Camgaze.js: A JavaScript Library for Eye Tracking and Gaze Prediction

Alex Wallar [1] Christian Poellabauer [2] Aleksejs Sazonovs [1]
Patrick Flynn [2]

[1]University of St Andrews

[2]University of Notre Dame

March 27, 2014

Table of contents

What is it?

- Eye tracking is a problem which tries to determine where a user is looking on the screen

What is it?

- Eye tracking is a problem which tries to determine where a user is looking on the screen
- Usually done using IR or 3D cameras

What is it?

- Eye tracking is a problem which tries to determine where a user is looking on the screen
- Usually done using IR or 3D cameras
- Some web-cam technologies have emerged

What is it?

- Eye tracking is a problem which tries to determine where a user is looking on the screen
- Usually done using IR or 3D cameras
- Some web-cam technologies have emerged
- However, no in-browser solutions have been presented solely using HTML5

What is it?

- Eye tracking is a problem which tries to determine where a user is looking on the screen
- Usually done using IR or 3D cameras
- Some web-cam technologies have emerged
- However, no in-browser solutions have been presented solely using HTML5
- Until now :)

Motivation

- Eye tracking can provide vital data about what is important on the screen

Motivation

- Eye tracking can provide vital data about what is important on the screen
- We can create more intuitive user interfaces

Motivation

- Eye tracking can provide vital data about what is important on the screen
- We can create more intuitive user interfaces
- Using the web, we can crowd source where people are looking at on the website

Motivation

- Eye tracking can provide vital data about what is important on the screen
- We can create more intuitive user interfaces
- Using the web, we can crowd source where people are looking at on the website
- Also, since all of the eye tracking is done on the client side, we can preserve user privacy

Camgaze.js

- A library for eye tracking that is done inside a web browser using JavaScript

Camgaze.js

- A library for eye tracking that is done inside a web browser using JavaScript
- Uses only commodity camera (i.e. a web-cam)

Camgaze.js

- A library for eye tracking that is done inside a web browser using JavaScript
- Uses only commodity camera (i.e. a web-cam)
- Anybody can use the library without downloading any external program besides a web browser

Camgaze.js

- A library for eye tracking that is done inside a web browser using JavaScript
- Uses only commodity camera (i.e. a web-cam)
- Anybody can use the library without downloading any external program besides a web browser
- It is possible to determine where the user is looking on the screen whilst preserving user privacy and limiting server load

Overview

1. Obtain video using Web RTC (Real Time Communication) library

Overview

1. Obtain video using Web RTC (Real Time Communication) library
2. Determine the positions of the pupil centroids

Overview

1. Obtain video using Web RTC (Real Time Communication) library
2. Determine the positions of the pupil centroids
3. Use a steady point on the face as a reference point to determine gaze

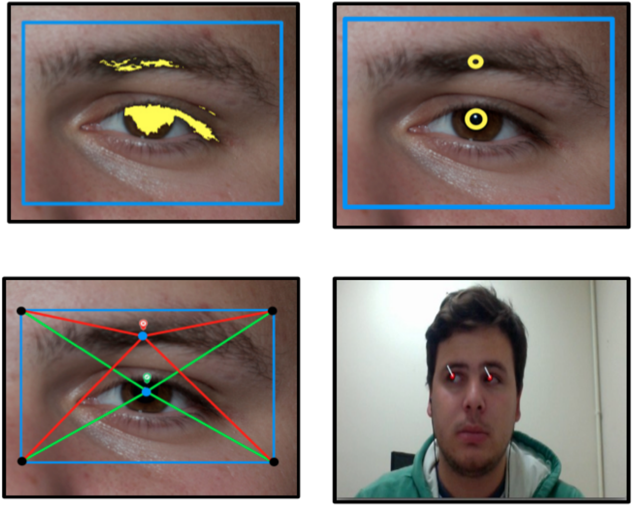
Overview

1. Obtain video using Web RTC (Real Time Communication) library
2. Determine the positions of the pupil centroids
3. Use a steady point on the face as a reference point to determine gaze
4. Add face position and head orientation vectors to the predicted gaze in order to get a result gaze vector

Overview

1. Obtain video using Web RTC (Real Time Communication) library
2. Determine the positions of the pupil centroids
3. Use a steady point on the face as a reference point to determine gaze
4. Add face position and head orientation vectors to the predicted gaze in order to get a result gaze vector
5. The gaze vector is mapped onto the screen using the gaze mapping from the calibration stage

Determining Gaze Direction



Calibration

1. Four circles appear at the corners of the screen in sequence

Calibration

1. Four circles appear at the corners of the screen in sequence
2. The user is asked to look at each circle in order for a matter of time

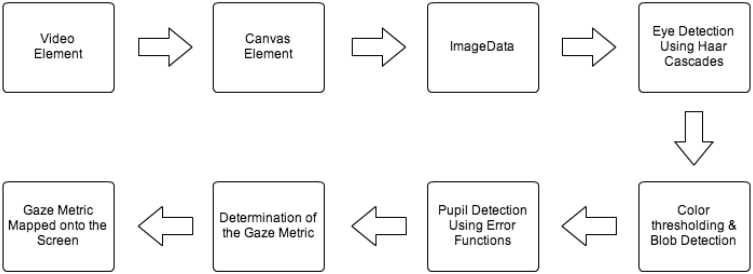
Calibration

1. Four circles appear at the corners of the screen in sequence
2. The user is asked to look at each circle in order for a matter of time
3. Data about the gaze vector is stored and averaged in order to create a mapping

Calibration

1. Four circles appear at the corners of the screen in sequence
2. The user is asked to look at each circle in order for a matter of time
3. Data about the gaze vector is stored and averaged in order to create a mapping
4. The mapping is returned such that new gaze input will correlate with a position on the screen within a window of error

Process


Video Element
Canvas Element
ImageData
Eye Detection Using Haar Cascades
Color thresholding & Blob Detection
Pupil Detection Using Error Functions
Determination of the Gaze Metric
Gaze Metric Mapped onto the Screen

Results

- We can determine the point of gaze within a $2.1in(\pm 0.1)$ radius

Results

- We can determine the point of gaze within a $2.1in (\pm 0.1)$ radius
- This means that we are within $0.6in$ of the results *Holland et al.* which used a native iPad application.

Future Work

- Use an Active Appearance Model instead of Haar Classifiers to determine the reference points and eye rectangles

Future Work

- Use an Active Appearance Model instead of Haar Classifiers to determine the reference points and eye rectangles
- Use a neural network for calibration instead of linear mapping

Future Work

- Use an Active Appearance Model instead of Haar Classifiers to determine the reference points and eye rectangles
- Use a neural network for calibration instead of linear mapping
- Undergo large scale, crowd sourced user testing

Questions