CARLETON UNVIVERSITY

Gaze Capture using webcam Eyetracking

Student: *Joshua Hills 101142996*

Course: *Computer Vision(COMP4102A)* – Professor: *Michael Genkin*
Due date: *April 12, 2022*

1. Abstract:

The goal of this project is to explore the limitations of webcam eye tracking and understand the methods that can be used to process webcam images into gaze vectors, which can then be used to map the gaze to a screen using calibration. In the Project Proposal it was explained that the goal of this project was to explore 3 different methods of webcam eye tracking: Using a CNN, using a geometric algorithm, and using a combination of the two. The focus of this report is different. This report's focus is exploring the best machine learning models for the task, not limited to CNN/Geometry.

2. Introduction

I chose this as my final project because I wanted to build my machine learning knowledge and understaning of how the machine learning algorithms work, and which task each model is best suited used for. I also find eye tracking very interesting and one of the most important computer vision applications. I am very interested in VR and AR, having eye tracking in those domains will be very influential. Understanding where a VR user is looking can change how we render games. Spending more computing power on what the user is focusing on will seriously improve fidelity. This process is called foveated rendering. This is a realtime application, so it's also important to understand the time it takes to predict a frame, faster of course being better.

3. Background

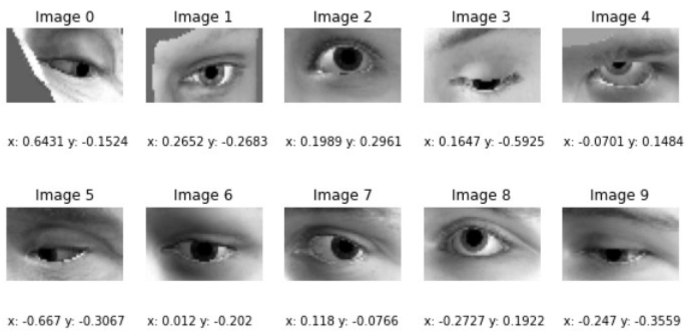
As reference I used three main papers.

- **Appearance-Based Gaze Estimation in the Wild [1]**
  This is a paper explaining the findings of the MPIIGaze Dataset, which is a dataset of real people eye tracking data. I didn't this dataset for this project keep in mind, but it did show a lot of important information that influenced my decision making. This dataset consists of 213659 images collected from 15 people during everyday laptop use. This lead to a lot of variability in the data - lighting conditions, head orientation. This made me realize that it is important to have variability in the dataset I end up using, fortunetly the EyeGaze dataset I used was able to synthesize lighting conditions and head orientation. This paper also shows that a CNN (convolutional neural network) outperforms all methods when doing cross data-set evaluations. I knew it was important to impletement a CNN.
  X. Zhang, Y. Sugano, M. Fritz and A. Bulling, "Appearance-based gaze estimation in the wild," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4511-4520, doi: 10.1109/CVPR.2015.7299081.

- **Learning-by-Synthesis for Appearance-based 3D Gaze Estimation [2]**
  This paper explains how the EyeGaze dataset was generated, and the advatages of using synthesized data over traditional methods. Basically this synthetic dataset was generated using Unity Eyes, which is a tool made for eye tracking researchers to generate synthetic eye images. To generate an image of an eye we pass metadata about the eye: x,y,z of the eye corner and the iris, look_vec which is what we use as our target variable, this is consists of two values which are the x and y vectors of the eyes, essentially where they're looking. This metadata also contains

pupil size, iris size, eye color, illumination details and head pose. All of these allow for a huge amount of variability, making sure we cover all different sorts of people and their enviroment's conditions.
Y. Sugano, Y. Matsushita and Y. Sato, "Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1821-1828, doi: 10.1109/CVPR.2014.235.

4. Methodology



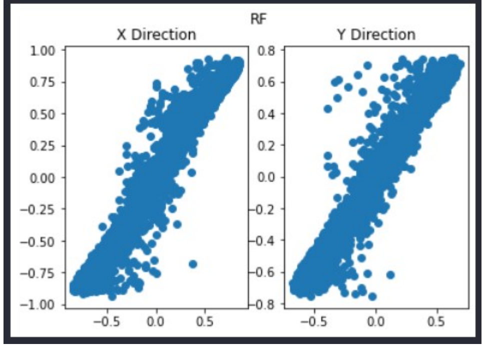
The above image shows the simulated images we are using. With x and y value to create a look vector for every eye. With this x and y value we estimate the gaze of an eye. Negative x means the eye is looking to the left, positive is right. Negative y means the eye is looking down, positive is up.

I used the EyeGaze data set which was generated using Unity Eyes which is synthetic data. My goal for this project is to test and find out the best machine learning model for the task. The first model I tested was Random Forest Regressor (RF) which fits many decision trees and then ensambles them, this model has outperformed all other models in the past according to [1], which is why I decided to include it. I also used a K-Nearest-Neighbors (KNN) which is traditionally used for classification, but it still works here if we are focused on speed. Time to predict and time to fit is important for realtime applications, so I've included a KNN in this experimentation. For these models I used a pipeline to first flatten the data and then fit it. This created two models which I stored the training time for. Lastly I used a convolutional neural network (CNN) which is a powerful machine learning algorithm that can be used for many different tasks. First the CNN takes in a greyscale image as input and then outputs a gaze vector. To compare these models I used a cross validation method, with a train test split of 80% and 20% respectively. I compute a mean squared error (MSE) for each model to find which of the 3 performs the best for this task. I will use 10000 images to train each of these three models.

5. Results

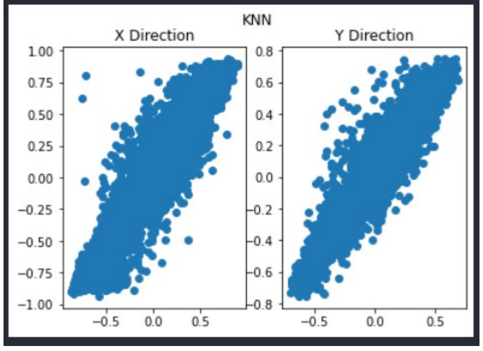
- **Random Forest Regressor**

```
RF MSE:
x     0.361002
y     0.216429
dtype: float64
```



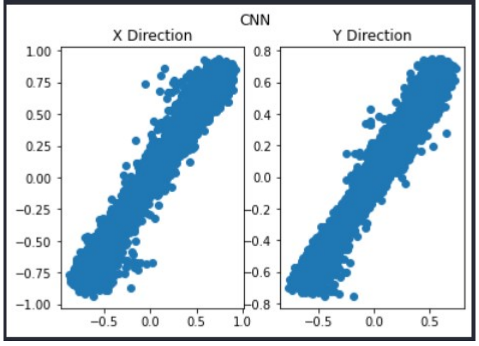
- **K-Nearest Neighbors**

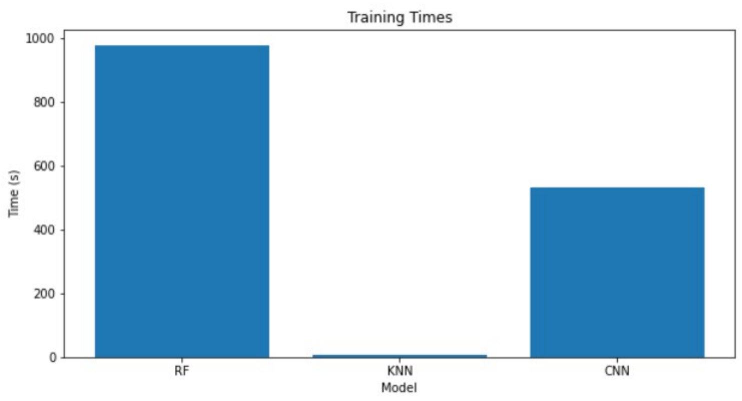
```
KNN MSE:
x     0.354601
y     0.221250
dtype: float64
```



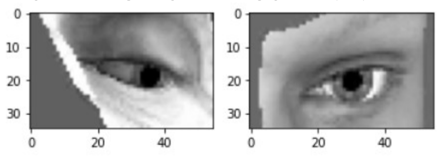
- **Convolutional Neural Network**

```
CNN MSE:
x     0.376124
y     0.225803
dtype: float64
```





So, which model is best? Taking a look at the resutls there is actually advantages for each of these models. First lets talk about Mean Squared Error (MSE), MSE is the average of the square of the errors. The larger the number the larger the error. These models are using images to predict two values, the x and y look vectors. x is looking left and right and y is looking up and down. The "X Direction" graph is comparing the test x values (on the y axis) to the predicted x values (on the x axis), so a perfect model would create a perfect line $y = x$. The "Y Direction" is the same but for y predictions. Onto the results, the best performing model is pretty close, CNN is the worst out of the three. But it's important to note that the CNN dispersity is low when looking at the graph, so in the case we want to avoid outliers we'd use the CNN. The KNN and the RF are tied for best, with super close MSEs. To break this tie lets look at the training times. The KNN took around 8 seconds to train on a dataset of 10000 images, while the RF took just over 16 minutes. Training time is less import than performance, but to break this tie I'd use KNN.



```
KNN x: -0.40906 y: 0.51164
CNN: x: -0.44555 y: 0.608541
Actual: x: -0.4494 y: 0.5905
```

```
RF x: 0.22238 y: 0.49746
CNN: x: 0.196132 y: 0.433581
Actual: x: 0.2343 y: 0.3897
```

The above image is showing some predicted vectors for 2 random eye images to show what our models are actually doing. Overall I think the results are great, machine learning does a great job at predicting gaze given just one image.

6. Discussion

Being able to predict gaze is super interesting, this type of data is certainly being used in the world of VR/AR and in Human Computer Interaction studies. There is lots of future work that could be done here, but is out of the scope for this project. One of the main things would be trying these models to see if they work as effectively on real eyes rather than just synthetic data, the usefulness of this model isn't really determined until that is tried. Another interesting thing to see is using models trained using SimGAN (Simulator-Based Generative Adversarial Networks) to combine real eye photos (like MPIIGaze) with synthetic data. This would allow researchers to create huge datasets for this type of training.