Realtime Gaze Estimation
with Online Calibration

Li Sun, Mingli Song, Zicheng Liu, Ming-Ting Sun
Outline

- Introduction
- Limitations & goals
- Proposed method
- Results
- Demo
Gaze Estimation

Diagnostic applications:
• Eye disease diagnosis
• Human behavior studies, e.g. visual attention analysis

Interactive applications:
• Human-computer interaction
• Gaze-contingent display
Gaze Estimation Methods

2D regression based

- Eye images, Pupil-glint vector
- Appearance/features
- Regression
- Support Vector Regression, Gaussian Processes, Neural Network
- Screen coordinates
- Point of gaze

Difficult to handle head movements, require many samples for calibration
Gaze Estimation Methods

3D model based

Image features

Glints, pupil/iris center

3D model

Model the structure of the eye and the geometrical relationship of the eye and the scene

Line of gaze

Visual axis, optical axis

Complex/expensive setup, user calibration for person-specific parameters
Limitations & Goals

Limitations:
• Intrusion
• Complex, expensive setup
• Cannot tolerate head movements
• Cumbersome calibration
• Cannot work in realtime
• Low accuracy (>5°)

Goals:
• Remote gaze estimation
• Simple, low-cost setup
• Free movements
• Easy calibration/calibration-free
• Realtime processing speed
• Accurate (<3°)
Proposed Method

Gaze Feature Extraction

- 3D Face Tracking
- Face & Eye Detection
  - Iris Center Location
  - Eye Corner Location

System Calibration

- Screen-Camera Calibration
- Online (User) Calibration

3D Gaze Estimation
3D Model
3D Gaze Estimation

Input:
- Gaze features: $R_h, p_i, p_a, z_a$
- Screen-camera parameters:
  $O_s, N_s, N_w, N_v$
  $w_s, h_s, w_r, h_r, f, o$
- Personal parameters:
  $r_e, V_{ae}^H$

Output:
Gaze point: $P_g, p_g$
3D Gaze Estimation

1. Compute the 3D position of $P_a$:
   $$ p_a - o = \frac{f}{z_a} \begin{bmatrix} x_a \\ -y_a \end{bmatrix} $$

2. Calculate the offset vector $V_{ae}$ and obtain the eyeball center $O_e$:
   $$ V_{ae} = R_h V_{ae}^H $$
   $$ O_e = R_h V_{ae} + P_a $$

3. Find the iris center $P_i$:
   $$ \begin{cases} p_i - o = \frac{f}{z_i} \begin{bmatrix} x_i \\ -y_i \end{bmatrix} \\ \|P_i - O_e\|_2 = r_e \end{cases} $$

4. Obtain the gaze direction $N_g$:
   $$ N_g = \frac{P_i - O_e}{r_e} $$
3D Gaze Estimation

5 obtain the 3D gaze point $\mathbf{P}_g$:
$$\mathbf{P}_g = \mathbf{O}_e + l\mathbf{N}_g$$

$$\lambda = \| \mathbf{O}_e \mathbf{P}_g \|_2 = - \frac{(\mathbf{O}_e - \mathbf{P}_g) \cdot \mathbf{N}_s}{\mathbf{N}_g \cdot \mathbf{N}_s}$$

$$= - \frac{\mathbf{O}_e \cdot \mathbf{N}_s + d}{\mathbf{N}_g \cdot \mathbf{N}_s}$$

6 obtain the 2D gaze point $\mathbf{p}_g (u_g, v_g)$:
$$\begin{cases} 
(\mathbf{P}_g - \mathbf{O}_s) \cdot \mathbf{N}_u - \frac{w_s}{w_r} u_g = 0 \\
(\mathbf{P}_g - \mathbf{O}_s) \cdot \mathbf{N}_v - \frac{h_s}{h_r} v_g = 0
\end{cases}$$
**Gaze Feature Extraction**

Gaze features:
- Rotation matrix $R_h$
- Depth value $z_a$
- 2D iris location $p_i$
- 2D anchor point location $p_a$

The anchor point:
Inner corner of the left eye
Gaze Feature Extraction

3D face tracking:
Microsoft face tracking SDK[1]

Face & eye detection:
OpenCV[2]
Visual Context Boosting[3]

Eye corner detection:
Template filtering [4]
Online Calibration

Offline calibration:

\[
\begin{cases}
 f \cdot (x_a + R_x \cdot V_{ae}^H + r_e N_g \cdot V_x) + \left(\frac{w}{2} - u_o\right) (z_a + R_z \cdot V_{ae}^H + r_e N_g \cdot V_z) = 0 \\
 f \cdot (y_a + R_y \cdot V_{ae}^H + r_e N_g \cdot V_y) + \left(v_o - \frac{h}{2}\right) (z_a + R_z \cdot V_{ae}^H + r_e N_g \cdot V_z) = 0
\end{cases}
\]

where \( V_x = (1,0,0)^T, V_y = (0,1,0)^T, V_z = (0,0,1)^T \)

\[
R_h = \begin{bmatrix}
R_x \\
R_y \\
R_z
\end{bmatrix}
\]

For T calibration points, there are \( 2T \) equations with 4 unknowns.

- There exist noise calibration points (changes in luminance, head pose, target position, etc), the calibration result varies from time to time.
- How many calibration points are required to ensure satisfactory calibration result?
- Too many calibration points have a large impact on the usability of the system.
Online Calibration

Rewrite the linear system into the form \( Ax = b \), the least square solution can be obtained from \( A^T Ax = A^T b \).

Online calibration:
Given a new calibration point with coefficient matrix \( C_{t+1} \) and column vector \( d_{t+1} \), the coefficient matrix and column vector can be updated as follows:
\[
A_{t+1}^T A_{t+1} = A_t^T A_t + C_{t+1}^T C_{t+1} \\
A_{t+1}^T b_{t+1} = A_t^T b_t + C_{t+1}^T d_{t+1}
\]

- During online calibration, the eye parameters are constantly improved with increasing gaze estimation accuracy.
- The online calibration completes as soon as the updates of eye parameters reach convergence.
- Increasing calibration points average out the impact of noise calibration points.
Results

System setup:
- Kinect sensor
- 19” LCD screen
- Desktop PC
- 4GB memory
- Intel Core2 Quad CPU Q9550 2.83GHz
- nVidia GT310
Results

Online calibration:
The updates of eye parameters (unit: mm)

![Graphs showing the updates of eye parameters](image)
Results

Online calibration vs. offline calibration

Average gaze estimation error (unit: degree) over first $n$ calibration points from the online calibration and offline calibration with fixed number (5, 9, 16) of calibration points.
## Results

Gaze estimation accuracy:

<table>
<thead>
<tr>
<th>User</th>
<th>Accuracy (degree)</th>
<th>Eyeball radius $r_e$</th>
<th>Offset vector $V_{ae}^H$ (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$x$</td>
</tr>
<tr>
<td>1</td>
<td>1.7703</td>
<td>11.0</td>
<td>-16.1</td>
</tr>
<tr>
<td>2</td>
<td>1.7276</td>
<td>9.4</td>
<td>-14.3</td>
</tr>
<tr>
<td>3</td>
<td>1.9243</td>
<td>10.2</td>
<td>-13.0</td>
</tr>
<tr>
<td>4</td>
<td>1.8547</td>
<td>10.8</td>
<td>-14.7</td>
</tr>
</tbody>
</table>
Results

Gaze estimation accuracy:
Results

Processing speed:
Computational cost in ms of the major steps of the proposed method.

<table>
<thead>
<tr>
<th></th>
<th>Face</th>
<th>Eye</th>
<th>Blink</th>
<th>Iris</th>
<th>Eye Corner</th>
<th>Gaze</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.78</td>
<td>4.98</td>
<td>0.38</td>
<td>18.14</td>
<td>0.23</td>
<td>0.64</td>
<td>30.15</td>
</tr>
</tbody>
</table>
Demo: 3D Chess Game
References


Thank you!