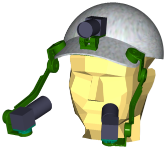
Lecture: Eye Camera Settings and Calibration

Andrew T. Duchowski

School of Computing, Clemson University

Univeristat Autònoma de Barcelona, 2009, 25 November, Barcelona, Spain



DIY Eye Tracking

- To fully understand how it works, build it yourself
- Babcock and Pelz (2004) were one of the first to describe a DIY wearable analog video tracker
  - an expensive video multiplexer was needed to synchronize eye and scene cameras
  - system relied on somewhat dated (by today's standards) video recorder (a Sony DCR-TRV19 DVR)
  - nevertheless, fostered nascent open-source movement

DIY Eye Tracking

- To fully understand how it works, build it yourself
- Babcock and Pelz (2004) were one of the first to describe a DIY wearable analog video tracker
  - an expensive video multiplexer was needed to synchronize eye and scene cameras
  - system relied on somewhat dated (by today's standards) video recorder (a Sony DCR-TRV19 DVR)
  - nevertheless, fostered nascent open-source movement

DIY Eye Tracking

- To fully understand how it works, build it yourself
- Babcock and Pelz (2004) were one of the first to describe a DIY wearable analog video tracker
  - an expensive video multiplexer was needed to synchronize eye and scene cameras
  - system relied on somewhat dated (by today's standards) video recorder (a Sony DCR-TRV19 DVR)
  - nevertheless, fostered nascent open-source movement

DIY Eye Tracking

- To fully understand how it works, build it yourself
- Babcock and Pelz (2004) were one of the first to describe a DIY wearable analog video tracker
    - an expensive video multiplexer was needed to synchronize eye and scene cameras
    - system relied on somewhat dated (by today's standards) video recorder (a Sony DCR-TRV19 DVR)
    - nevertheless, fostered nascent open-source movement

DIY Eye Tracking

- To fully understand how it works, build it yourself
- Babcock and Pelz (2004) were one of the first to describe a DIY wearable analog video tracker
  - an expensive video multiplexer was needed to synchronize eye and scene cameras
  - system relied on somewhat dated (by today's standards) video recorder (a Sony DCR-TRV19 DVR)
  - nevertheless, fostered nascent open-source movement

General Motion Capture

An eye tracker is a specialized kind of motion capture (*mocap*) device, delineated within four classes:

- **Electromechanical:** skin sensors around the eye cavity (electro-oculography, or EOG)
- **Electromagnetic:** a metallic stalk is fixed to contact lens
- **Video-oculography, or VOG:** analysis (historically manual) of video frames
- **Video-based with IR reflection:** eye trackers benefit from one (or more) of four types of IR reflections, two from the front and rear surface of the cornea, two from the front and rear surface of the crystalline lens; all four reflections are known as the Purkinje images (Crane, 1994)

For comparison with video mocap, see Menache (2000)

General Motion Capture

An eye tracker is a specialized kind of motion capture (*mocap*) device, delineated within four classes:

- **Electromechanical:** skin sensors around the eye cavity (electro-oculography, or EOG)
- **Electromagnetic:** a metallic stalk is fixed to contact lens
- **Video-oculography, or VOG:** analysis (historically manual) of video frames
- **Video-based with IR reflection:** eye trackers benefit from one (or more) of four types of IR reflections, two from the front and rear surface of the cornea, two from the front and rear surface of the crystalline lens; all four reflections are known as the Purkinje images (Crane, 1994)

For comparison with video mocap, see Menache (2000)

General Motion Capture

An eye tracker is a specialized kind of motion capture (*mocap*) device, delineated within four classes:

- **Electromechanical:** skin sensors around the eye cavity (electro-oculography, or EOG)
- **Electromagnetic:** a metallic stalk is fixed to contact lens
- **Video-oculography, or VOG:** analysis (historically manual) of video frames
- **Video-based with IR reflection:** eye trackers benefit from one (or more) of four types of IR reflections, two from the front and rear surface of the cornea, two from the front and rear surface of the crystalline lens; all four reflections are known as the Purkinje images (Crane, 1994)

For comparison with video mocap, see Menache (2000)

General Motion Capture

An eye tracker is a specialized kind of motion capture (*mocap*) device, delineated within four classes:

- **Electromechanical:** skin sensors around the eye cavity (electro-oculography, or EOG)
- **Electromagnetic:** a metallic stalk is fixed to contact lens
- **Video-oculography, or VOG:** analysis (historically manual) of video frames
- **Video-based with IR reflection:** eye trackers benefit from one (or more) of four types of IR reflections, two from the front and rear surface of the cornea, two from the front and rear surface of the crystalline lens; all four reflections are known as the Purkinje images (Crane, 1994)

For comparison with video mocap, see Menache (2000)

Two Approaches to Video-Based Eye Tracking

1. Based on a 3D eyeball model (e.g., Gullstrand's), compute the ray emanating from a central (nodal) point within the eye, then compute ray/plane intersections within the scene,

$$p = c + t \cdot L,$$

   where p is the 3D Point Of Gaze, or POG, c is the center of the eye (or cornea, modeled by a sphere), t is the parametric distance along the ray L coinciding with the optical or visual axis
2. Based on traditional video-oculography, estimate the limbus center (x, y)—the POG in the scene image is obtained via a mapping of (x, y) to scene coordinates (s_x, s_y) given known calibration point coordinators

- See Hennessey (2008) for a detailed review

Two Approaches to Video-Based Eye Tracking

1. Based on a 3D eyeball model (e.g., Gullstrand's), compute the ray emanating from a central (nodal) point within the eye, then compute ray/plane intersections within the scene,

$$p = c + t \cdot L,$$

   where p is the 3D Point Of Gaze, or POG, c is the center of the eye (or cornea, modeled by a sphere), t is the parametric distance along the ray L coinciding with the optical or visual axis
2. Based on traditional video-oculography, estimate the limbus center (x, y)—the POG in the scene image is obtained via a mapping of (x, y) to scene coordinates (s_x, s_y) given known calibration point coordinators

- See Hennessey (2008) for a detailed review

Two Approaches to Video-Based Eye Tracking

1. Based on a 3D eyeball model (e.g., Gullstrand's), compute the ray emanating from a central (nodal) point within the eye, then compute ray/plane intersections within the scene,

$$p = c + t \cdot L,$$

   where p is the 3D Point Of Gaze, or POG, c is the center of the eye (or cornea, modeled by a sphere), t is the parametric distance along the ray L coinciding with the optical or visual axis
2. Based on traditional video-oculography, estimate the limbus center (x, y)—the POG in the scene image is obtained via a mapping of (x, y) to scene coordinates (s_x, s_y) given known calibration point coordinators

- See Hennessey (2008) for a detailed review

Building Our Own

- For years I refused to build my own
- With digital cameras becoming affordable and easier to use, it became feasible (and fairly easy)
- Derrick Parkhurst suggested a pair of cameras to use
- My graduate student, Wayne J. Ryan, took up the project (most of these notes are based on his wok)

Building Our Own

- For years I refused to build my own
- With digital cameras becoming affordable and easier to use, it became feasible (and fairly easy)
- Derrick Parkhurst suggested a pair of cameras to use
- My graduate student, Wayne J. Ryan, took up the project (most of these notes are based on his wok)

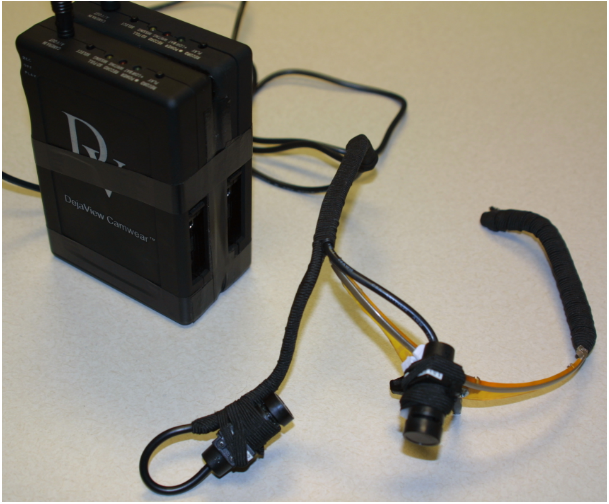
Building Our Own

- For years I refused to build my own
- With digital cameras becoming affordable and easier to use, it became feasible (and fairly easy)
- Derrick Parkhurst suggested a pair of cameras to use
- My graduate student, Wayne J. Ryan, took up the project (most of these notes are based on his wok)

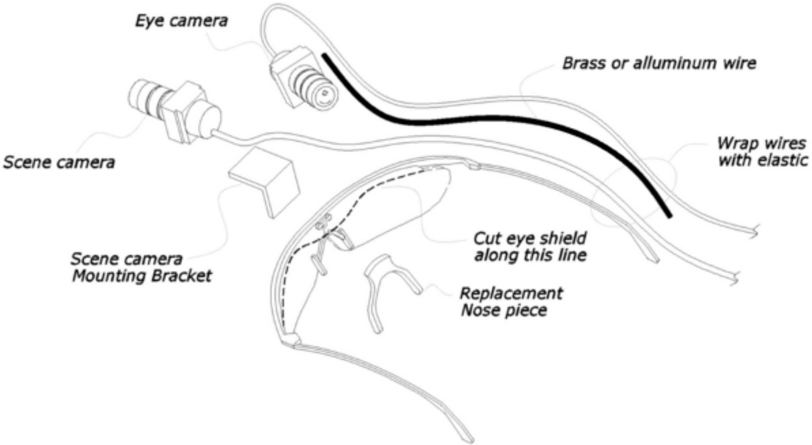
Building Our Own

- For years I refused to build my own
- With digital cameras becoming affordable and easier to use, it became feasible (and fairly easy)
- Derrick Parkhurst suggested a pair of cameras to use
- My graduate student, Wayne J. Ryan, took up the project (most of these notes are based on his wok)

Off-line Visible Spectrum Wearable Tracker



Visible Spectrum Wearable Schematic



Parts List

Vendor	Description	Price
AOSafety	X-Factor XF503 safety glasses	$10.00
AOSafety	I-Pilot 90714 nose piece	$10.00
Jose Ann Fabric	Polyester braided elastic	$1.50
Home Depot	Screws	$0.75
Amazon.com	Aluminum pod 1/4″ dia. 18″ long	$5.00
DejaView	Two Camwear Model 200 cameras	$500.00
	Total	$527.25

Visible Spectrum Design

- Design based to a large extent on that of Li et al. (2006)
- We first explored tracking in the visible spectrum, following Li and Parkhurst (2006)
- Gaze tracking relies on tracking of the limbus, or the iris/sclera boundary
- No IR simplifies design but complicates image analysis
- Headgear must be stable to offer sufficiently stable reference point for gaze point estimation

Visible Spectrum Design

- Design based to a large extent on that of Li et al. (2006)
- We first explored tracking in the visible spectrum, following Li and Parkhurst (2006)
- Gaze tracking relies on tracking of the limbus, or the iris/sclera boundary
- No IR simplifies design but complicates image analysis
- Headgear must be stable to offer sufficiently stable reference point for gaze point estimation

Visible Spectrum Design

- Design based to a large extent on that of Li et al. (2006)
- We first explored tracking in the visible spectrum, following Li and Parkhurst (2006)
- Gaze tracking relies on tracking of the limbus, or the iris/sclera boundary
- No IR simplifies design but complicates image analysis
- Headgear must be stable to offer sufficiently stable reference point for gaze point estimation

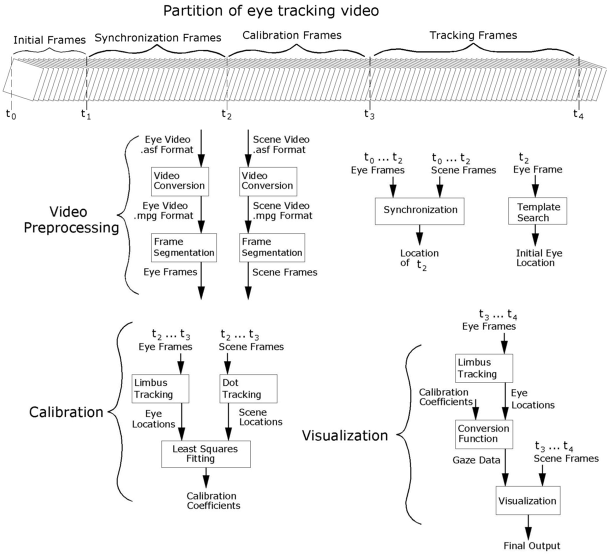
Visible Spectrum Design

- Design based to a large extent on that of Li et al. (2006)
- We first explored tracking in the visible spectrum, following Li and Parkhurst (2006)
- Gaze tracking relies on tracking of the limbus, or the iris/sclera boundary
- No IR simplifies design but complicates image analysis
- Headgear must be stable to offer sufficiently stable reference point for gaze point estimation

Visible Spectrum Design

- Design based to a large extent on that of Li et al. (2006)
- We first explored tracking in the visible spectrum, following Li and Parkhurst (2006)
- Gaze tracking relies on tracking of the limbus, or the iris/sclera boundary
- No IR simplifies design but complicates image analysis
- Headgear must be stable to offer sufficiently stable reference point for gaze point estimation

Software Architecture



Basic Steps

1. Video preprocessing: convert to some known format from which you can extract frames
   - we had problem with `.asf` format; turns out it was "proprietary" by virtue of 4 unknown bytes in file header, easily "fixed" once known
   - finding good open source codec library was a problem; latest version of `ffmpeg` is excellent
2. Synchronization: line up eye and scene video feeds via "flash screen" (as suggested by Li and Parkhurst (2006))
3. Calibration: track pupil center and calibration dot
4. Visualization/Analysis: detect fixations in raw gaze point data stream

Basic Steps

1. Video preprocessing: convert to some known format from which you can extract frames
   - we had problem with `.asf` format; turns out it was "proprietary" by virtue of 4 unknown bytes in file header, easily "fixed" once known
   - finding good open source codec library was a problem; latest version of `ffmpeg` is excellent
2. Synchronization: line up eye and scene video feeds via "flash screen" (as suggested by Li and Parkhurst (2006))
3. Calibration: track pupil center and calibration dot
4. Visualization/Analysis: detect fixations in raw gaze point data stream

Basic Steps

1. Video preprocessing: convert to some known format from which you can extract frames
   - we had problem with `.asf` format; turns out it was “proprietary” by virtue of 4 unknown bytes in file header, easily “fixed” once known
   - finding good open source codec library was a problem; latest version of `ffmpeg` is excellent
2. Synchronization: line up eye and scene video feeds via “flash screen” (as suggested by Li and Parkhurst (2006))
3. Calibration: track pupil center and calibration dot
4. Visualization/Analysis: detect fixations in raw gaze point data stream

Basic Steps

1. Video preprocessing: convert to some known format from which you can extract frames
   - we had problem with `.asf` format; turns out it was "proprietary" by virtue of 4 unknown bytes in file header, easily "fixed" once known
   - finding good open source codec library was a problem; latest version of `ffmpeg` is excellent
2. Synchronization: line up eye and scene video feeds via "flash screen" (as suggested by Li and Parkhurst (2006))
3. Calibration: track pupil center and calibration dot
4. Visualization/Analysis: detect fixations in raw gaze point data stream

Basic Steps

1. Video preprocessing: convert to some known format from which you can extract frames
   - we had problem with `.asf` format; turns out it was "proprietary" by virtue of 4 unknown bytes in file header, easily "fixed" once known
   - finding good open source codec library was a problem; latest version of `ffmpeg` is excellent
2. Synchronization: line up eye and scene video feeds via "flash screen" (as suggested by Li and Parkhurst (2006))
3. Calibration: track pupil center and calibration dot
4. Visualization/Analysis: detect fixations in raw gaze point data stream

Basic Steps

1. Video preprocessing: convert to some known format from which you can extract frames
   - we had problem with `.asf` format; turns out it was "proprietary" by virtue of 4 unknown bytes in file header, easily "fixed" once known
   - finding good open source codec library was a problem; latest version of `ffmpeg` is excellent
2. Synchronization: line up eye and scene video feeds via "flash screen" (as suggested by Li and Parkhurst (2006))
3. Calibration: track pupil center and calibration dot
4. Visualization/Analysis: detect fixations in raw gaze point data stream

Pupil Tracking

- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

(a) image frame (b) Canny edges (c) chamfer image (d) pattern

Pupil Tracking

- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

(a) image frame (b) Canny edges (c) chamfer image (d) pattern

Pupil Tracking

- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

(a) image frame (b) Canny edges (c) chamfer image (d) pattern

Pupil Tracking

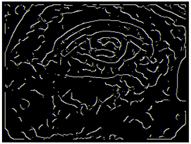
- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

(a) image frame (b) Canny edges (c) chamfer image (d) pattern

Pupil Tracking

- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

(a) image frame



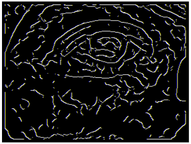
(b) Canny edges

(c) chamfer image (d) pattern

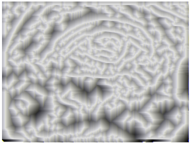
Pupil Tracking

- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

(a) image frame



(b) Canny edges



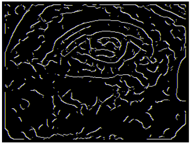
(c) chamfer image

(d) pattern

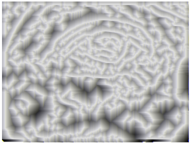
Pupil Tracking

- Major problem is pupil tracking—a good starting point is the *Starburst* algorithm (Li et al., 2006)
- Idea is to shoot out rays from within pupil to locate limbic boundary feature points along those rays
- A seed point is needed from which to shoot rays
- Our initial approach was based on template matching:
  - Canny's (1986) edge detector
  - Borgefors' (1988) chamfer image
  - pixel pattern

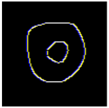
(a) image frame



(b) Canny edges



(c) chamfer image



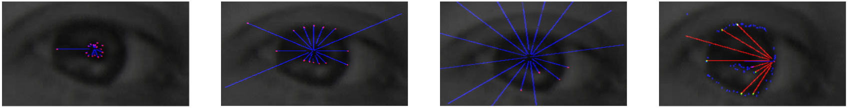
(d) pattern

Searchmap

- A gradient image is also use to guide the algorithm for finding Starburst's initial seed point

```
MakeSearchMap(Gradient image G, Chamfer image C,
              Template image T, Template gradient GT)
{
  for all template locations x,y {
    sum = 0;
    for all locations dx,dy ∈ T ≠ 0
      sum += G(x + dx, y + dy) · GT(dx, dy) × C(x + dx, y + dy);
    M(x, y) = sum; // M is the search map;
  }
  return maximum pixel in M;
}
```

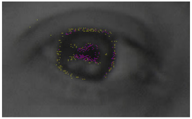
Feature Point Detection (Starburst)



FeatureDetect(Gradient image G, Threshold T, Start point S)
{
  for θ **from** 0 **to** 2π { // phase 1
    $L = S$; // L is the current location
    $\Delta_x = \cos(\theta)$; $\Delta_y = \sin(\theta)$;
    while $G(L) \cdot \Delta < T$ { $L(x)$ += Δ_x; $L(y)$ += Δ_y; }
    push L **on** P; // P is a point stack
  }
  for θ **from** 2π **to** 0 { // phase 2
    pop L **off** P; **add** L **to** P'; // P' is a point set
    for α **from** $-(\theta + \frac{\pi}{6})$ **to** $-(\theta - \frac{\pi}{6})$
      while $G_L \cdot \Delta < T$ { $L(x)$ += Δ_x; $L(y)$ += Δ_y; }
    add L **to** P';
  }
  return P';
}

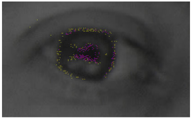
Ellipse Fitting

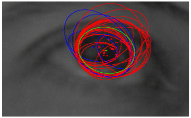
- Starburst returns a set of feature points
- Next step is to fit ellipses to points
- Starburst used RANSAC (Random Sample Consensus)
- Our visible spectrum approach fit multiple ellipses and distinguished between those that fit the pupil, those that fit the limbus, and those that spanned both (erroneous)
- Ellipse is used to estimate pupil center (x, y)



Ellipse Fitting

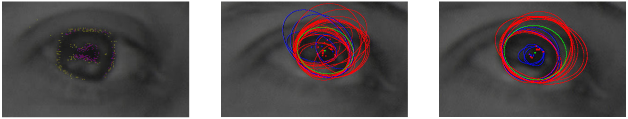
- Starburst returns a set of feature points
- Next step is to fit ellipses to points
- Starburst used RANSAC (Random Sample Consensus)
- Our visible spectrum approach fit multiple ellipses and distinguished between those that fit the pupil, those that fit the limbus, and those that spanned both (erroneous)
- Ellipse is used to estimate pupil center (x, y)





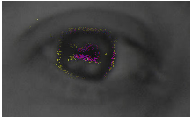
Ellipse Fitting

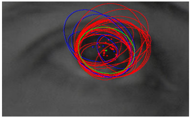
- Starburst returns a set of feature points
- Next step is to fit ellipses to points
- Starburst used RANSAC (Random Sample Consensus)
- Our visible spectrum approach fit multiple ellipses and distinguished between those that fit the pupil, those that fit the limbus, and those that spanned both (erroneous)
- Ellipse is used to estimate pupil center (x, y)

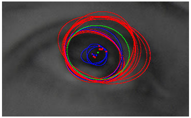


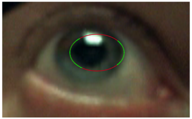
Ellipse Fitting

- Starburst returns a set of feature points
- Next step is to fit ellipses to points
- Starburst used RANSAC (Random Sample Consensus)
- Our visible spectrum approach fit multiple ellipses and distinguished between those that fit the pupil, those that fit the limbus, and those that spanned both (erroneous)
- Ellipse is used to estimate pupil center (x, y)



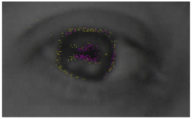
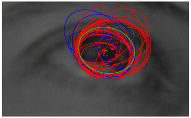
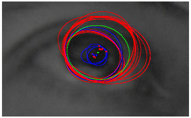
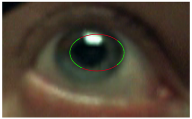






Ellipse Fitting

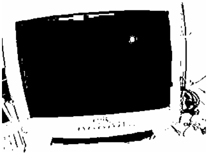
- Starburst returns a set of feature points
- Next step is to fit ellipses to points
- Starburst used RANSAC (Random Sample Consensus)
- Our visible spectrum approach fit multiple ellipses and distinguished between those that fit the pupil, those that fit the limbus, and those that spanned both (erroneous)
- Ellipse is used to estimate pupil center (x, y)

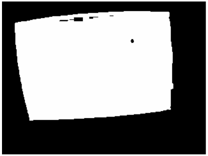
Calibration

- Calibration maps pupil center (x, y) in eye frames to calibration points (s_x, s_y) in the scene frames
- Calibration dots are displayed on a computer monitor viewed by the scene camera, tracked by a 5-step algorithm

(a) scene frame



(b) low threshold



(c) get largest region

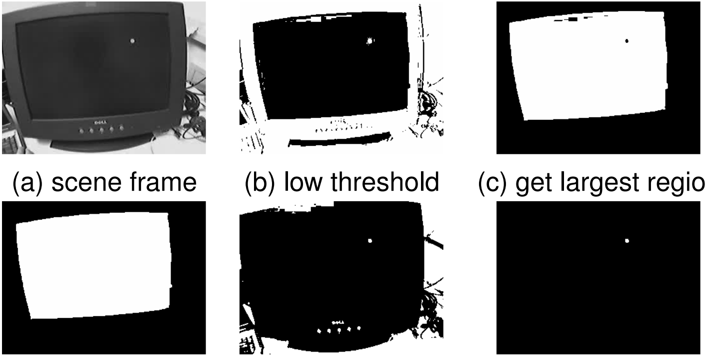
(d) flood fill holes

(e) high threshold

(f) final result

Calibration

- Calibration maps pupil center (x, y) in eye frames to calibration points (s_x, s_y) in the scene frames
- Calibration dots are displayed on a computer monitor viewed by the scene camera, tracked by a 5-step algorithm



(a) scene frame (b) low threshold (c) get largest region

(d) flood fill holes (e) high threshold (f) final result

Computing the Calibration Homography

- With calibration screen coordinates (s_x, s_y) and pupil center (x, y), the mapping is expressed as a second order polynomial (C. H. Morimoto & Mimica, 2005):

$$\begin{aligned} s_x &= a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 \\ s_y &= b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2 \end{aligned}$$

- The parameters $\{a_0, \ldots, a_5\}$ and $\{b_0, \ldots, b_5\}$ can be computed via Lagrange's method of least squares (e.g, see Lancaster and Šalkauskas (1986))

Computing the Calibration Homography

- With calibration screen coordinates (s_x, s_y) and pupil center (x, y), the mapping is expressed as a second order polynomial (C. H. Morimoto & Mimica, 2005):

$$\begin{aligned} s_x &= a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 \\ s_y &= b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2 \end{aligned}$$

- The parameters $\{a_0, \ldots, a_5\}$ and $\{b_0, \ldots, b_5\}$ can be computed via Lagrange's method of least squares (e.g, see Lancaster and Šalkauskas (1986))

Computing the Calibration Homography

- Reformulate the above equations by writing:

$$\begin{bmatrix} s_{1x} & s_{1y} \\ s_{2x} & s_{2y} \\ \vdots & \vdots \\ s_{ix} & s_{iy} \\ \vdots & \vdots \\ s_{nx} & s_{ny} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & xy_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & xy_2 & x_2^2 & y_2^2 \\ \vdots & \vdots & \vdots & & & \\ 1 & x_i & y_i & xy_i & x_i^2 & y_i^2 \\ \vdots & \vdots & \vdots & & & \\ 1 & x_n & y_n & xy_n & x_n^2 & y_n^2 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ a_5 & b_5 \end{bmatrix}$$

or in matrix notation,

$$\mathbf{Y} = \mathbf{X}\hat{\mathbf{B}}$$

- An estimate of $\hat{\mathbf{B}}$ is obtained by left-multiplying by $(\mathbf{X}^T\mathbf{X})^{-1}$

$$\hat{\mathbf{B}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- Note that this is done for n calibration points, each sampled multiple times

Computing the Calibration Homography

- Reformulate the above equations by writing:

$$\begin{bmatrix} s_{1x} & s_{1y} \\ s_{2x} & s_{2y} \\ \vdots & \vdots \\ s_{ix} & s_{iy} \\ \vdots & \vdots \\ s_{nx} & s_{ny} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & xy_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & xy_2 & x_2^2 & y_2^2 \\ \vdots & \vdots & \vdots & & & \\ 1 & x_i & y_i & xy_i & x_i^2 & y_i^2 \\ \vdots & \vdots & \vdots & & & \\ 1 & x_n & y_n & xy_n & x_n^2 & y_n^2 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ a_5 & b_5 \end{bmatrix}$$

or in matrix notation,

$$\mathbf{Y} = \mathbf{X}\hat{\mathbf{B}}$$

- An estimate of $\hat{\mathbf{B}}$ is obtained by left-multiplying by $(\mathbf{X}^T\mathbf{X})^{-1}$

$$\hat{\mathbf{B}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- Note that this is done for n calibration points, each sampled multiple times

Computing the Calibration Homography

- Reformulate the above equations by writing:

$$\begin{bmatrix} s_{1x} & s_{1y} \\ s_{2x} & s_{2y} \\ \vdots & \vdots \\ s_{ix} & s_{iy} \\ \vdots & \vdots \\ s_{nx} & s_{ny} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & xy_1 & x_1^2 & y_1^2 \\ 1 & x_2 & y_2 & xy_2 & x_2^2 & y_2^2 \\ \vdots & \vdots & \vdots & & & \\ 1 & x_i & y_i & xy_i & x_i^2 & y_i^2 \\ \vdots & \vdots & \vdots & & & \\ 1 & x_n & y_n & xy_n & x_n^2 & y_n^2 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ a_5 & b_5 \end{bmatrix}$$

or in matrix notation,

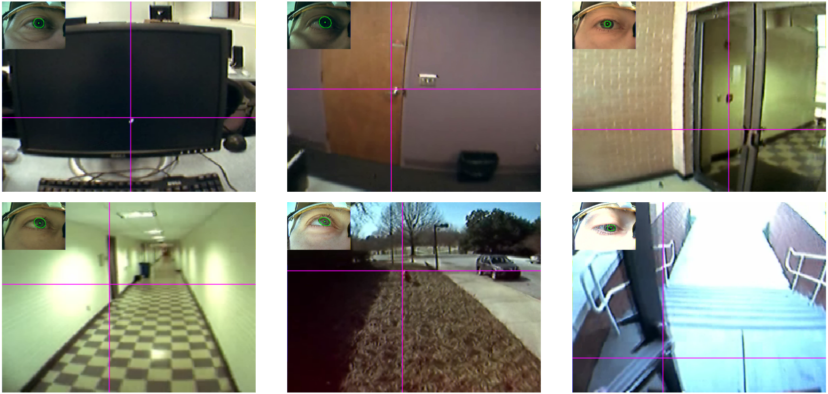
$$\mathbf{Y} = \mathbf{X}\hat{\mathbf{B}}$$

- An estimate of $\hat{\mathbf{B}}$ is obtained by left-multiplying by $(\mathbf{X}^T\mathbf{X})^{-1}$

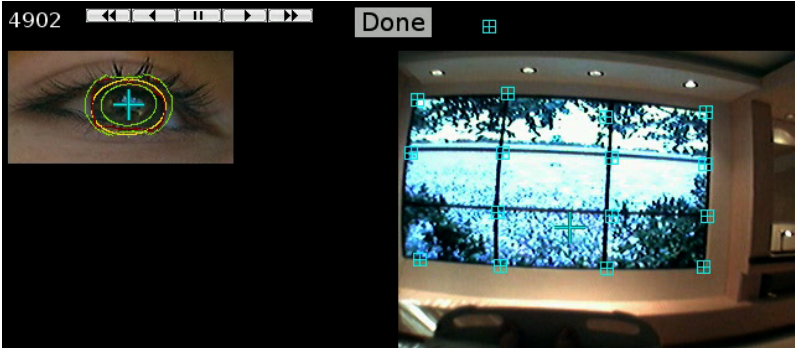
$$\hat{\mathbf{B}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

- Note that this is done for n calibration points, each sampled multiple times

Preliminary Results

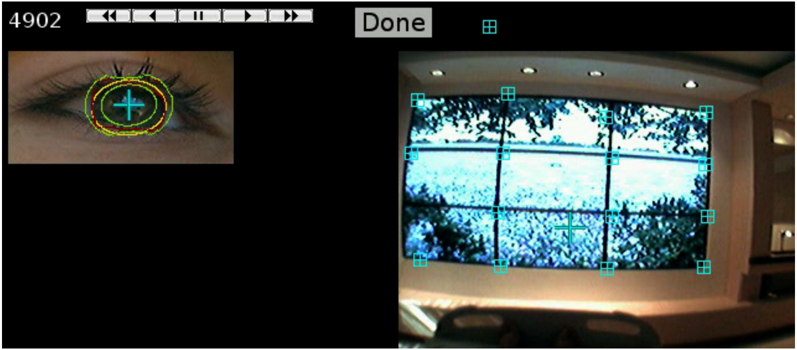


Natural Tasks



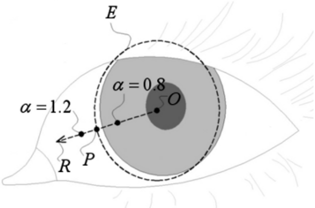
- Eventually we'd like to investigate natural tasks
- Count fixations falling over Areas Of Interest (AOIs)

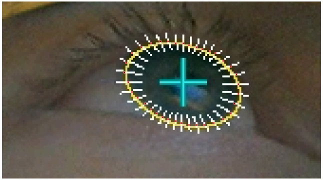
Natural Tasks



- Eventually we'd like to investigate natural tasks
- Count fixations falling over Areas Of Interest (AOIs)

Tracking Refinements

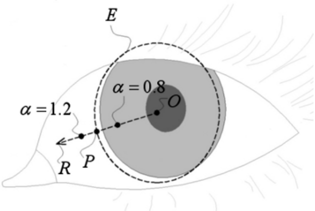


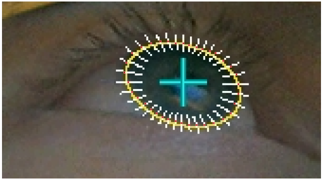


- Cast ray R from origin O
- Terminate ray if it exits a dark region
  (check gradient magnitude ∇ collinear with ray)
- max ∇ signifies feature point
- Constrain search by exploiting temporal locality
  - assume points are near E identified in previous frame
  - if P is intersection of R and E constrain search via

$$\max \nabla(O + \alpha(P - O)) \; : \; 0.8 < \alpha < 1.2)$$

Tracking Refinements

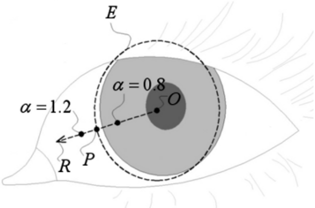


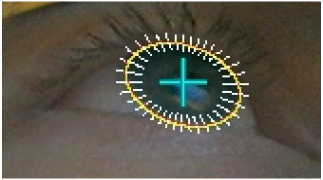


- Cast ray R from origin O
- Terminate ray if it exits a dark region
  (check gradient magnitude ∇ collinear with ray)
- max ∇ signifies feature point
- Constrain search by exploiting temporal locality
  - assume points are near E identified in previous frame
  - if P is intersection of R and E constrain search via

$$\max \nabla(O + \alpha(P - O)) \;:\; 0.8 < \alpha < 1.2)$$

Tracking Refinements

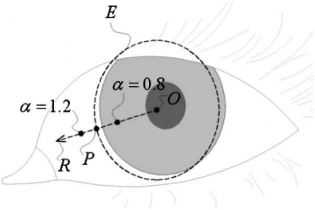


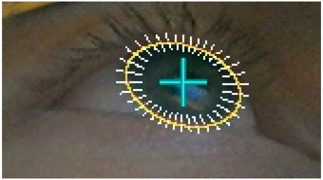


- Cast ray R from origin O
- Terminate ray if it exits a dark region (check gradient magnitude ∇ collinear with ray)
- max ∇ signifies feature point
- Constrain search by exploiting temporal locality
  - assume points are near E identified in previous frame
  - if P is intersection of R and E constrain search via

$$\max \nabla(O + \alpha(P - O)) \ : \ 0.8 < \alpha < 1.2)$$

Tracking Refinements

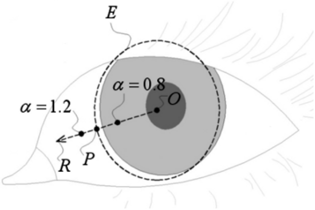


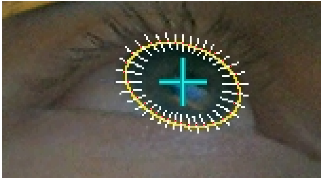


- Cast ray R from origin O
- Terminate ray if it exits a dark region (check gradient magnitude ∇ collinear with ray)
- max ∇ signifies feature point
- Constrain search by exploiting temporal locality
  - assume points are near E identified in previous frame
  - if P is intersection of R and E constrain search via

$$\max \nabla(O + \alpha(P - O)) \ : \ 0.8 < \alpha < 1.2)$$

Tracking Refinements

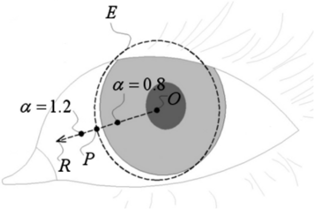


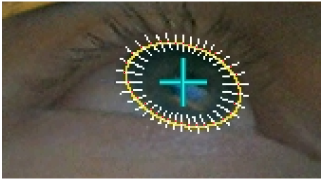


- Cast ray R from origin O
- Terminate ray if it exits a dark region
  (check gradient magnitude ∇ collinear with ray)
- max ∇ signifies feature point
- Constrain search by exploiting temporal locality
  - assume points are near E identified in previous frame
  - if P is intersection of R and E constrain search via

$$\max \nabla(O + \alpha(P - O)) \ : \ 0.8 < \alpha < 1.2)$$

Tracking Refinements

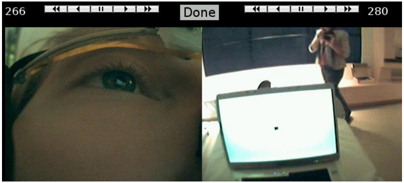


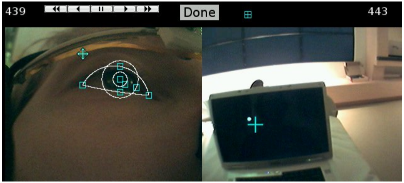


- Cast ray *R* from origin *O*
- Terminate ray if it exits a dark region (check gradient magnitude ∇ collinear with ray)
- max ∇ signifies feature point
- Constrain search by exploiting temporal locality
  - assume points are near *E* identified in previous frame
  - if *P* is intersection of *R* and *E* constrain search via

$$\max \nabla(O + \alpha(P - O)) \;:\; 0.8 < \alpha < 1.2)$$

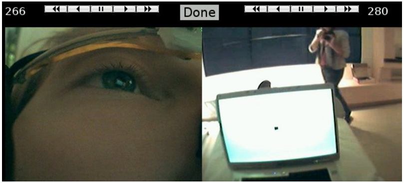
Interactive Manual Intervention

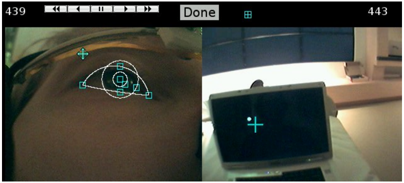




- Manual intervention via GUI to synchronize video streams (find flashes in image frames)
- Seed Starburst via manually overlayed template
- Similarly, use crosshair for initial location of calibration dot

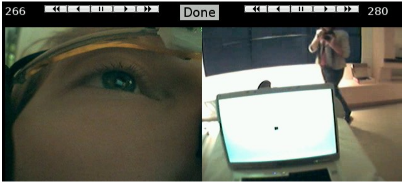
Interactive Manual Intervention

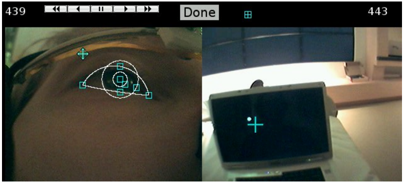




- Manual intervention via GUI to synchronize video streams (find flashes in image frames)
- Seed Starburst via manually overlayed template
- Similarly, use crosshair for initial location of calibration dot

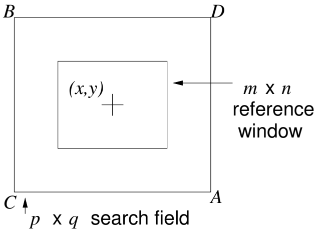
Interactive Manual Intervention





- Manual intervention via GUI to synchronize video streams (find flashes in image frames)
- Seed Starburst via manually overlayed template
- Similarly, use crosshair for initial location of calibration dot

AOI Tracking

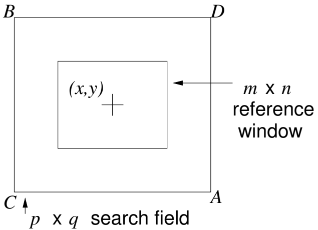


- Need to track AOIs to count fixations falling within
- Similar to *match-moving* trackbox (Paolini, 2006)
- Bright spot is found where $I(x, y) - \mu$ is maximum
- For efficiency, a summed area table S is precomputed (Crow, 1984; Viola & Jones, 2004)

$$S(x, y) = \sum_i \sum_j I(i, j), \quad 0 < i < x, \quad 0 < j < y$$

$$\mu = \frac{(S(A) + S(B)) - (S(C) + S(D))}{p \times q}$$

AOI Tracking

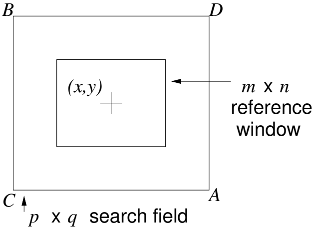


- Need to track AOIs to count fixations falling within
- Similar to *match-moving* trackbox (Paolini, 2006)
- Bright spot is found where $I(x, y) - \mu$ is maximum
- For efficiency, a summed area table S is precomputed (Crow, 1984; Viola & Jones, 2004)

$$S(x, y) = \sum_i \sum_j I(i, j), \quad 0 < i < x, \quad 0 < j < y$$

$$\mu = \frac{(S(A) + S(B)) - (S(C) + S(D))}{p \times q}$$

AOI Tracking

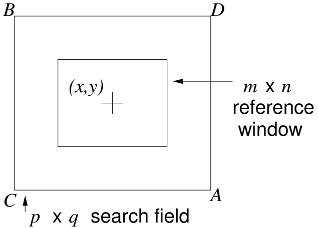


- Need to track AOIs to count fixations falling within
- Similar to *match-moving* trackbox (Paolini, 2006)
- Bright spot is found where $I(x, y) - \mu$ is maximum
- For efficiency, a summed area table S is precomputed (Crow, 1984; Viola & Jones, 2004)

$$S(x, y) = \sum_i \sum_j I(i, j), \quad 0 < i < x, \quad 0 < j < y$$

$$\mu = \frac{(S(A) + S(B)) - (S(C) + S(D))}{p \times q}$$

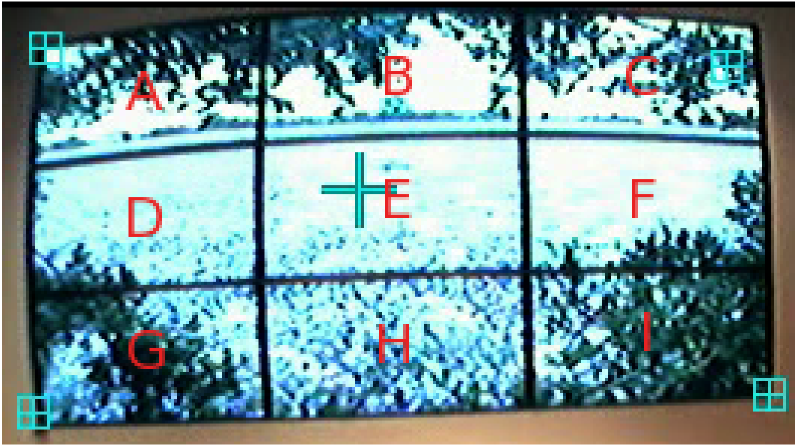
AOI Tracking



- Need to track AOIs to count fixations falling within
- Similar to *match-moving* trackbox (Paolini, 2006)
- Bright spot is found where $I(x, y) - \mu$ is maximum
- For efficiency, a summed area table S is precomputed (Crow, 1984; Viola & Jones, 2004)

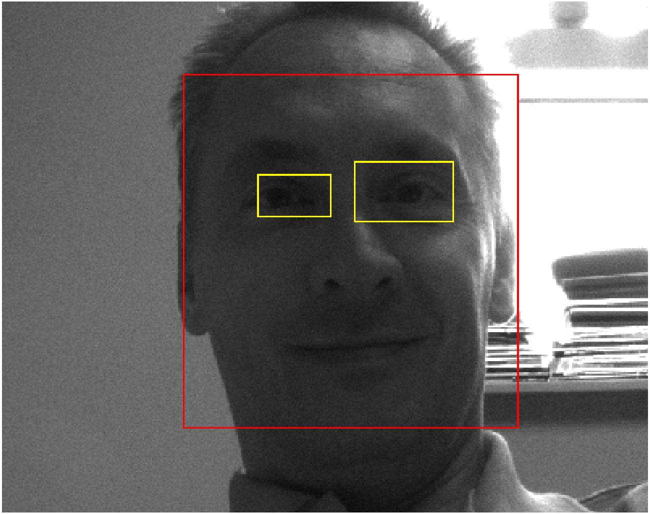
$$
\begin{aligned}
S(x, y) &= \sum_i \sum_j I(i, j), \quad 0 < i < x, \quad 0 < j < y \\
\mu &= \frac{(S(A) + S(B)) - (S(C) + S(D))}{p \times q}
\end{aligned}
$$

Applied Example



- Technique worked fairly well in applied setting

OpenCV



- We are exploring OpenCV for feature tracking

Revisiting Ellipse Fitting

- Ellipse fitting is an essential eye tracking component
- RANSAC fits many ellipses, one winner—slow
- Given n points $\{x_i, y_i\}$, fit the quadratic such that

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- We can rewrite this as

$$\begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0$$

- Then rewrite in matrix notation

$$\mathbf{XA} = \mathbf{B}$$

- And solve for $\mathbf{A}$

Revisiting Ellipse Fitting

- Ellipse fitting is an essential eye tracking component
- RANSAC fits many ellipses, one winner—slow
- Given n points $\{x_i, y_i\}$, fit the quadratic such that

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- We can rewrite this as

$$\begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0$$

- Then rewrite in matrix notation

$$\mathbf{XA} = \mathbf{B}$$

- And solve for $\mathbf{A}$

Revisiting Ellipse Fitting

- Ellipse fitting is an essential eye tracking component
- RANSAC fits many ellipses, one winner—slow
- Given n points $\{x_i, y_i\}$, fit the quadratic such that

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- We can rewrite this as

$$\begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0$$

- Then rewrite in matrix notation

$$\mathbf{XA} = \mathbf{B}$$

- And solve for **A**

Revisiting Ellipse Fitting

- Ellipse fitting is an essential eye tracking component
- RANSAC fits many ellipses, one winner—slow
- Given n points $\{x_i, y_i\}$, fit the quadratic such that

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- We can rewrite this as

$$\begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0$$

- Then rewrite in matrix notation

$$\mathbf{XA} = \mathbf{B}$$

- And solve for $\mathbf{A}$

Revisiting Ellipse Fitting

- Ellipse fitting is an essential eye tracking component
- RANSAC fits many ellipses, one winner—slow
- Given n points $\{x_i, y_i\}$, fit the quadratic such that

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- We can rewrite this as

$$\begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0$$

- Then rewrite in matrix notation

$$\mathbf{XA} = \mathbf{B}$$

- And solve for **A**

Revisiting Ellipse Fitting

- Ellipse fitting is an essential eye tracking component
- RANSAC fits many ellipses, one winner—slow
- Given n points $\{x_i, y_i\}$, fit the quadratic such that

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- We can rewrite this as

$$\begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0$$

- Then rewrite in matrix notation

$$\mathbf{XA} = \mathbf{B}$$

- And solve for $\mathbf{A}$

Ellipse definition

- Alternate representation of ellipse is

$$\frac{(x-h)^2}{r^2} + \frac{(y-k)^2}{s^2} = 1$$

- With center (h, k) and axes $(\mathbf{r}, \mathbf{s})$ of length $r = \|\mathbf{r}\|$ and $s = \|\mathbf{s}\|$ respectively, coefficients are expressed as:

$$\begin{aligned}
a &= s^2M^2 + r^2N^2 \\
b &= 2MN(s^2 - r^2) \\
c &= s^2N^2 + r^2M^2 \\
d &= -2h(s^2M^2 + r^2N^2) - 2MNk(s^2 - r^2) \\
e &= -2k(s^2N^2 + r^2M^2) - 2MNh(s^2 - r^2) \\
f &= M^2(s^2h^2 + r^2k^2) + N^2(r^2h^2 + s^2k^2) + 2MNhk(s^2 - r^2) - r^2s^2
\end{aligned}$$

- With $M = \cos(\theta)$, $N = \sin(\theta)$ for an ellipse rotated about its center by angle $\theta = \tan^{-1}(\mathbf{r}_y/\mathbf{r}_x)$

Ellipse definition

- Alternate representation of ellipse is

$$\frac{(x-h)^2}{r^2}+\frac{(y-k)^2}{s^2}=1$$

- With center (h,k) and axes $(\mathbf{r},\mathbf{s})$ of length $r=\|\mathbf{r}\|$ and $s=\|\mathbf{s}\|$ respectively, coefficients are expressed as:

$$\begin{aligned}
a &= s^2M^2+r^2N^2\\
b &= 2MN(s^2-r^2)\\
c &= s^2N^2+r^2M^2\\
d &= -2h(s^2M^2+r^2N^2)-2MNk(s^2-r^2)\\
e &= -2k(s^2N^2+r^2M^2)-2MNh(s^2-r^2)\\
f &= M^2(s^2h^2+r^2k^2)+N^2(r^2h^2+s^2k^2)+2MNhk(s^2-r^2)-r^2s^2
\end{aligned}$$

- With $M=\cos(\theta)$, $N=\sin(\theta)$ for an ellipse rotated about its center by angle $\theta=\tan^{-1}(\mathbf{r}_y/\mathbf{r}_x)$

Ellipse definition

- Alternate representation of ellipse is

$$\frac{(x-h)^2}{r^2} + \frac{(y-k)^2}{s^2} = 1$$

- With center (h, k) and axes $(\mathbf{r}, \mathbf{s})$ of length $r = \|\mathbf{r}\|$ and $s = \|\mathbf{s}\|$ respectively, coefficients are expressed as:

$$\begin{aligned}
a &= s^2M^2 + r^2N^2 \\
b &= 2MN(s^2 - r^2) \\
c &= s^2N^2 + r^2M^2 \\
d &= -2h(s^2M^2 + r^2N^2) - 2MNk(s^2 - r^2) \\
e &= -2k(s^2N^2 + r^2M^2) - 2MNh(s^2 - r^2) \\
f &= M^2(s^2h^2 + r^2k^2) + N^2(r^2h^2 + s^2k^2) + 2MNhk(s^2 - r^2) - r^2s^2
\end{aligned}$$

- With $M = \cos(\theta)$, $N = \sin(\theta)$ for an ellipse rotated about its center by angle $\theta = \tan^{-1}(\mathbf{r}_y/\mathbf{r}_x)$

Method of Least-Squares

- We can set $f = 1$ (or any non-zero arbitrary constant),

$$ax^2 + bxy + cy^2 + dx + ey = -1$$

- Then rewrite as

$$\begin{aligned} \begin{bmatrix} x^2 & xy & y^2 & x & y \end{bmatrix} \begin{bmatrix} a & b & c & d & e \end{bmatrix}^T &= -f \\ \mathbf{XA} &= \mathbf{B} \end{aligned}$$

- And solve using the pseudo-inverse of $\mathbf{X}$

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{B}$$

- With

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} x_i^2 & x_i y_i & y_i^2 & x_i & y_i \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} -1_1 & -1_2 & \dots & -1_n \end{bmatrix}^T \end{aligned}$$

- This works reasonably well, but recovering center (h, k), axes $(\mathbf{r}, \mathbf{s})$, and angle θ is problematic

Method of Least-Squares

- We can set $f = 1$ (or any non-zero arbitrary constant),

$$ax^2 + bxy + cy^2 + dx + ey = -1$$

- Then rewrite as

$$\begin{aligned} \begin{bmatrix} x^2 & xy & y^2 & x & y \end{bmatrix} \begin{bmatrix} a & b & c & d & e \end{bmatrix}^T &= -f \\ \mathbf{XA} &= \mathbf{B} \end{aligned}$$

- And solve using the pseudo-inverse of $\mathbf{X}$

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{B}$$

- With

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} x_i^2 & x_i y_i & y_i^2 & x_i & y_i \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} -1_1 & -1_2 & \dots & -1_n \end{bmatrix}^T \end{aligned}$$

- This works reasonably well, but recovering center (h, k), axes $(\mathbf{r}, \mathbf{s})$, and angle θ is problematic

Method of Least-Squares

- We can set $f = 1$ (or any non-zero arbitrary constant),

$$ax^2 + bxy + cy^2 + dx + ey = -1$$

- Then rewrite as

$$\begin{aligned} \begin{bmatrix} x^2 & xy & y^2 & x & y \end{bmatrix} \begin{bmatrix} a & b & c & d & e \end{bmatrix}^T &= -f \\ \mathbf{XA} &= \mathbf{B} \end{aligned}$$

- And solve using the pseudo-inverse of $\mathbf{X}$

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{B}$$

- With

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} x_i^2 & x_iy_i & y_i^2 & x_i & y_i \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} -1_1 & -1_2 & \dots & -1_n \end{bmatrix}^T \end{aligned}$$

- This works reasonably well, but recovering center (h, k), axes $(\mathbf{r}, \mathbf{s})$, and angle θ is problematic

Method of Least-Squares

- We can set $f = 1$ (or any non-zero arbitrary constant),

$$ax^2 + bxy + cy^2 + dx + ey = -1$$

- Then rewrite as

$$\begin{aligned} \begin{bmatrix} x^2 & xy & y^2 & x & y \end{bmatrix} \begin{bmatrix} a & b & c & d & e \end{bmatrix}^T &= -f \\ \mathbf{XA} &= \mathbf{B} \end{aligned}$$

- And solve using the pseudo-inverse of $\mathbf{X}$

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{B}$$

- With

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} x_i^2 & x_iy_i & y_i^2 & x_i & y_i \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} -1_1 & -1_2 & \dots & -1_n \end{bmatrix}^T \end{aligned}$$

- This works reasonably well, but recovering center (h, k), axes $(\mathbf{r}, \mathbf{s})$, and angle θ is problematic

Method of Least-Squares

- We can set $f = 1$ (or any non-zero arbitrary constant),

$$ax^2 + bxy + cy^2 + dx + ey = -1$$

- Then rewrite as

$$\begin{aligned} \begin{bmatrix} x^2 & xy & y^2 & x & y \end{bmatrix} \begin{bmatrix} a & b & c & d & e \end{bmatrix}^T &= -f \\ \mathbf{XA} &= \mathbf{B} \end{aligned}$$

- And solve using the pseudo-inverse of $\mathbf{X}$

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{B}$$

- With

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} x_i^2 & x_iy_i & y_i^2 & x_i & y_i \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} -1_1 & -1_2 & \dots & -1_n \end{bmatrix}^T \end{aligned}$$

- This works reasonably well, but recovering center (h, k), axes $(\mathbf{r}, \mathbf{s})$, and angle θ is problematic

Fitzgibbon et al. (1996)

- Most eye tracking papers refer to Fitzgibbon et al. (1996)
- Method exploits ellipse-specific constraint $b^2 - 4ac < 0$
- The fitting problem is reformulated as

$$\min_{\mathbf{a}} \|\mathbf{D}\mathbf{a}\|^2 \quad \text{subject to} \quad \mathbf{a}^T\mathbf{C}\mathbf{a} = 1 \tag{1}$$

where the *design matrix* $\mathbf{D}$ of size $n \times 6$,

$$\mathbf{D} = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & & & & & \\ x_i^2 & x_iy_i & y_i^2 & x_i & y_i & 1 \\ \vdots & & & & & \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1 \end{pmatrix}$$

represents the least squares minimization

Fitzgibbon et al. (1996)

- Most eye tracking papers refer to Fitzgibbon et al. (1996)
- Method exploits ellipse-specific constraint $b^2 - 4ac < 0$
- The fitting problem is reformulated as

$$\min_{\mathbf{a}} \|\mathbf{D}\mathbf{a}\|^2 \quad \text{subject to} \quad \mathbf{a}^T\mathbf{C}\mathbf{a} = 1 \tag{1}$$

where the *design matrix* $\mathbf{D}$ of size $n \times 6$,

$$\mathbf{D} = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & & & & & \\ x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \\ \vdots & & & & & \\ x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 \end{pmatrix}$$

represents the least squares minimization

Fitzgibbon et al. (1996)

- Most eye tracking papers refer to Fitzgibbon et al. (1996)
- Method exploits ellipse-specific constraint $b^2 - 4ac < 0$
- The fitting problem is reformulated as

$$\min_{\mathbf{a}} \|\mathbf{D}\mathbf{a}\|^2 \quad \text{subject to} \quad \mathbf{a}^T\mathbf{C}\mathbf{a} = 1 \tag{1}$$

where the *design matrix* $\mathbf{D}$ of size $n \times 6$,

$$\mathbf{D} = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & & & & & \\ x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \\ \vdots & & & & & \\ x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 \end{pmatrix}$$

represents the least squares minimization

Fitzgibbon et al. (1996)

- The *constraint matrix* **C** of size 6×6, expresses the constraint $b^2 - 4ac < 0$,

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- The minimization problem (1) is solved by a quadratically constrained least squares minimization
- Apply the Lagrange multipliers for the optimal solution **a**

$$\mathbf{S}\mathbf{a} = \lambda \mathbf{C}\mathbf{a} \tag{2}$$

where **S** is the *scatter matrix* of size 6×6,

$$\mathbf{S} = \mathbf{D}^T\mathbf{D}$$

Fitzgibbon et al. (1996)

- The *constraint matrix* **C** of size 6×6, expresses the constraint $b^2 - 4ac < 0$,

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- The minimization problem (1) is solved by a quadratically constrained least squares minimization
- Apply the Lagrange multipliers for the optimal solution **a**

$$\mathbf{Sa} = \lambda \mathbf{Ca} \tag{2}$$

where **S** is the *scatter matrix* of size 6×6,

$$\mathbf{S} = \mathbf{D}^T \mathbf{D}$$

Fitzgibbon et al. (1996)

- The *constraint matrix* **C** of size 6×6, expresses the constraint $b^2 - 4ac < 0$,

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- The minimization problem (1) is solved by a quadratically constrained least squares minimization
- Apply the Lagrange multipliers for the optimal solution **a**

$$\mathbf{Sa} = \lambda \mathbf{Ca} \tag{2}$$

where **S** is the *scatter matrix* of size 6×6,

$$\mathbf{S} = \mathbf{D}^T \mathbf{D}$$

Fitzgibbon et al. (1996)

- The scatter matrix **S**

$$\mathbf{S} = \begin{pmatrix} S_{x^4} & S_{x^3y} & S_{x^2y^2} & S_{x^3} & S_{x^2y} & S_{x^2} \\ S_{x^3y} & S_{x^2y^2} & S_{xy^3} & S_{x^2y} & S_{xy^2} & S_{xy} \\ S_{x^2y^2} & S_{xy^3} & S_{y^4} & S_{xy^2} & S_{y^3} & S_{y^2} \\ S_{x^3} & S_{x^2y} & S_{xy^2} & S_{x^2} & S_{xy} & S_{x} \\ S_{x^2y} & S_{xy^2} & S_{y^3} & S_{xy} & S_{y^2} & S_{y} \\ S_{x^2} & S_{xy} & S_{y^2} & S_{x} & S_{y} & S_{1} \end{pmatrix}$$

represents the least squares minimization in which the operator S denotes the sum

$$S_{x^ay^b} = \sum_{i=1}^{n} x_i^a y_i^b$$

- The system (2) is solved by using generalized eigenvectors

Fitzgibbon et al. (1996)

- The scatter matrix **S**

$$\mathbf{S} = \begin{pmatrix} S_{x^4} & S_{x^3 y} & S_{x^2 y^2} & S_{x^3} & S_{x^2 y} & S_{x^2} \\ S_{x^3 y} & S_{x^2 y^2} & S_{xy^3} & S_{x^2 y} & S_{xy^2} & S_{xy} \\ S_{x^2 y^2} & S_{xy^3} & S_{y^4} & S_{xy^2} & S_{y^3} & S_{y^2} \\ S_{x^3} & S_{x^2 y} & S_{xy^2} & S_{x^2} & S_{xy} & S_{x} \\ S_{x^2 y} & S_{xy^2} & S_{y^3} & S_{xy} & S_{y^2} & S_{y} \\ S_{x^2} & S_{xy} & S_{y^2} & S_{x} & S_{y} & S_{1} \end{pmatrix}$$

represents the least squares minimization in which the operator S denotes the sum

$$S_{x^a y^b} = \sum_{i=1}^{n} x_i^a y_i^b$$

- The system (2) is solved by using generalized eigenvectors

Fitzgibbon et al. (1996)

- Fitzgibbon et al. (1996) give a 7-line MATLAB program
- Halíř and Flusser (1998) identify several problems:
  - the matrix **C** is singular
  - the matrix **S** is also nearly singular (and is singular if all data points lie exactly on an ellipse)
  - the computation of eigenvectors (2) is numerically unstable and can produce wrong results

Fitzgibbon et al. (1996)

- Fitzgibbon et al. (1996) give a 7-line MATLAB program
- Halíř and Flusser (1998) identify several problems:
  - the matrix **C** is singular
  - the matrix **S** is also nearly singular (and is singular if all data points lie exactly on an ellipse)
  - the computation of eigenvectors (2) is numerically unstable and can produce wrong results

Fitzgibbon et al. (1996)

- Fitzgibbon et al. (1996) give a 7-line MATLAB program
- Halíř and Flusser (1998) identify several problems:
  - the matrix **C** is singular
  - the matrix **S** is also nearly singular (and is singular if all data points lie exactly on an ellipse)
  - the computation of eigenvectors (2) is numerically unstable and can produce wrong results

Fitzgibbon et al. (1996)

- Fitzgibbon et al. (1996) give a 7-line MATLAB program
- Halíř and Flusser (1998) identify several problems:
  - the matrix **C** is singular
  - the matrix **S** is also nearly singular (and is singular if all data points lie exactly on an ellipse)
  - the computation of eigenvectors (2) is numerically unstable and can produce wrong results

Fitzgibbon et al. (1996)

- Fitzgibbon et al. (1996) give a 7-line MATLAB program
- Halíř and Flusser (1998) identify several problems:
  - the matrix **C** is singular
  - the matrix **S** is also nearly singular (and is singular if all data points lie exactly on an ellipse)
  - the computation of eigenvectors (2) is numerically unstable and can produce wrong results

Halíř and Flusser (1998)

- Halíř and Flusser (1998) break up the design matrix **D** into its quadratic and linear components

$$\mathbf{D} = (\mathbf{D}_1 \mid \mathbf{D}_2)$$

where

$$\mathbf{D}_1 = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 \\ \vdots & & \\ x_i^2 & x_i y_i & y_i^2 \\ \vdots & & \\ x_n^2 & x_n y_n & y_n^2 \end{pmatrix} \quad \text{and} \quad \mathbf{D}_2 = \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & & \\ x_i & y_i & 1 \\ \vdots & & \\ x_n & y_n & 1 \end{pmatrix}$$

Halíř and Flusser (1998)

- Next, the scatter matrix **S** is split as follows

$$\mathbf{S} = \left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \quad \text{with} \quad \begin{aligned} \mathbf{S}_1 &= \mathbf{D}_1^T \mathbf{D}_1 \\ \mathbf{S}_2 &= \mathbf{D}_1^T \mathbf{D}_2 \\ \mathbf{S}_3 &= \mathbf{D}_2^T \mathbf{D}_2 \end{aligned}$$

- The constraint matrix **C** is expressed similarly

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \quad \text{where} \quad \mathbf{C}_1 = \begin{pmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ -2 & 0 & 0 \end{pmatrix}$$

- The vector of coefficients is split into

$$\mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{pmatrix} \quad \text{where} \quad \mathbf{a}_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \text{and} \quad \mathbf{a}_2 = \begin{pmatrix} d \\ e \\ f \end{pmatrix}$$

Halíř and Flusser (1998)

- Next, the scatter matrix **S** is split as follows

$$\mathbf{S} = \left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \quad \text{with} \quad \begin{aligned} \mathbf{S}_1 &= \mathbf{D}_1^T \mathbf{D}_1 \\ \mathbf{S}_2 &= \mathbf{D}_1^T \mathbf{D}_2 \\ \mathbf{S}_3 &= \mathbf{D}_2^T \mathbf{D}_2 \end{aligned}$$

- The constraint matrix **C** is expressed similarly

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \quad \text{where} \quad \mathbf{C}_1 = \begin{pmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ -2 & 0 & 0 \end{pmatrix}$$

- The vector of coefficients is split into

$$\mathbf{a} = \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right) \quad \text{where} \quad \mathbf{a}_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \text{and} \quad \mathbf{a}_2 = \begin{pmatrix} d \\ e \\ f \end{pmatrix}$$

Halíř and Flusser (1998)

- Next, the scatter matrix $\mathbf{S}$ is split as follows

$$\mathbf{S} = \left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \quad \text{with} \quad \begin{aligned} \mathbf{S}_1 &= \mathbf{D}_1^T \mathbf{D}_1 \\ \mathbf{S}_2 &= \mathbf{D}_1^T \mathbf{D}_2 \\ \mathbf{S}_3 &= \mathbf{D}_2^T \mathbf{D}_2 \end{aligned}$$

- The constraint matrix $\mathbf{C}$ is expressed similarly

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \quad \text{where} \quad \mathbf{C}_1 = \begin{pmatrix} 0 & 0 & -2 \\ 0 & 1 & 0 \\ -2 & 0 & 0 \end{pmatrix}$$

- The vector of coefficients is split into

$$\mathbf{a} = \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right) \quad \text{where} \quad \mathbf{a}_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \text{and} \quad \mathbf{a}_2 = \begin{pmatrix} d \\ e \\ f \end{pmatrix}$$

Halíř and Flusser (1998)

- The optimal solution (2) is rewritten as

$$\left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right)$$

- which is equivalent to the following two equations

$$\mathbf{S}_1\mathbf{a}_1 + \mathbf{S}_2\mathbf{a}_2 = \lambda\mathbf{C}_1\mathbf{a}_1 \quad (3)$$

$$\mathbf{S}_2^T\mathbf{a}_1 + \mathbf{S}_3\mathbf{a}_2 = 0 \quad (4)$$

- From (4), solve for $\mathbf{a}_2$

$$\mathbf{a}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^T\mathbf{a}_1 \quad (5)$$

- and plug (5) into (3) to yield

$$\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{C}_1\mathbf{a}_1 \quad (6)$$

Halíř and Flusser (1998)

- The optimal solution (2) is rewritten as

$$\left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right)$$

- which is equivalent to the following two equations

$$\mathbf{S}_1\mathbf{a}_1 + \mathbf{S}_2\mathbf{a}_2 = \lambda\mathbf{C}_1\mathbf{a}_1 \quad (3)$$

$$\mathbf{S}_2^T\mathbf{a}_1 + \mathbf{S}_3\mathbf{a}_2 = 0 \quad (4)$$

- From (4), solve for $\mathbf{a}_2$

$$\mathbf{a}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^T\mathbf{a}_1 \quad (5)$$

- and plug (5) into (3) to yield

$$\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{C}_1\mathbf{a}_1 \quad (6)$$

Halíř and Flusser (1998)

- The optimal solution (2) is rewritten as

$$\left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right)$$

- which is equivalent to the following two equations

$$\mathbf{S}_1\mathbf{a}_1 + \mathbf{S}_2\mathbf{a}_2 = \lambda\mathbf{C}_1\mathbf{a}_1 \quad (3)$$

$$\mathbf{S}_2^T\mathbf{a}_1 + \mathbf{S}_3\mathbf{a}_2 = 0 \quad (4)$$

- From (4), solve for $\mathbf{a}_2$

$$\mathbf{a}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^T\mathbf{a}_1 \quad (5)$$

- and plug (5) into (3) to yield

$$\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{C}_1\mathbf{a}_1 \quad (6)$$

Halíř and Flusser (1998)

- The optimal solution (2) is rewritten as

$$\left(\begin{array}{c|c} \mathbf{S}_1 & \mathbf{S}_2 \\ \hline \mathbf{S}_2^T & \mathbf{S}_3 \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right) = \left(\begin{array}{c|c} \mathbf{C}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right) \cdot \left(\begin{array}{c} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \end{array}\right)$$

- which is equivalent to the following two equations

$$\begin{aligned} \mathbf{S}_1\mathbf{a}_1 + \mathbf{S}_2\mathbf{a}_2 &= \lambda\mathbf{C}_1\mathbf{a}_1 && (3) \\ \mathbf{S}_2^T\mathbf{a}_1 + \mathbf{S}_3\mathbf{a}_2 &= 0 && (4) \end{aligned}$$

- From (4), solve for $\mathbf{a}_2$

$$\mathbf{a}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^T\mathbf{a}_1 \qquad (5)$$

- and plug (5) into (3) to yield

$$\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{C}_1\mathbf{a}_1 \qquad (6)$$

Halíř and Flusser (1998)

- Because $\mathbf{C}_1$ is regular, (6) can be rewritten as

$$\mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{a}_1 \qquad (7)$$

- where $\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$ is the *reduced scatter matrix* of size 3×3
- For the optimal fit, it is enough to find the appropriate eigenvector $\mathbf{a}_1$ of matrix $\mathbf{M}$
- Halíř and Flusser (1998) also provide MATLAB code

Halíř and Flusser (1998)

- Because $\mathbf{C}_1$ is regular, (6) can be rewritten as

$$\mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{a}_1 \qquad (7)$$

- where $\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$ is the *reduced scatter matrix* of size 3×3
- For the optimal fit, it is enough to find the appropriate eigenvector $\mathbf{a}_1$ of matrix $\mathbf{M}$
- Halíř and Flusser (1998) also provide MATLAB code

Halíř and Flusser (1998)

- Because $\mathbf{C}_1$ is regular, (6) can be rewritten as

$$\mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{a}_1 \tag{7}$$

- where $\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$ is the *reduced scatter matrix* of size 3×3
- For the optimal fit, it is enough to find the appropriate eigenvector $\mathbf{a}_1$ of matrix $\mathbf{M}$
- Halíř and Flusser (1998) also provide MATLAB code

Halíř and Flusser (1998)

- Because $\mathbf{C}_1$ is regular, (6) can be rewritten as

$$\mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)\mathbf{a}_1 = \lambda\mathbf{a}_1 \quad (7)$$

- where $\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$ is the *reduced scatter matrix* of size 3×3
- For the optimal fit, it is enough to find the appropriate eigenvector $\mathbf{a}_1$ of matrix $\mathbf{M}$
- Halíř and Flusser (1998) also provide MATLAB code

Further enhancements

- For improved numerical stability, Paul and Zsombor-Murray (2004) suggest first translating all data points then scaling isotropically to ensure $\sqrt{2}$ root mean square distance of the points to the origin, i.e., using the transformation matrix $\mathbf{T}$

$$\mathbf{T} = \begin{pmatrix} s & 0 & -s\overline{x} \\ 0 & s & -s\overline{y} \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$s = \frac{n\sqrt{2}}{\sum_{i=1}^{n} \sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2}}$$

Further Enhancements

- After fitting, Paul and Zsombor-Murray (2004) suggest transforming data back to the original position and scaling via

$$\mathbf{K}^* = \mathbf{T}^T\mathbf{K}\mathbf{T}$$

where $\mathbf{K}$ is the conic matrix

$$\mathbf{K} = \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix}$$

- Paul and Zsombor-Murray also provide MATLAB code

Further Enhancements

- After fitting, Paul and Zsombor-Murray (2004) suggest transforming data back to the original position and scaling via

$$\mathbf{K}^* = \mathbf{T}^T \mathbf{K} \mathbf{T}$$

  where **K** is the conic matrix

$$\mathbf{K} = \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix}$$

- Paul and Zsombor-Murray also provide MATLAB code

Further Enhancements

- Harker et al. (2008) suggest a similar approach with scaling factor m yielding so-called *mean-free* coordinates

$$m = \sqrt{\frac{2n}{\sum_{i=1}^{n} (x_i - \overline{x})^2 + (y_i - \overline{y})^2}}$$

- They, however, do not provide MATLAB code

Further Enhancements

- Harker et al. (2008) suggest a similar approach with scaling factor m yielding so-called *mean-free* coordinates

$$m = \sqrt{\frac{2n}{\sum_{i=1}^{n} (x_i - \overline{x})^2 + (y_i - \overline{y})^2}}$$

- They, however, do not provide MATLAB code

Implementation Notes

- Eventual solution obtained “officially” from the IEEE PAMI paper by Fitzgibbon et al. (1999), but practically speaking from their web page:

`http://research.microsoft.com/en-us/um/people/awf/ellipse/`

  containing, you guessed it, MATLAB code
- They also normalize the points by shifting to the centroid and scaling, i.e.,

$$\mathbf{T} = \begin{pmatrix} 1/\mathbf{s}_x & 0 & -\overline{x} \\ 0 & 1/\mathbf{s}_y & -\overline{y} \\ 0 & 0 & 0 \end{pmatrix}$$

  where

$$\mathbf{s} = \begin{bmatrix} \frac{\max(x_i) - \min(x_i)}{2} & \frac{\max(y_i) - \min(y_i)}{2} \end{bmatrix}$$

Implementation Notes

- Eventual solution obtained “officially” from the IEEE PAMI paper by Fitzgibbon et al. (1999), but practically speaking from their web page:
`http://research.microsoft.com/en-us/um/people/awf/ellipse/`
containing, you guessed it, MATLAB code
- They also normalize the points by shifting to the centroid and scaling, i.e.,

$$\mathbf{T} = \begin{pmatrix} 1/\mathbf{s}_x & 0 & -\overline{x} \\ 0 & 1/\mathbf{s}_y & -\overline{y} \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$\mathbf{s} = \left[\frac{\max(x_i) - \min(x_i)}{2} \quad \frac{\max(y_i) - \min(y_i)}{2} \right]$$

Fitzgibbon et al. (1999)

- What's really useful about Fitzgibbon et al.'s MATLAB code is that it includes the steps to "unnormalize" the fitted ellipse and to recover the ellipse parameters (h, k), (r, s), and θ
- Only problem is this line:

```
[evec_x, eval_x] = eig(inv(tmpD) * (tmpA - tmpB*tmpE));
```

which is the general solution to the eigensystem represented by

$$\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$$

which happens to be non-symmetric
- I can see why MATLAB is popular...

Fitzgibbon et al. (1999)

- What's really useful about Fitzgibbon et al.'s MATLAB code is that it includes the steps to "unnormalize" the fitted ellipse and to recover the ellipse parameters (h, k), (r, s), and θ
- Only problem is this line:

```
[evec_x, eval_x] = eig(inv(tmpD) * (tmpA - tmpB*tmpE));
```

which is the general solution to the eigensystem represented by

$$\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$$

which happens to be non-symmetric

- I can see why MATLAB is popular...

Fitzgibbon et al. (1999)

- What's really useful about Fitzgibbon et al.'s MATLAB code is that it includes the steps to "unnormalize" the fitted ellipse and to recover the ellipse parameters (h, k), (r, s), and θ
- Only problem is this line:

```
[evec_x, eval_x] = eig(inv(tmpD) * (tmpA - tmpB*tmpE));
```

which is the general solution to the eigensystem represented by

$$\mathbf{M} = \mathbf{C}_1^{-1}\left(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T\right)$$

which happens to be non-symmetric

- I can see why MATLAB is popular...

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)
- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)

- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration

- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)
- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)
- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

> *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)

- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)
- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)
- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

Obtaining eigenvectors from non-symmetric matrix

- My go-to book for this sort of thing states:

  > *We consider the problem of finding* all *eigenvectors of a nonsymmetric matrix as lying beyond the scope of this book.*

  (Numerical Recipes in C, 2nd ed., Press et al., 1992)
- What's involved?
  1. balancing the matrix
  2. reducing to Hessenberg form
  3. applying the QR method for a non-singular square matrix
  4. solving for eigenvectors via inverse iteration
- Which was really more than what I wanted to know; sometimes you just want to call a math library function

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`

- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`

- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

- Luckily, we have Google!
- It came up with the Template Numerical Toolkit, made freely available by NIST here:

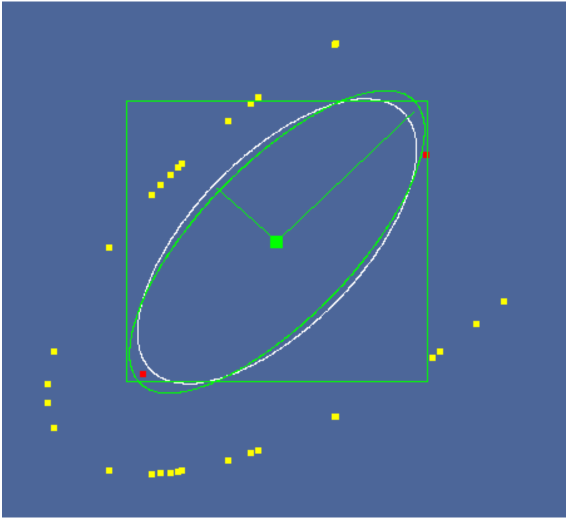
  `http://math.nist.gov/tnt/`
- This has both TNT, templated C++ arrays, as well as JAMA
- JAMA/C++ is a translation of the Java Matrix Library, developed by the Mathworks and NIST, into C++ classical matrix linear algebra, and includes:
  - LU
  - QR
  - SVD
  - Cholesky
  - Eigenvalue solver
- Java Matrix Library? Did I read that correctly?

TNT and JAMA

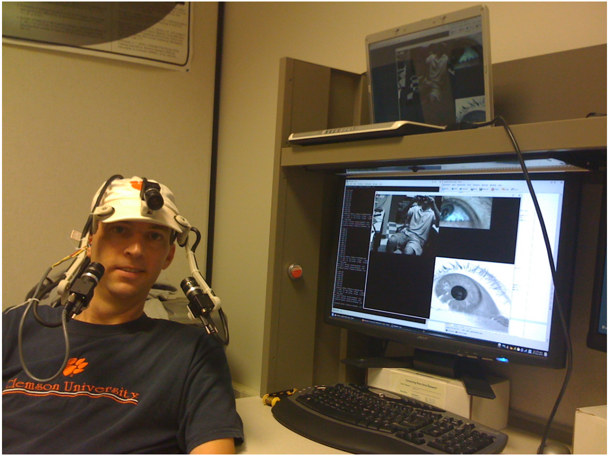
- The last piece of the puzzle I needed was the C++ equivalent to the `eig()` MATLAB call:

```
JAMA::Eigenvalue<double> eigen(TNT_M);
eigen.getV(TNT_V);
eigen.getRealEigenvalues(TNT_E);
```

Demo



On-line NIR Spectrum Wearable Tracker



Parts List

Vendor	Part No.	Desc.	Quant.	Unit Cost
Point Grey	FL2-03S2C	Scene Camera (c)	1	$695
Point Grey	FL2-03S2M	Eye Cameras (b/w)	2	$695
Point Grey	FWB-EC-2PORT	1394 ExpressCard	1	$80
Point Grey	HF12.5HA-1B	Eye Camera Lenses	2	$170
Point Grey	DF6HA-1B	Scene Camera Lens	1	$190
Point Grey	ACC-01-2012	1394b Cables	3	$150
Point Grey	FW-HUB-5PORT	1394b Hub	1	$139
UKA Optics	IR83M25.5	IR filter	2	$185
QuickParts	Custom Order	Plastic Mounts		$476
Various		LED, misc. parts		$50
			Total	$4180

Courtesy of Lafayette Instruments, Inc.

Current Features

- On-line, real-time processing
- IEEE 1394b (`libraw1394, libdc1394`)
  - DMA access to cameras
  - Format-7 allows specification of hardware ROI
- Video encoding/decoding (`ffmpeg`)
- Direct ellipse fitting (Fitzgibbon et al., 1999)

Current Features

- On-line, real-time processing
- IEEE 1394b (`libraw1394, libdc1394`)
  - DMA access to cameras
  - Format-7 allows specification of hardware ROI
- Video encoding/decoding (`ffmpeg`)
- Direct ellipse fitting (Fitzgibbon et al., 1999)

Current Features

- On-line, real-time processing
- IEEE 1394b (`libraw1394, libdc1394`)
  - DMA access to cameras
  - Format-7 allows specification of hardware ROI
- Video encoding/decoding (`ffmpeg`)
- Direct ellipse fitting (Fitzgibbon et al., 1999)

Current Features

- On-line, real-time processing
- IEEE 1394b (`libraw1394, libdc1394`)
  - DMA access to cameras
  - Format-7 allows specification of hardware ROI
- Video encoding/decoding (`ffmpeg`)
- Direct ellipse fitting (Fitzgibbon et al., 1999)

Current Features

- On-line, real-time processing
- IEEE 1394b (`libraw1394, libdc1394`)
  - DMA access to cameras
  - Format-7 allows specification of hardware ROI
- Video encoding/decoding (`ffmpeg`)
- Direct ellipse fitting (Fitzgibbon et al., 1999)

Current Features

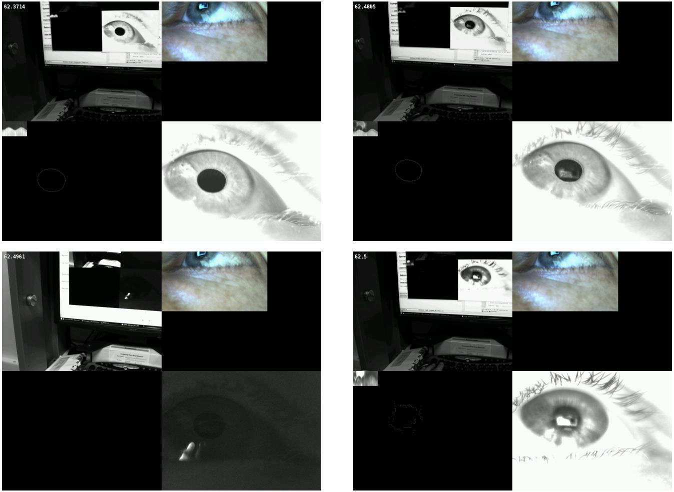
- On-line, real-time processing
- IEEE 1394b (`libraw1394, libdc1394`)
  - DMA access to cameras
  - Format-7 allows specification of hardware ROI
- Video encoding/decoding (`ffmpeg`)
- Direct ellipse fitting (Fitzgibbon et al., 1999)

Why NIR?

IR filter | no IR filter

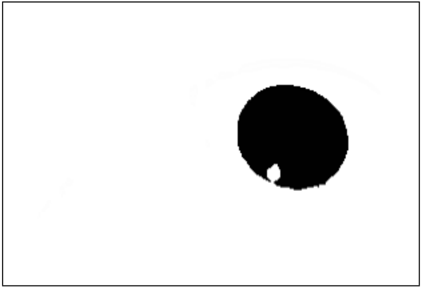
NIR LED

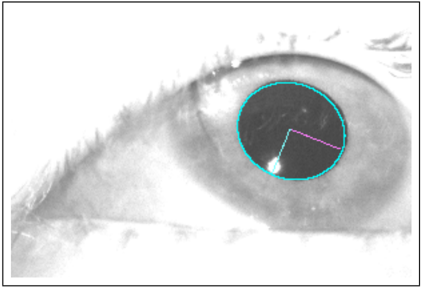
no NIR LED



NIR

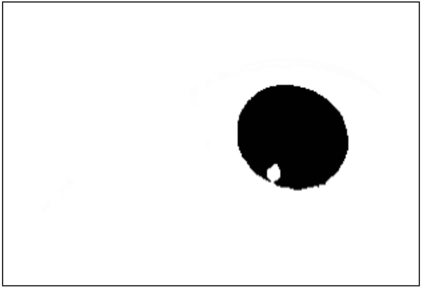
- NIR illumination: 940 nm LED, radiant power output 16 mW
- Simplifies pupil detection (thresholding)
- Allows frame-differencing method as proposed by Ebisawa and Satoh (1993), popularized by C. Morimoto et al. (2000)

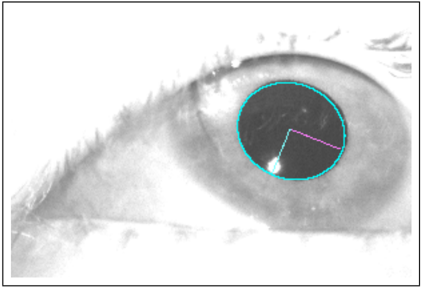




NIR

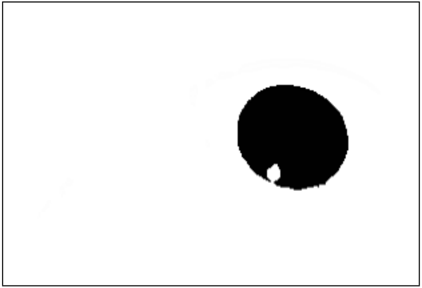
- NIR illumination: 940 nm LED, radiant power output 16 mW
- Simplifies pupil detection (thresholding)
- Allows frame-differencing method as proposed by Ebisawa and Satoh (1993), popularized by C. Morimoto et al. (2000)

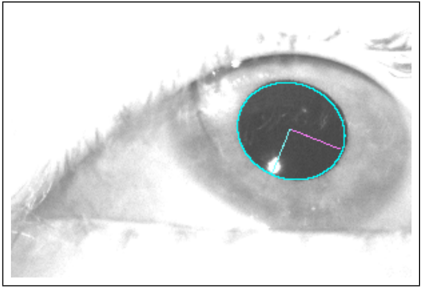




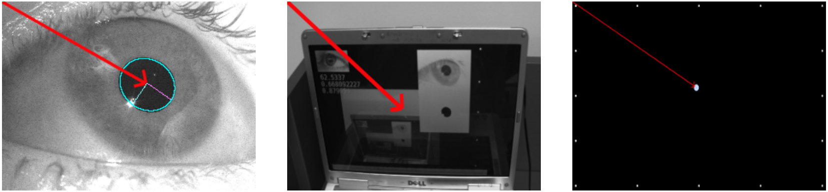
NIR

- NIR illumination: 940 nm LED, radiant power output 16 mW
- Simplifies pupil detection (thresholding)
- Allows frame-differencing method as proposed by Ebisawa and Satoh (1993), popularized by C. Morimoto et al. (2000)



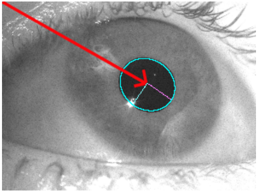


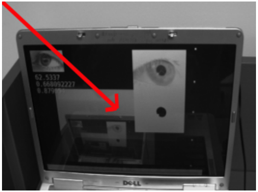
Calibration

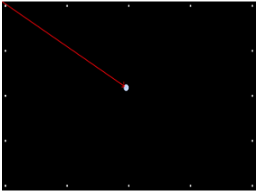


- Calibration relies on image frame origin
  (we should really be using corneal reflection)
- Using laptop screen to display calibration dots

Calibration



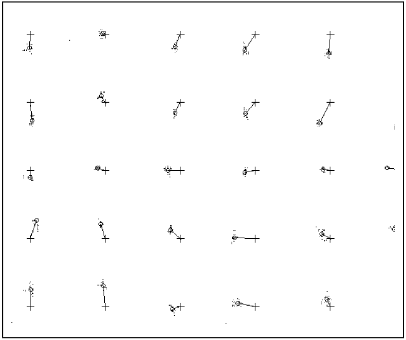


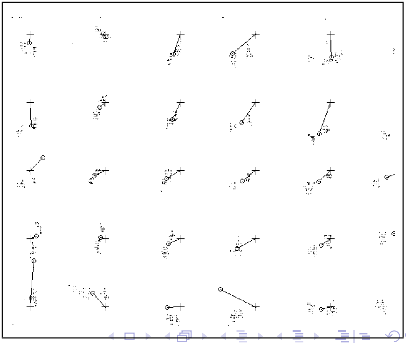


- Calibration relies on image frame origin
  (we should really be using corneal reflection)
- Using laptop screen to display calibration dots

Summary

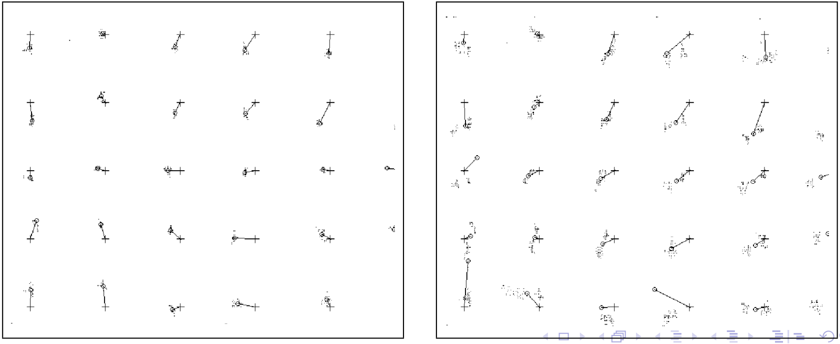
- Keep in mind importance of IR light
- Watch for spurious reflections (mascara, spectacles, etc.)
- As the late Prof. Stark said, “Calibrate, calibrate, calibrate!”
  - technology has improved since Larry’s time, but . . .
  - . . . still relevant if you want to measure “slippage”





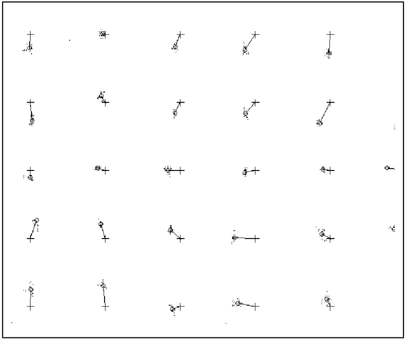
Summary

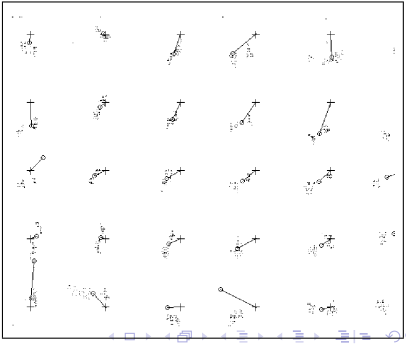
- Keep in mind importance of IR light
- Watch for spurious reflections (mascara, spectacles, etc.)
- As the late Prof. Stark said, “Calibrate, calibrate, calibrate!”
    - technology has improved since Larry's time, but . . .
    - . . . still relevant if you want to measure “slippage”



Summary

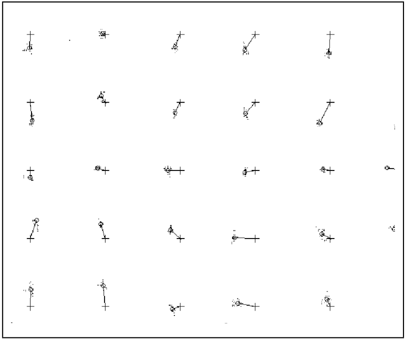
- Keep in mind importance of IR light
- Watch for spurious reflections (mascara, spectacles, etc.)
- As the late Prof. Stark said, “Calibrate, calibrate, calibrate!”
  - technology has improved since Larry’s time, but ...
  - ... still relevant if you want to measure “slippage”

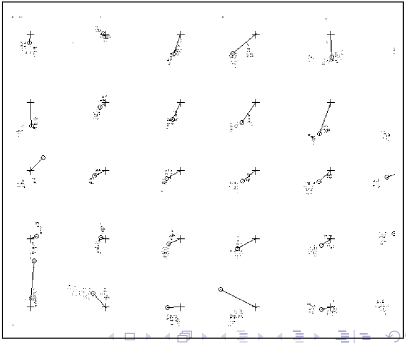




Summary

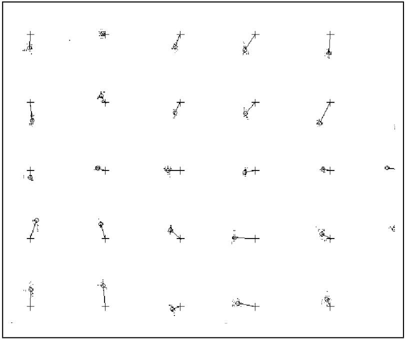
- Keep in mind importance of IR light
- Watch for spurious reflections (mascara, spectacles, etc.)
- As the late Prof. Stark said, “Calibrate, calibrate, calibrate!”
  - technology has improved since Larry’s time, but . . .
  - . . . still relevant if you want to measure “slippage”

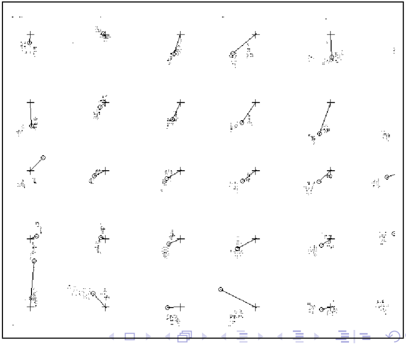




Summary

- Keep in mind importance of IR light
- Watch for spurious reflections (mascara, spectacles, etc.)
- As the late Prof. Stark said, “Calibrate, calibrate, calibrate!”
    - technology has improved since Larry’s time, but . . .
    - . . . still relevant if you want to measure “slippage”





To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

To DIY or not to DIY?

- If you want to build it yourself, use good cameras, e.g.,
  - IEEE 1394b Digital Cameras support 80+ fps with hardware ROI (format7)
  - e.g., Flea2 cameras from Point Grey Research
  - be careful about built-in IR filter—b/w Flea2s don't have one, color Flea2s do
  - choose correct lens FOV, e.g., we have
    - 6mm (wide angle for scene)
    - 12.5mm (medium range)
    - 25mm (macro-like for close range)
  - IR lens filters seem difficult to obtain, but not impossible
  - PGR cameras also come with powered "pigtail" that let you easily connect IR LEDs

Comments, Questions, Discussion?

Selected References I

Babcock, J. S., & Pelz, J. B. (2004). Building a Lightweight Eyetracking Headgear. In *ETRA '04: Proceedings of the 2004 Symposium on Eye Tracking Research & Applications* (pp. 109–114). San Antonio, TX.

Borgefors, G. (1988). Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *10*(6), 849–865.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*, 679-698.

Selected References II

Crane, H. D. (1994). The Purkinje Image Eyetracker, Image Stabilization, and Related Forms of Stimulus Manipulation. In D. H. Kelly (Ed.), *Visual Science and Engineering: Models and Applications* (p. 13-89). New York, NY: Marcel Dekker, Inc.

Crow, F. C. (1984). Summed-area tables for texture mapping. In *Siggraph '84: Proceedings of the 11th annual conference on computer graphics and interactive techniques* (pp. 207–212). New York, NY: ACM.

Ebisawa, Y., & Satoh, S. (1993). Effectiveness of pupil area detection technique using two light sources and image difference method. In *Proc. 15th IEEE Int. Conf. on Medicine and Biology Society* (pp. 1268–1269).

Selected References III

Fitzgibbon, A. W., Pilu, M., & Fisher, R. B. (1996, January). *Direct Least Square Fitting of Ellipses* (Tech. Rep. No. DAIRP-794). Department of Artificial Intelligence, University of Edinburgh.

Fitzgibbon, A. W., Pilu, M., & Fisher, R. B. (1999, May). Direct Least Square Fitting of Ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *21*(5), 476–480.

Halíř, R., & Flusser, J. (1998, 9-13 February). *Numerically Stable Direct Least Squares Fitting of Ellipses.*

Harker, M., Paul, O., & Zsombor-Murray, P. (2008). Direct type-specific conic fitting and eigenvalue bias correction. *Image and Vision Computing*, *26*, 372–381.

Selected References IV

Hennessey, C. (2008). *Point-of-Gaze Estimation in Three Dimensions*. Unpublished doctoral dissertation, University of British Columbia, Vancouver, BC, Canada.

Lancaster, P., & Šalkauskas, K. (1986). *Curve and Surface Fitting: An Introduction*. San Diego, CA: Academic Press.

Li, D., Babcock, J., & Parkhurst, D. J. (2006). openEyes: A Low-Cost Head-Mounted Eye-Tracking Solution. In *ETRA '06: Proceedings of the 2006 Symposium on Eye Tracking Research & Applications.* San Diego, CA.

Li, D., & Parkhurst, D. (2006, 4-5 September). Open-Source Software for Real-Time Visible-Spectrum Eye Tracking. In *Conference on Communication by Gaze Interaction.* Turin, Italy.

Selected References V

Menache, A. (2000). *Understanding Motion Capture for Computer Animation and Video Games*. San Diego, CA: Morgan Kaufmann, (Academic Press).

Morimoto, C., Koons, D., Amir, A., & Flickner, M. (2000). Pupil detection and tracking using multiple light sources. *Image and Vision Computing*, *18*(4), 331–336.

Morimoto, C. H., & Mimica, M. R. M. (2005, April). Eye Gaze Tracking Techniques for Interactive Applications. *Computer Vision and Image Understanding*, *98*, 4–24.

Paolini, M. (2006). *Apple Pro Training Series: Shake 4*. Berkeley, CA: Peachpit Press.

Paul, O., & Zsombor-Murray, P. (2004, July). Direct and specific least-square fitting of hyperbolæand ellipses. *Journal of Electronic Imaging*, *13*(3), 492–503.

Selected References VI

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing* (2nd ed.). Cambridge: Cambridge University Press.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*(2), 137–154.