by giving it a set of positive and negative pictures using python’s command traincascade which takes in images as an input. Once a classifier is trained it can be applied to a bigger region of interest. For example, we can train a classifier to locate eyes then apply it to other images. When applying it to other images for testing, the classifier uses a window that moves across the image searching for the object. 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 must now test our application. To do this we will use a series of 10 different images. Seven of the 10 pictures will be positive, in which we expect our program to identify faces and facial features in some sense. These images will test our program’s success rate by testing its ability to identify faces and facial features in a variety of ways. For example, some pictures could have multiple people, a person smiling or crying, a person with glasses and so on. We also expect our program to view the other 3 out of 10 pictures as negatives.