Rapid Facial Feature Detection in iOS

Instructor - Simon Lucey

16-423 - Designing Computer Vision Apps
Today

• Facial Feature Detection in iOS.
• Active Appearance Models.
Face Detection
Face Detection in iOS

• In iOS face detection comes built in, and can be performed much more efficiently than standard OpenCV.
• Utilizes the QuartzCore and CoreImage frameworks within the project.
Face Detection in iOS

- In iOS face detection comes built in, and can be performed much more efficiently than standard OpenCV.
- Utilizes the **QuartzCore** and **CoreImage** frameworks within the project.
Facial Feature Detection
A `CIFaceFeature` object describes a face detected in a still or video image. Its properties provide information about the face’s eyes and mouth. A face object in a video can also have properties that track its location over time—tracking ID and frame count.
Locating Faces

bounds Property
hasFaceAngle Property
faceAngle Property

Identifying Facial Features

hasLeftEyePosition Property
hasRightEyePosition Property
hasMouthPosition Property
leftEyePosition Property
rightEyePosition Property
mouthPosition Property
hasSmile Property
leftEyeClosed Property
rightEyeClosed Property
CIFaceFeature Example

- We are now going to demonstrate a simple example of face detection in iOS.
- On your browser please go to the address,

  https://github.com/slucey-cs-cmu-edu/CIFaceFeature_Lena

- Or better yet, if you have git installed you can type from the command line.

  $ git clone https://github.com/slucey-cs-cmu-edu/CIFaceFeature_Lena.git
// 3. Load the image and display
UIImage *image = [UIImage imageNamed:@"lena.png"]; 
if(image != nil) imageView_.image = image; // Display the image if it is there....
else cout << "Cannot read in the file" << endl;

// 4. Initialize the face detector and its options
NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow }; 
CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];

// 5. Apply the image through the face detector
NSArray *features = [faceDetector featuresInImage:[CIImage imageWithCGImage: [image CGImage]]];

// 6. Loop through each detected face
for(CIFaceFeature* faceFeature in features) {

    // Check facial features

    // Smile
    if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
    else cout << "Lena is Not Smiling" << endl;

    // Head angle
    if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
    else cout << "Lena's head is not at an angle" << endl;

    // Left Eye location
    CGPoint lEye = faceFeature.leftEyePosition;
    if(faceFeature.hasLeftEyePosition)
        cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
    else
        cout << "Lena's left eye is occluded!!" << endl;
}
// 3. Load the image and display
UIImage *image = [UIImage imageNamed:@"lena.png"];
if(image != nil) imageView_.image = image; // Display the image if it is there....
else cout << "Cannot read in the file" << endl;

// 4. Initialize the face detector and its options
NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow };
CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];

// 5. Apply the image through the face detector
NSArray *features = [faceDetector featuresInImage:[CIImage imageWithCGImage: [image CGImage]]];

// 6. Loop through each detected face
for(CIFaceFeature* faceFeature in features) {
    // Check facial features
    // Smile
    if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
    else cout << "Lena is Not Smiling" << endl;

    // Head angle
    if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
    else cout << "Lena's head is not at an angle" << endl;

    // Left Eye location
    CGPoint lEye = faceFeature.leftEyePosition;
    if(faceFeature.hasLeftEyePosition)
        cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
    else
        cout << "Lena's left eye is occluded!!" << endl;
}
// 3. Load the image and display
UI闩mage *image = [UIImage imageNamed:@"lena.png"];
if(image != nil) imageView_.image = image; // Display the image if it is there....
else cout << "Cannot read in the file" << endl;

// 4. Initialize the face detector and its options
NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow };
CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];

// 5. Apply the image through the face detector
NSArray *features = [faceDetector featuresInImage:[CII mage imageWithCGImage: [image CGImage]]];

// 6. Loop through each detected face
for(CIFaceFeature* faceFeature in features) {
    // Check facial features
    // Smile
    if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
    else cout << "Lena is Not Smiling" << endl;

    // Head angle
    if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
    else cout << "Lena's head is not at an angle" << endl;

    // Left Eye location
    CGPoint lEye = faceFeature.leftEyePosition;
    if(faceFeature.hasLeftEyePosition)
        cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
    else
        cout << "Lena's left eye is occluded!!" << endl;
}
// 6. Loop through each detected face
for(CIFaceFeature* faceFeature in features) {

    // Check facial features

    // Smile
    if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
    else cout << "Lena is Not Smiling" << endl;

    // Head angle
    if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
    else cout << "Lena's head is not at an angle" << endl;

    // Left Eye location
    CGPoint lEye = faceFeature.leftEyePosition;
    if(faceFeature.hasLeftEyePosition)
        cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
    else
        cout << "Lena's left eye is occluded!!" << endl;
}
CIFaceFeature Example
Smerk and GPUImage

- Recently, an extension to GPUImage was proposed to allow for the utilization of iOS face detection within iOS.
- Called “Smerk” - GitHub project page can be found at: https://github.com/MattFoley/Smerk

```swift
// Set the face detector to be used
detector_ = [[SMKDetectionCamera
    AVCaptureDevicePositionFront]
    for
    [detector_ setOutputImageOrientation:
cameraView_.fillMode = kGPUImageFillMode]
    [detector_ addTarget:cameraView_]

// Important: add as a subview
[self.view addSubview:cameraView_]

// Setup the face box view
[self setupFaceTrackingViews];
[self calculateTransformations];

// Set the block for running face
[detection beginDetecting:kFaceFeature
    codeTypes:@[AV
    withDetectionBlock:^(
        // Check if the kFace
        if (detectionType ==
            self updateFace
        )
    )
];

// Finally start the camera
[detection startCameraCapture];
```
Smerk and GPUImage

- Recently, an extension to GPUImage was proposed to allow for the utilization of iOS face detection within iOS.
- Called “Smerk” - GitHub project page can be found at: https://github.com/MattFoley/Smerk
Smerk Example

• We are now going to demonstrate how we can perform real-time face tracking through GPUImage.
• On your browser please go to the address,

https://github.com/slucey-cs-cmu-edu/Smerk_Example

• Or better yet, if you have git installed you can type from the command line.

$ git clone https://github.com/slucey-cs-cmu-edu/Smerk_Example.git
```
// ViewController.m
// Smerk_Example
//
// Created by Simon Lucey on 11/30/15.
// Copyright © 2015 CMU_16432. All rights reserved.

#import "ViewController.h"
#import "SMKDetectionCamera.h"
#import <GPUImage/GPUImage.h>

@interface ViewController () {
    // Setup the view (this time using GPUImageView)
    GPUImageView *cameraView_
    SMKDetectionCamera *detector_; // Detector that scans
    UIView *faceFeatureTrackingView_; // View for showing
    CGAffineTransform cameraOutputToPreviewFrameTransform;
    CGAffineTransform portraitRotationTransform_;  // transform with
    CGAffineTransform texelToPixelTransform_;  
}
@end

@implementation ViewController
```
Smerk Example

```cpp
// ViewController.m
// Smerk_Example
//
// Created by Simon Lucey on 11/30/15.
// Copyright © 2015 CMU_16432. All rights reserved.

#import "ViewController.h"
#import "SMKDetectionCamera.h"
#import <GPUImage/GPUImage.h>

@interface ViewController () {
    GPUImageView *cameraView; // Detector that shows
    SMKDetectionCamera *detector_; // View for showing
    UIView *faceFeatureTrackingView; // Preview frame transformation
    CGAffineTransform cameraOutputToPreviewFrameTransform;
    CGAffineTransform portraitRotationTransform;
    CGAffineTransform texelToPixelTransform;
}
@end

@implementation ViewController

// Setup the view (this time using GPUImage)
```
// ViewController.m
// Smerk_Example
//
// Created by Simon Lucey on 11/30/15.
// Copyright © 2015 CMU_16432. All rights reserved.
//
#import "ViewController.h"
#import "SMKDetectionCamera.h"
#import <GPUImage/GPUImage.h>

@interface ViewController () {
    // Setup the view (this time using GPUImageView)
    GPUImageView *cameraView_;
    SMKDetectionCamera *detector_; // Detector that should be used
    UIView *faceFeatureTrackingView_; // View for showing bounding box around the face
    CGAffineTransform cameraOutputToPreviewFrameTransform_;
    CGAffineTransform portraitRotationTransform_;
    CGAffineTransform texelToPixelTransform_;
}
Smerk Example

// ViewController.m
// Smerk_Example
//
// Created by Simon Lucey on 11/30/15.
// Copyright © 2015 CMU_16432. All rights reserved.
//
#import "ViewController.h"
#import "SMKDetectionCamera.h"
#import <GPUImage/GPUImage.h>

@interface ViewController () {
    // Setup the view (this time using GPUImageViewController)
    GPUImageView *cameraView_
    SMKDetectionCamera *detector_; // Detector that should be used
    UIView *faceFeatureTrackingView_; // View for showing bounding box around the face
    CGAffineTransform cameraOutputToPreviewFrameTransform_
    CGAffineTransform portraitRotationTransform_
    CGAffineTransform texelToPixelTransform_
}
Smerk Example

// ViewController.m  
// Smerk_Example
//
// Created by Simon Lucey on 11/30/15.
// Copyright © 2015 CMU_16432. All rights reserved.
//

#import "ViewController.h"
#import "SMKDetectionCamera.h"
#import <GPUImage/GPUImage.h>

@interface ViewController () {
    // Setup the view (this time using GPUImageView)
    GPUImageView *cameraView_
    SMKDetectionCamera *detector_;  // Detector that should be used
    UIView *faceFeatureTrackingView_; // View for showing bounding box around the face
    CGAffineTransform cameraOutputToPreviewFrameTransform_;  
    CGAffineTransform portraitRotationTransform_;  
    CGAffineTransform texelToPixelTransform_; 
}

Smerk Example

```objc
// Setup GPUImageView (not we are not using UI ImageView here)........
cameraView_ = [[GPUImageView alloc] initWithFrame: CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)];

// Set the face detector to be used
detector_ = [[SMKDetectionCamera alloc] initWithSessionPreset: AVCaptureSessionPreset640x480 cameraPosition: .front];
detector_ setOutputImageOrientation: UIInterfaceOrientationPortrait]; // Set to portrait
cameraView_.fillMode = kGPUImageFillModePreserveAspectRatio;
detector_ addTarget:cameraView_];

// Important: add as a subview
[self.view addSubview:cameraView_];

// Setup the face box view
[self setupFaceTrackingViews];
[self calculateTransformations];

// Set the block for running face detector
detector_ beginDetecting:kFaceFeatures | kMachineAndFaceMetaDataTableypes:@[AVMetadataObjectTypeQRC code]
withDetectionBlock:^{SMKDetectionOptions detectionType, NSArray *detectedObjects, CGRect rect)
    // Check if the kFaceFeatures have been discovered
    if (detectionType & kFaceFeatures) {
        [self updateFaceFeatureTrackingViewWithObjects:detectedObjects];
    }
}];

// Finally start the camera
detector_ startCameraCapture];
```
Smerk Example

// Setup GPUImage View (not we are not using UIImage View here) .........
cameraView_ = [[GPUImageView alloc] initWithFrame: CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)];

// Set the face detector to be used
detector_ = [[SMKDetectionCamera alloc] initWithSessionPreset:AVCaptureSessionPreset640x480 cameraParameters:[cameraParameters objectForKey:AVCaptureDeviceTypeFace]);
detector_.setOutputImageOrientation:UIInterfaceOrientationPortrait; // Set to portrait
cameraView_.fillMode = kGPUImageViewFillModePreserveAspectRatio;
detector_.addTarget:cameraView_;

// Important: add as a subview
[self.view addSubview:cameraView_];

// Setup the face box view
[self setupFaceTrackingViews];
[self calculateTransformations];

// Set the block for running face detector
detector_.beginDetecting:kFaceFeatures | kMachineAndFaceMetaData
codeTypes:@[AVMetadataObjectTypeQRCode]
withDetectionBlock:^SMKDetectionOptions detectionType, NSArray *detectedObjects, CGRect rect)
    // Check if the kFaceFeatures have been discovered
    if (detectionType & kFaceFeatures) {
        [self updateFaceFeatureTrackingViewWithObjects:detectedObjects];
    }
}];

// Finally start the camera
detector_.startCameraCapture];
// Setup GPUImage View (not we are not using UIImageView here)........
cameraView_ = [[GPUImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width, 

// Set the face detector to be used
detector_ = [[SMKDetectionCamera alloc] initWithSessionPreset:AVCaptureSessionPreset640x480 cameraPresetType:AVCaptureVideoPresetCameraPreset640x480]; // Set to portrait
cameraView_.fillMode = kGPUImageFillModePreserveAspectRatio;
[detector_ addTarget:cameraView_];

// Important: add as a subview
[self.view addSubview:cameraView_];

// Setup the face box view
[self setupFaceTrackingViews];
[self calculateTransformations];

// Set the block for running face detector
[detector_ beginDetecting:kFaceFeatures | kMachineAndFaceMetaData
codeTypes:@[AVMetadataObjectTypeQRCode]
withDetectionBlock:^([SMKDetectionOptions detectionType, NSArray *detectedObjects, CGRect)
// Check if the kFaceFeatures have been discovered
if (detectionType & kFaceFeatures) {
    [self updateFaceFeatureTrackingViewWithObjects:detectedObjects];
}
}

// Finally start the camera
[detector_ startCameraCapture];
Today

• Facial Feature Detection in iOS.

• Active Appearance Models.
Rigid Warps

- Up until now, we have been limited to dealing with objects that lie conveniently within a rectangle.
- These objects can only deform rigidly,
Non-Rigid Warps

- What happens if I want to deal with non-rigid warps,

(Szeliski and Fleet)
Non-Rigid Warps

• What happens if I want to deal with non-rigid warps,

(Szeliski and Fleet)
Active Appearance Models

• Introduced by Cootes and Taylor in 1998
• Example of class of deformable models which also includes Active Blobs (Sclaroff and Isidoror), Active Shape Models (Cootes and Taylor) and Morphable Models (Blanz and Vetter)
• Used successfully on faces, hands, and in medical imaging

(Matthews and Gross)
Active Appearance Models

- Separately model object shape and appearance

AAM

- Shape
- Appearance

(Matthews and Gross)
Active Appearance Models

- Separately model object shape and appearance

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>-0.34</td>
<td></td>
</tr>
<tr>
<td>6.78</td>
<td></td>
</tr>
<tr>
<td>-12.2</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td></td>
</tr>
</tbody>
</table>

AAM

- Shape
- Appearance

(Matthews and Gross)
Active Appearance Models

- Separately model object shape and appearance

Parameters:
0.12  
-0.34  
6.78   
-12.2  
0.01   

AAM

Shape

Appearance

Image

(Matthews and Gross)
Active Appearance Models

- Separately model object shape and appearance

Parameters
- 0.12
- -0.34
- 6.78
- -12.2
- 0.01

AAM
- Shape
- Appearance

Image

(Matthews and Gross)
Active Appearance Models

- Separately model object shape and appearance

Parameters
0.12
-0.34
6.78
-12.2
0.01

AAM
Shape
Appearance

Fitting
Modeling
Image

(Matthews and Gross)
Active Appearance Models

(Matthews and Gross)
Active Appearance Models

Appearance  

Shape  

Combined  

(Matthews and Gross)
AAM Overview

• How do you build an AAM?
• How do you fit an AAM to an image?
  • What is the general approach?
  • How can we do it efficiently?
• Extensions

(Matthews and Gross)
Model Building - Definition

- Define a set of landmarks on a face that can be reliably located in an image

- Still open questions:
  - How many landmarks, which landmarks

(Matthews and Gross)
Model Building - Definition

• Define a set of landmarks on a face that can be reliably located in an image

• Still open questions:
  • How many landmarks, which landmarks

(Matthews and Gross)
Model Building - Labeling

\[ (x_1, y_1, x_2, y_2, \ldots, x_n, y_n) \]

Manually label lots and lots of data

(Matthews and Gross)
Model Building - Shape

- Various sources of shape variations between individual images
  - Identity
  - Facial expression
  - Position in the image

(Matthews and Gross)
Model Building - Shape

- Various sources of shape variations between individual images
  - Identity
  - Facial expression
  - Position in the image

Don’t Care!

(Matthews and Gross)
Shape Alignment

• Procrustes analysis
  • Removes variation due to predetermined shape transformation

(Matthews and Gross)
Shape Alignment

• Procrustes analysis
  • Removes variation due to predetermined shape transformation

(Matthews and Gross)
Shape Alignment

- Procrustes analysis
  - Removes variation due to predetermined shape transformation

(Matthews and Gross)
Building a Shape Model

Input Shape = Basis Parameters

(Matthews and Gross)
Building a Shape Model

Input Shape = Basis Parameters

Principal Component Analysis

(Matthews and Gross)
PCA on Appearance

Original

(Matthews and Gross)
PCA on Appearance

Original

Full

(Matthews and Gross)
PCA on Appearance

Original

Full

20 dim.

(Matthews and Gross)
PCA on Appearance

Original

Full

20 dim.

10 dim.

(Matthews and Gross)
PCA Shape Model

• **Input:**
  Bunch of aligned face shapes

• **Output**
  Mean shape and the direction of the largest variations (eigenvectors)

(Matthews and Gross)
PCA Shape Model

• Input: Bunch of aligned face shapes
• Output: Mean shape and the direction of the largest variations (eigenvectors)

(Matthews and Gross)
PCA Shape Model

Mean Face

First 3 Shape Modes

\[
\mathcal{W}(z; p) = s_0 + \sum_{k=1}^{K} p_k s_k
\]

Shape Model

Shape Parameters

\[
p = [p_1, p_2, \ldots, p_K]^T
\]  

(Matthews and Gross)
Construction of Appearance Model

Warp to mean shape

$S_0$

PCA on Appearance

(Matthews and Gross)
PCA Appearance Model

- Input:
  Bunch of shape normalized textures
- Output:
  Mean appearance and appearance eigenvectors

(Matthews and Gross)
PCA Appearance Model

• Input:
  Bunch of shape normalized textures

• Output:
  Mean appearance and appearance eigenvectors

(Matthews and Gross)
AAM Appearance Model

Mean Face

First 3 Appearance Modes

\[ A_0(z) \]

\[ A_1(z) \]

\[ A_2(z) \]

\[ A_3(z) \]

Appearance Model

\[ T(z) = A_0(z) + \sum_{m=1}^{M} \lambda_i A_m(z) \]

Appearance Parameters

\[ (\lambda_1, \lambda_2, \ldots, \lambda_n)^T \]

(Matthews and Gross)
AAM Model Building Overview

Shape Modes

Procrustes

PCA

Appearance Modes
AAM Model Building Overview

- **Shape Modes**: $s_0$, $s_1$, $s_2$
- **Appearance Modes**: $A_0$, $A_1$, $A_2$

Procrustes

PCA
Model Instantiation

\[ T(z) = A_0(z) + 3559A_1(z) + 351A_2(z) - 256A_3(z) \ldots \]

(Matthews and Gross)
Model Instantiation

\[ \mathcal{W}(z; p) \rightarrow T(\mathcal{W}(z; p)) \]

\[ \text{Appearance, } T(z) = A_0(z) + 3559A_1(z) + 351A_2(z) - 256A_3(z) \ldots \]

\[ \text{Shape, } z = s_0 - 54s_1 + 10s_2 - 9.1s_3 \ldots \]

(Matthews and Gross)
$T(z) = A_0(z) + \sum_{m=1}^{M} \lambda_i A_m(z)$

$\mathcal{W}(z; p) = s_0 + \sum_{k=1}^{K} p_k s_k$

(Matthews and Gross)
Model Instantiation

\[ T(z) = A_0(z) + \sum_{m=1}^{M} \lambda_i A_m(z) \]

\[ \mathcal{W}(z; p) = s_0 + \sum_{k=1}^{K} p_k s_k \]

Source Image \rightarrow 2D Similarity Transform \rightarrow \text{Warp} \rightarrow T(\mathcal{W}(z; p))

(Matthews and Gross)
A Parameterized Face

(Matthews and Gross)
A Parameterized Face

(Matthews and Gross)
Things to Note

• AAMs are **linear** models
  • Efficiently capture non-rigid face deformations

• AAMs are entirely **data-driven**
  • No “face constraints”

• We discussed independent AAMs. “Combined” AAMs add an additional PCA on the combined shape and appearance parameters.

(Matthews and Gross)
Hand AAM

(Matthews and Gross)
Hand AAM

(Matthews and Gross)
Things to Note

- AAMs are **linear** models
  - Efficiently capture non-rigid face deformations

- AAMs are entirely **data-driven**
  - No “face constraints”

- We discussed independent AAMs. “Combined” AAMs add an additional PCA on the combined shape and appearance parameters.

(Matthews and Gross)
Active Appearance Models

Determine the best model parameters to reconstruct the image

AAM

Shape

Appearance

(Matthews and Gross)
Determine the best model parameters to reconstruct the image

AAM

Shape

Appearance

Parameters

0.12
-0.34
6.78
-12.2
0.01

(Matthews and Gross)
Determine the best model parameters to reconstruct the image

(Matthews and Gross)
Active Appearance Models

Determine the best model parameters to reconstruct the image

Parameters
0.12
-0.34
6.78
-12.2
0.01

AAM

Shape

Appearance

Modeling

Image

(Matthews and Gross)
Active Appearance Models

Determine the best model parameters to reconstruct the image

(Matthews and Gross)
AAM - Fitting “Simultaneous”

- Now that we have defined an AAM, how should we do the fitting?
- Turns out, we can still use the “simultaneous” extension of the LK algorithm,

\[
\arg \min_{\Delta \lambda, \Delta p} \| T(z) + A \Delta \lambda - I(W(z; p)) - J \Delta p \|^2
\]

- where,

\[
\frac{\partial W(z; p)^T}{\partial p} = [s_1, \ldots, s_K] \quad J = \frac{\partial I(W(z; p)^T}{\partial W(z; p)} \frac{\partial W(z; p)^T}{\partial p}
\]
Even faster if we use the inverse composition (IC) extension as well as “projecting out” the appearance,

$$\arg \min_{\Delta p} \left\| T(z) + J_{ic} \Delta p - I(W(z; p)) \right\|^2_{null(A)}$$

where,

$$\frac{\partial W(z; 0)^T}{\partial p} = [s_1, \ldots, s_K] \quad J_{ic} = \frac{\partial T(W(z; 0)^T}{\partial W(z; 0)} \frac{\partial W(z; 0)}{\partial p}$$
IC AAM Fitting

- Runs at 230 Hz on a 3.2GHz PC

(Matthews and Gross)
IC AAM Fitting

- Runs at 230 Hz on a 3.2GHz PC

(Matthews and Gross)
Why We Need High Speed?

Re-initialize model multiple times if tracking fails and still track in real time

(Matthews and Gross)
Why We Need High Speed?

Re-initialize model multiple times if tracking fails and still track in real time

(Matthews and Gross)
Original model does not handle occlusion well

(Matthews and Gross)
Not All Is Peachy

Original model does not handle occlusion well

(Matthews and Gross)
To handle occlusions we can employ the robust error function,

\[
\arg \min_{\Delta \lambda, \Delta p} \eta(T(z) + A \Delta \lambda - I(W(z; p)) - J \Delta p)
\]

where,

\[
\frac{\partial W(z; p)^T}{\partial p} = [s_1, \ldots, s_K] \quad J = \frac{\partial I(W(z; p)^T}{\partial W(z; p)} \frac{\partial W(z; p)^T}{\partial p}
\]
AAMs with Occlusion Modeling

(Matthews and Gross)
AAMs with Occlusion Modeling

(Matthews and Gross)
Applications

- **User Interfaces:**
  - Mouse replacement
  - Automotive: Windshield Displays, Smart Airbags

- **Face Recognition:**
  - Pose Normalization

- **Lipreading/Audio-Visual Speech Recognition**

- **Rendering and Animation:**
  - Low-Bandwidth Video Conferencing
  - Audio-Visual Speech Synthesis

(Matthews and Gross)
User Interfaces: Head Pose

- Mouse replacement

(Matthews and Gross)
User Interfaces: Head Pose

- Mouse replacement

(Matthews and Gross)
User Interfaces: Gaze Tracking

Driver Camera

Exterior View Camera

(Matthews and Gross)
User Interfaces: Gaze Tracking

Driver Camera

Exterior View Camera

(Matthews and Gross)
Face Recognition: Pose Normalization

(Matthews and Gross)
Face Recognition: Pose Normalization

(Matthews and Gross)
Animation Generation

(Matthews and Gross)
Animation Generation

(Matthews and Gross)
Audio-Visual Speech Synthesis

Jingle Bells, Jingle Bells,
Jingle All the Way, ...

(Matthews and Gross)
Audio-Visual Speech Synthesis

Jingle Bells, Jingle Bells,
Jingle All the Way, ...

(Matthews and Gross)