Computer vision based eye-gaze tracking

Prasad Gabbur

Lecture given by Prasad due to NSF panel travel.
The general thrust is relevant for the exam.

Ellipse fitting and tracking in the presence of outliers

- Seems easy?
- Hard problem!
  - Presence of noisy data and outliers

Pupil (ellipse) tracking

- Pupil center is a feature for eye-gaze estimation
- Track pupil boundary ellipse

Pupil boundary edge points

Ellipse overlaid on the eye image

Setup (simplified)

- Even field => ON-AXIS LED is ON
- Odd field => ON-AXIS LED is OFF
Bright and dark pupil

Bright pupil formation is similar to red-eye effect in photography.

Red-eye effect

Red-eye is due to the image of the retinal surface
Red in humans due to retinal blood vessels
Other animals show other colors due to a reflective layer—tapetum—helps them in night vision

Even and odd field images

- Even field – bright pupil formation (similar to red-eye effect).
- Odd field – dark pupil.
- Difference image – pupil region enhanced.

Pupil localization

1. Correct aspect ratio
2. Threshold and localize pupil region
Pupil edge detection

- Gradient magnitude within the processing window
  
- Large gradient magnitude => potential edge

Radial edge detection with hysteresis
- 180-200 edge points
- Spurious edge pixels (e.g., glint hole boundary)

Ellipse fitting

- Bunch of points
  - How to fit an ellipse?
  - Noise in localization
  - Outliers

- Hough transform
  - Time intensive!
- Least squares
  - Minimize sum squared of some distance measure (?)

Algebraic Distance (AD)

Equation of ellipse
\[ F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0 \] (Conic)
\[ 4ac - b^2 > 0 \] (Constraint for ellipse)

Algebraic distance
\[ AD(x_i, y_i) = F(x_i, y_i) = ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f \]

To avoid sign of AD, use squared AD: \[ AD^2(x_i, y_i) = F^2(x_i, y_i) \]
Least squares ellipse fitting

- Edge points: \( (x_i, y_i), i = 1 \ldots N \)
- Minimize sum of square of ADs:
  \[
  E = \sum_{i=1}^{N} F^2(x_i, y_i)
  \]
- Possible to convert to linear least squares (Homogeneous)
  - Non-homogeneous
  - Homogeneous

Least squares ellipse fitting

- Let
  \[
  U = \begin{bmatrix}
    u_{11}^T \\
    u_{21}^T \\
    \vdots \\
    u_{N1}^T
  \end{bmatrix},
  p = \begin{bmatrix}
    a \\
    b \\
    c \\
    d \\
    e
  \end{bmatrix}
  \]
- Minimize
  \[
  E = \sum_{i=1}^{N} F^2(x_i, y_i) = (Up)^T Up = p^T U^T Up
  \]
- Inequality constraint \( 4ac - b^2 > 0 \)
- Equality constraint \( 4ac - b^2 = 1 \)
  - Avoid trivial solution \( p = 0 \)
  - Force a scale on \( p \) because if \( p \) is a solution, then any scalar multiple of \( p \) is also a solution

Least squares ellipse fitting

- Quadratic constraint \( 4ac - b^2 = 1 \)
- is equivalent to
  \[
  \begin{bmatrix}
    0 & 0 & 2 & 0 & 0 & 0 \\
    0 & -1 & 0 & 0 & 0 & 0 \\
    2 & 0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & 0 & 0
  \end{bmatrix}
  \]
\[
  p = \begin{bmatrix}
    0 \\
    1 \\
    0 \\
    0 \\
    0 \\
    0
  \end{bmatrix}
  \]
\[
  p^T Cp = 1
  \]

**Least squares ellipse fitting**

- Solution is the positive generalized eigenvector of
  \[ U^T U p = \lambda C p \]

- Least squares
  - Is it a perfect (flawless) solution?
  - Problems?

**Least squares ellipse fitting**

- Direct least squares ellipse fitting*
  - Use RANSAC (robust to outliers)
    - Random (RAN) Sample (SA) Consensus (C)


**RANSAC**

- Robust statistical model selection technique in the presence of outliers
- Idea
  - Fit model using minimum # training points chosen at random
  - Evaluate the model against the remaining points according to an error threshold
  - Points agree (consensus) or disagree (outlier)
  - Compute % of consensus points
  - Repeat the above for certain #iterations
  - Keep the model with the highest % consensus
  - Refit using consensus + training

**RANSAC for ellipse fitting**

- Min # of training points: 6
- Randomly sample
- Fit using least squares
- Evaluate
- Choose the best
RANSAC

- Number of iterations
  \[ n = \frac{\log(1-\varepsilon)}{\log(1-\alpha^k)} \]
- \( n \) – Number of iterations
- \( \varepsilon \) - Probability of finding at least one outlier free sample
- \( \alpha \) - Probability of inliers in your data
- \( k \) – Number of training points sampled every iteration

Sample result

Tracking

- Brute force: Detect ellipse every video frame
  - RANSAC: Computationally intensive
- Better: Detect + Track
  - Ellipse usually does not change too much between adjacent frames
- Principle
  - Detect ellipse in a frame
  - Predict ellipse in next frame
  - Refine prediction using data available from next frame
  - If track lost, re-detect and continue

Detect+Track
Tracking example