Gaze Estimation with Pupil Tracking in Real-Time

Final Presentation

Tomas Bartipan
Supervisor: Benjamin Busam

Technical University of Munich
Chair of Computer Aided Medical Procedures
Project Management and Software Development for Medical Applications

February 1, 2018

tomas.bartipan@tum.de
Recap

Figure: NIR light reflection and eye schematic

Source:
Real-Time Eye Gaze Tracking Under Natural Head Movements
https://www.ecse.rpi.edu/~cvrl/zhiwei/gazetracking/gazetracking.html
Accessed: 9/11/17
Recap hardware

Figure: Hardware in use
The steps to get there

1. Locate eye region (Haar)
2. Locate iris center (Snakuscule)
3. Locate glint (Ellipse fitting)
4. Calculate gaze from iris-glint vector and eye position (Linear regression)
The steps to get there

1. Locate eye region (Haar)
2. Locate iris center (Snakuscule)
3. Locate glint (Ellipse fitting)
4. Calculate gaze from iris-glint vector and eye position (Linear regression)
The steps to get there

1. Locate eye region (Haar)
2. Locate iris center (Snakuscule)
3. Locate glint (Ellipse fitting)
4. Calculate gaze from iris-glint vector and eye position (Linear regression)
The steps to get there

1. Locate eye region (Haar)
2. Locate iris center (Snakuscule)
3. Locate glint (Ellipse fitting)
4. Calculate gaze from iris-glint vector and eye position (Linear regression)
Snakuscule
Glint detection

1. Locate the brightest pixels in cornea region
2. Find a contour
3. Fit an ellipse
   - Get a proper light! Until then, just use a point.
Glint detection

1. Locate the brightest pixels in cornea region
2. Find a contour
3. Fit an ellipse
   - Get a proper light! Until then, just use a point.
Glint detection

1. Locate the brightest pixels in cornea region
2. Find a contour
3. Fit an ellipse
   - Get a proper light! Until then, just use a point.
Glint detection

1. Locate the brightest pixels in cornea region
2. Find a contour
3. Fit an ellipse
   - Get a proper light! Until then, just use a point.
Last step - Linear regression

1. Create a polynomial transformation function
2. Input(variables) = eye center position, glint parameters
3. Output = screen coordinates
4. In calibration phase, let user look at many (15-30) points and calculate polynomial coefficients that minimize squared error
5. In action phase, plug the actual values into the polynomial
1. Create a polynomial transformation function
2. Input(variables) = eye center position, glint parameters
3. Output = screen coordinates
4. In calibration phase, let user look at many (15-30) points and calculate polynomial coefficients that minimize squared error
5. In action phase, plug the actual values into the polynomial
Last step - Linear regression

1. Create a polynomial transformation function
2. Input(variables) = eye center position, glint parameters
3. Output = screen coordinates
4. In calibration phase, let user look at many (15-30) points and calculate polynomial coefficients that minimize squared error
5. In action phase, plug the actual values into the polynomial
Last step - Linear regression

1. Create a polynomial transformation function
2. Input(variables) = eye center position, glint parameters
3. Output = screen coordinates
4. In callibration phase, let user look at many (15-30) points and calculate polynomial coefficients that minimize squared error
5. In action phase, plug the actual values into the polynomial
Last step - Linear regression

1. Create a polynomial transformation function
2. Input(variables) = eye center position, glint parameters
3. Output = screen coordinates
4. In calibration phase, let user look at many (15-30) points and calculate polynomial coefficients that minimize squared error
5. In action phase, plug the actual values into the polynomial
Linear regression is simple, but hard to do right!

1. Needs many calibration points
2. Coefficients need to have the same order of magnitude.
3. Small noise in calibration data results in big errors
4. Better to use non-realtime methods in calibration phase
Linear regression is simple, but hard to do right!

1. Needs many calibration points
2. Coefficients need to have the same order of magnitude.
3. Small noise in calibration data results in big errors
4. Better to use non-realtime methods in calibration phase

```cpp
// TODO 2 separate polynomials for left/right?
VectorXf xPolynomial = m_parameterMatrix.jacobiSvd(ComputeThinU | ComputeThinV).solve(m_xRhs);
VectorXf yPolynomial = m_parameterMatrix.jacobiSvd(ComputeThinU | ComputeThinV).solve(m_yRhs);
```
Linear regression is simple, but hard to do right!

1. Needs many callibration points
2. Coefficients need to have the same order of magnitude.
3. Small noise in callibration data results in big errors
4. Better to use non-realtime methods in callibration phase

// TODO 2 separate polynomials for left/right?
VectorXf xPolynomial = m_parameterMatrix.jacobiSvd(ComputeThinU | ComputeThinV).solve(m_xRhs);
VectorXf yPolynomial = m_parameterMatrix.jacobiSvd(ComputeThinU | ComputeThinV).solve(m_yRhs);
Linear regression is simple, but hard to do right!

1. Needs many callibration points
2. Coefficients need to have the same order of magnitude.
3. Small noise in callibration data results in big errors
4. Better to use non-realtime methods in callibration phase
Callibration Phase

start

Gaze Calibration Draw

Display Image

User Input Dialog

Sensor Command

COMMAND SET

Acquire

DATA RECEIVED

User Input Dialog

No

Yes

Gaze Calibrator

Yes

No

End
Running Phase
Eye center localization - done!
  - Could be more stable...
Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.
Callibration data acquisition - done!
  - Works fine.
Linear regression - (kind of) done!
  - Can’t get it right! :(
Status report

- Eye center localization - done!
  - Could be more stable...
- Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.
- Calibration data acquisition - done!
  - Works fine.
- Linear regression - (kind of) done!
  - Can’t get it right! :(
Eye center localization - done!
   Could be more stable...

Glint fitting - done!
   Needs bigger lights to be actual ellipse fitting.

Callibration data acquisition - done!
   Works fine.

Linear regression - (kind of) done!
   Can't get it right! :(
Status report

- Eye center localization - done!
  - Could be more stable...

- Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.

- Callibration data acquisition - done!
  - Works fine.

- Linear regression - (kind of) done!
  - Can’t get it right! :(
Status report

- Eye center localization - done!
  - Could be more stable...
- Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.
- Calibration data acquisition - done!
  - Works fine.
- Linear regression - (kind of) done!
  - Can’t get it right! :(
Status report

- Eye center localization - done!
  - Could be more stable...

- Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.

- Callibration data acquisition - done!
  - Works fine.

- Linear regression - (kind of) done!
  - Can’t get it right! :(
Status report

- Eye center localization - done!
  - Could be more stable...

- Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.

- Calibration data acquisition - done!
  - Works fine.

- Linear regression - (kind of) done!
  - Can’t get it right! :(
Status report

- Eye center localization - done!
  - Could be more stable...
- Glint fitting - done!
  - Needs bigger lights to be actual ellipse fitting.
- Callibration data acquisition - done!
  - Works fine.
- Linear regression - (kind of) done!
  - Can’t get it right! :(
General Lessons learned

- Never (ever, ever, EVER!) code C++ without a debugger.
- Start earlier.
- Linear regression is hard.
General Lessons learned

- Never (ever, ever, EVER!) code C++ without a debugger.
- Start earlier.
- Linear regression is hard.
General Lessons learned

- Never (ever, ever, EVER!) code C++ without a debugger.
- Start earlier.
- Linear regression is hard.
Task specific Lessons learned

- Have to use more lights.
  - One gets occluded easily.
- Use bigger lights for ellipse fitting.
Task specific Lessons learned

- Have to use more lights.
  - One gets occluded easily.
- Use bigger lights for ellipse fitting.
Questions?

Source: https://commons.wikimedia.org/wiki/File:Thats_all_folks.svg, Public Domain
All images without credit are my own work