

# **Rapid Facial Feature Detection in iOS**

Instructor - Simon Lucey

**16-423 - Designing Computer Vision Apps**



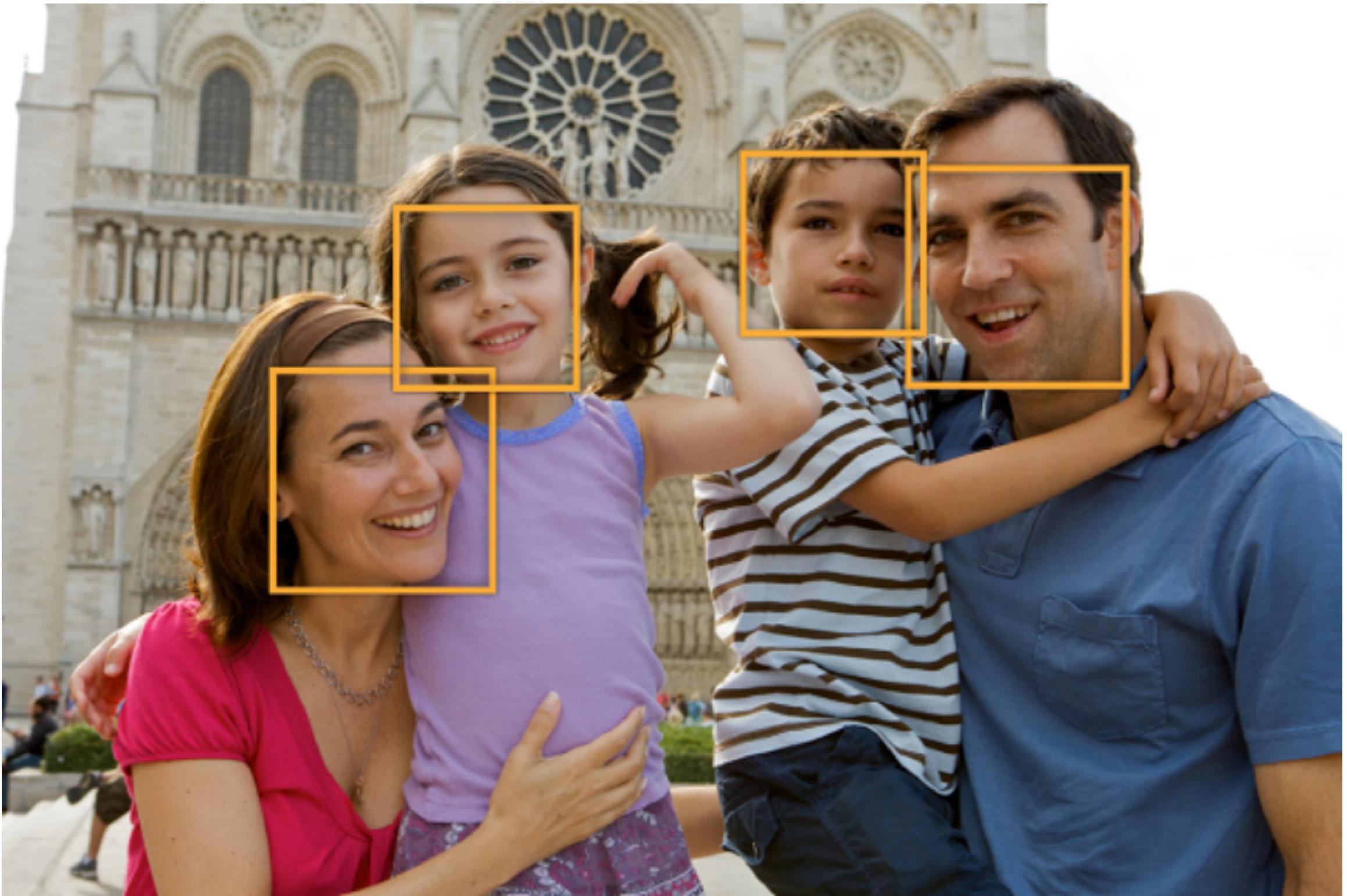
**Carnegie Mellon**  
THE ROBOTICS INSTITUTE

# Today

---

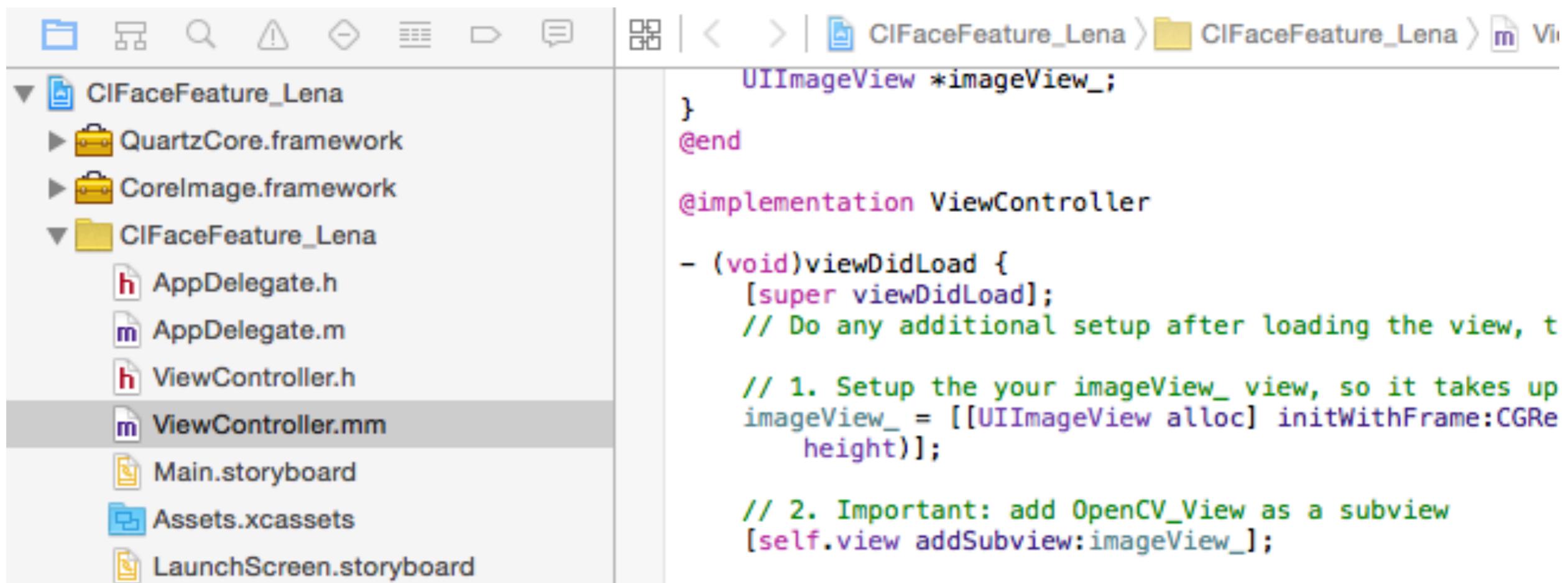
- Facial Feature Detection in iOS.
- Active Appearance Models.

# 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.



The screenshot shows the Xcode interface. On the left is a file browser with the following structure:

- CIfaceFeature\_Lena (grouped under CIfaceFeature\_Lena)
- QuartzCore.framework
- CoreImage.framework
- CIfaceFeature\_Lena (grouped under CIfaceFeature\_Lena)
  - AppDelegate.h
  - AppDelegate.m
  - ViewController.h
  - ViewController.mm (selected)
  - Main.storyboard
  - Assets.xcassets
  - LaunchScreen.storyboard

The code editor on the right displays the content of ViewController.mm:

```
UIImageView *imageView_;
```

```
}
```

```
@end
```

```
@implementation ViewController
```

```
- (void)viewDidLoad {
```

```
    [super viewDidLoad];
```

```
    // Do any additional setup after loading the view, t
```

```
    // 1. Setup the your imageView_ view, so it takes up
```

```
    imageView_ = [[UIImageView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
```

```
    // 2. Important: add OpenCV_View as a subview
```

```
    [self.view addSubview:imageView_];
```

# 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.

The screenshot shows the Xcode interface with the project navigation bar at the top and the file browser on the left. The main area displays the ViewController.mm file containing Objective-C code for setting up a UIImageView. Two framework files, QuartzCore.framework and CoreImage.framework, are highlighted with a red box in the project navigator.

```
UIImageView *imageView_;
```

```
}
```

```
@end
```

```
@implementation ViewController
```

```
- (void)viewDidLoad {
```

```
    [super viewDidLoad];
```

```
    // Do any additional setup after loading the view, t
```

```
    // 1. Setup the your imageView_ view, so it takes up
```

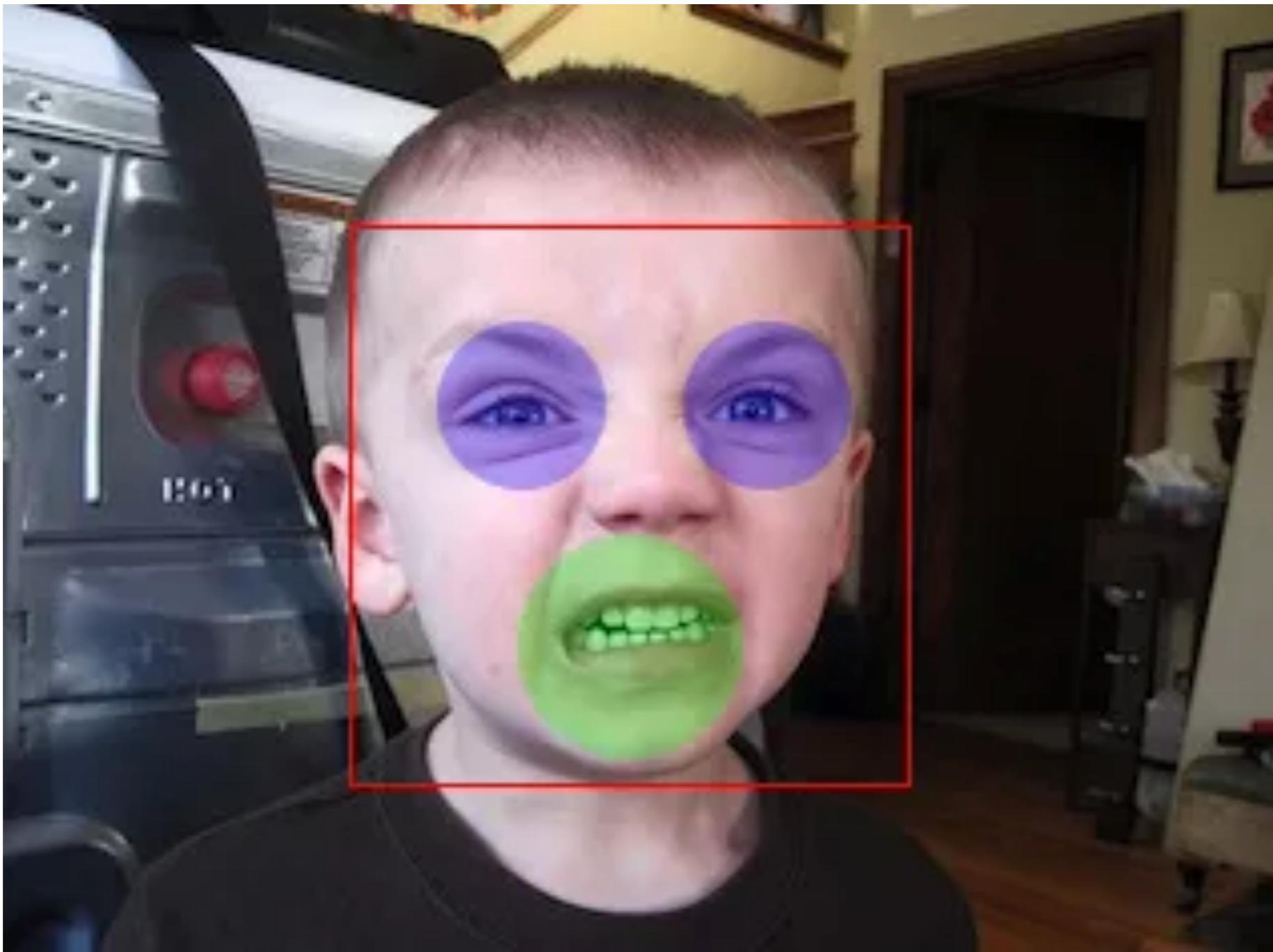
```
    imageView_ = [[UIImageView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
```

```
    // 2. Important: add OpenCV_View as a subview
```

```
    [self.view addSubview:imageView_];
```

# Facial Feature Detection

---



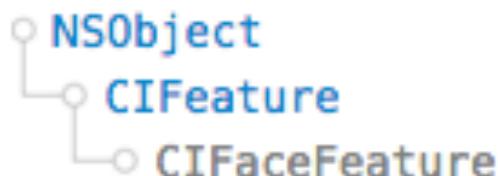
# CIFaceFeature

## CIFaceFeature

### IMPORTANT

This is a preliminary document for an API or technology in development. Apple is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein for use on Apple-branded products. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future betas of the API or technology.

### Inherits From



### Conforms To

NSObject

### Import Statement

```
OBJECTIVE-C  
@import CoreImage;
```

### Availability

Available in iOS 5.0 and later

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.

# CIFaceFeature

## 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](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
```

# CIFaceFeature Example

```
36 // 3. Load the image and display
37 UIImage *image = [UIImage imageNamed:@"lena.png"];
38 if(image != nil) imageView_.image = image; // Display the image if it is there....
39 else cout << "Cannot read in the file" << endl;
40
41 // 4. Initialize the face detector and its options
42 NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow };
43 CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];
44
45 // 5. Apply the image through the face detector
46 NSArray *features = [faceDetector featuresInImage:[CIImage imageWithCGImage: [image CGImage]]];
47
48 // 6. Loop through each detected face
49 for(CIFaceFeature* faceFeature in features) {
50
51     // Check facial features
52
53     // Smile
54     if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
55     else cout << "Lena is Not Smiling" << endl;
56
57     // Head angle
58     if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
59     else cout << "Lena's head is not at an angle" << endl;
60
61     // Left Eye location
62     CGPoint lEye = faceFeature.leftEyePosition;
63     if(faceFeature.hasLeftEyePosition)
64         cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
65     else
66         cout << "Lena's left eye is occluded!!" << endl;
67 }
```

# CIFaceFeature Example

```
36 // 3. Load the image and display
37 UIImage *image = [UIImage imageNamed:@"lena.png"];
38 if(image != nil) imageView_.image = image; // Display the image if it is there....
39 else cout << "Cannot read in the file" << endl;
40
41 // 4. Initialize the face detector and its options
42 NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow };
43 CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];
44
45 // 5. Apply the image through the face detector
46 NSArray *features = [faceDetector featuresInImage:[CIImage imageWithCGImage: [image CGImage]]];
47
48 // 6. Loop through each detected face
49 for(CIFaceFeature* faceFeature in features) {
50
51     // Check facial features
52
53     // Smile
54     if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
55     else cout << "Lena is Not Smiling" << endl;
56
57     // Head angle
58     if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
59     else cout << "Lena's head is not at an angle" << endl;
60
61     // Left Eye location
62     CGPoint lEye = faceFeature.leftEyePosition;
63     if(faceFeature.hasLeftEyePosition)
64         cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
65     else
66         cout << "Lena's left eye is occluded!!" << endl;
67 }
```

# CIFaceFeature Example

```
36 // 3. Load the image and display
37 UIImage *image = [UIImage imageNamed:@"lena.png"];
38 if(image != nil) imageView_.image = image; // Display the image if it is there....
39 else cout << "Cannot read in the file" << endl;
40
41 // 4. Initialize the face detector and its options
42 NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow };
43 CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];
44
45 // 5. Apply the image through the face detector
46 NSArray *features = [faceDetector featuresInImage:[CIImage imageWithCGImage: [image CGImage]]];
47
48 // 6. Loop through each detected face
49 for(CIFaceFeature* faceFeature in features) {
50
51     // Check facial features
52
53     // Smile
54     if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
55     else cout << "Lena is Not Smiling" << endl;
56
57     // Head angle
58     if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
59     else cout << "Lena's head is not at an angle" << endl;
60
61     // Left Eye location
62     CGPoint lEye = faceFeature.leftEyePosition;
63     if(faceFeature.hasLeftEyePosition)
64         cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
65     else
66         cout << "Lena's left eye is occluded!!" << endl;
67 }
```

# CIFaceFeature Example

```
36 // 3. Load the image and display
37 UIImage *image = [UIImage imageNamed:@"lena.png"];
38 if(image != nil) imageView_.image = image; // Display the image if it is there....
39 else cout << "Cannot read in the file" << endl;
40
41 // 4. Initialize the face detector and its options
42 NSDictionary *detectorOptions = @{ CIDetectorAccuracy : CIDetectorAccuracyLow };
43 CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace context:nil options:detectorOptions];
44
45 // 5. Apply the image through the face detector
46 NSArray *features = [faceDetector featuresInImage:[CIImage imageWithCGImage: [image CGImage]]];
47
48 // 6. Loop through each detected face
49 for(CIFaceFeature* faceFeature in features) {
50
51     // Check facial features
52
53     // Smile
54     if(faceFeature.hasSmile) cout << "Lena is Smiling" << endl;
55     else cout << "Lena is Not Smiling" << endl;
56
57     // Head angle
58     if(faceFeature.hasFaceAngle) cout << "Lena's head is at an angle of " << faceFeature.faceAngle << endl;
59     else cout << "Lena's head is not at an angle" << endl;
60
61     // Left Eye location
62     CGPoint lEye = faceFeature.leftEyePosition;
63     if(faceFeature.hasLeftEyePosition)
64         cout << "Lena's left eye is positioned at (" << lEye.x << "," << lEye.y << ")" << endl;
65     else
66         cout << "Lena's left eye is occluded!!" << endl;
67 }
```

# CIFaceFeature Example

The screenshot shows the Xcode interface with the CIFaceFeature example project open. The top status bar indicates "iPhone 6 - iPhone 6 / iOS 9.0 (13A4325c)" and the time "10:25 AM". The main window displays the project structure and code for the CIFaceFeature\_Lena application.

**Project Structure:**

- iPhone 6 - iPhone 6 / iOS 9.0 (13A4325c)
- 10:25 AM
- File Inspector
- Project Navigator
- Assistant Editor
- Utilities
- Editor

**Code Editor (ViewController.mm):**

```
imageView_ = [[UIImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)];  
  
// 2. Important: add OpenCV_View as a subview  
[self.view addSubview:imageView_];  
  
// Ensure aspect ratio looks correct  
imageView_.contentMode = UIViewContentModeScaleAspectFit;  
  
// 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;  
}
```

**Output Window:**

```
Lena is Not Smiling  
Lena's head is at an angle of 3  
Lena's left eye is positioned at (267,242)
```

# CIFaceFeature Example

The screenshot displays two Xcode projects side-by-side.

**CIFaceFeature\_Lena Project:**

- File Navigator:** Shows files like AppDelegate.h, AppDelegate.m, ViewController.h, and ViewController.mm.
- Code Editor:** Displays the ViewController.mm code for face detection and analysis.
- Output Window:** Shows the console output with results of facial features.

```
imageView_ = [[UIImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width, self.view.frame.size.height)];  
// 2. Important: add OpenCV_View as a subview  
[self.view addSubview:imageView_];  
// Ensure aspect ratio looks correct  
imageView_.contentMode = UIViewContentModeScaleAspectFit;  
// 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;  
}
```

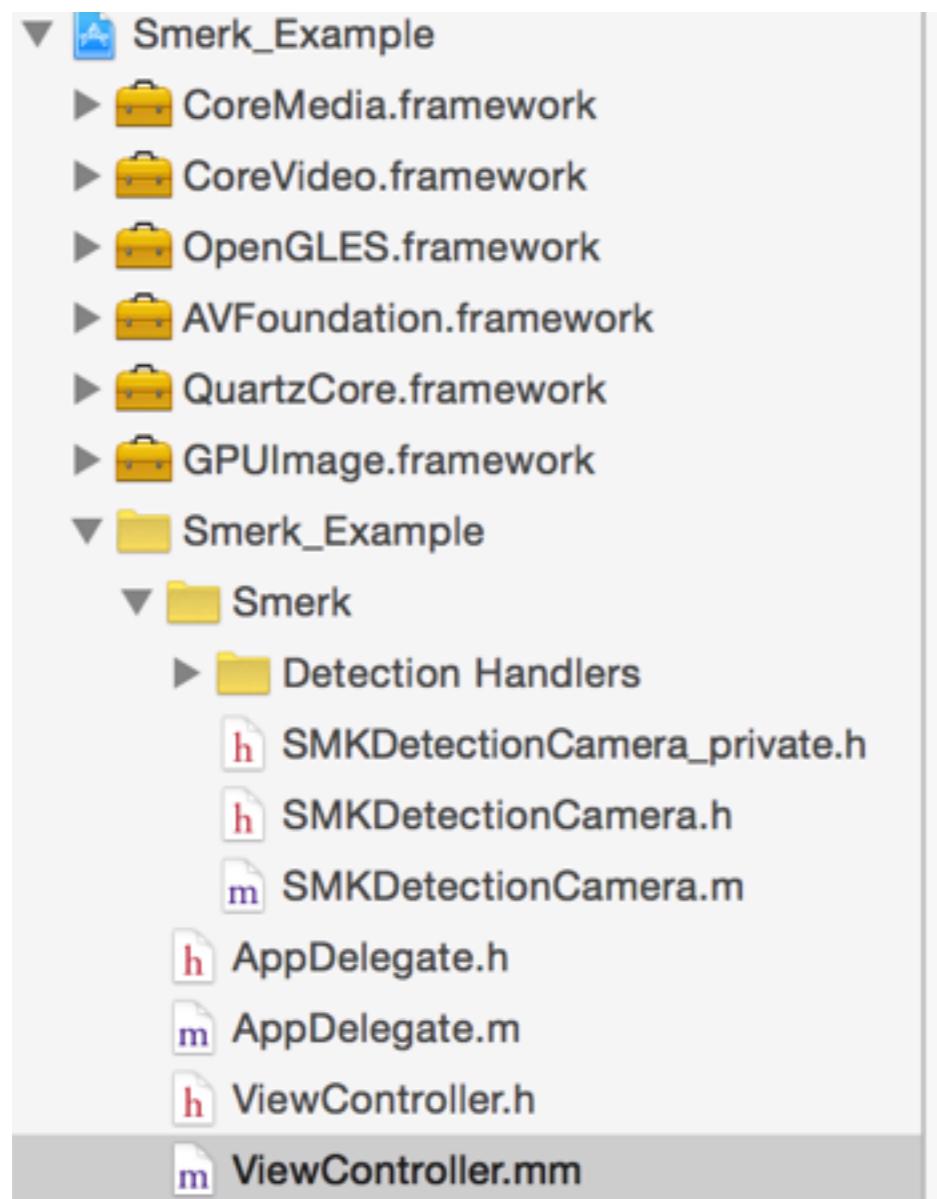
**XCode\_MBR Project:**

- File Navigator:** Shows files like IMUAppDelegate.h, IMUAppDelegate.m, MainMenu.xib, Supporting Files, Frameworks, and Products.
- Code Editor:** Displays the IMUAppDelegate.m code for initializing MBR fitting.
- Output Window:** Shows the build log for MBR fitting.

```
Initialize method for MBR fitting  
void initMBRfitting  
  
// Raw filter that takes the RGBA input from the other  
rawfilter_ = [[GPUImageRawDataOutput alloc] initWithResultsInBGRAFormat:YES];  
_unsafe_unretained GPUImageRawDataOutput * weakOutput;  
[rawfilter_ setNewFrameAvailableBlock:^(  
    GLubyte *outputBytes = [weakOutput rawBytesForIn  
    NSInteger bytesPerRow = [weakOutput bytesPerRow];  
    if(detected_) {  
        crop_pts_ = crop_ref_pts_; // Initialize each  
        // Apply the MBR  
        tracker_.ApplyMBR((const unsigned char*)outputBytes, bytesPerRow, crop_pts_);  
    }  
    // Initiate the detector if needed  
    if(!init_detected_) {  
        init_detected_ = YES;  
        [self initDetector];  
    }  
});
```

# 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>



```
// Set the face detector to be used
detector_ = [[SMKDetectionCamera alloc] initWithCaptureDevice:[AVCaptureDevice deviceWithPosition:AVCaptureDevicePositionFront]];
[detector_ setOutputImageOrientation:cameraView_.fillMode];
[cameraView_.fillMode = kGPUImageFillNone];
[detector_ addTarget:cameraView_];

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

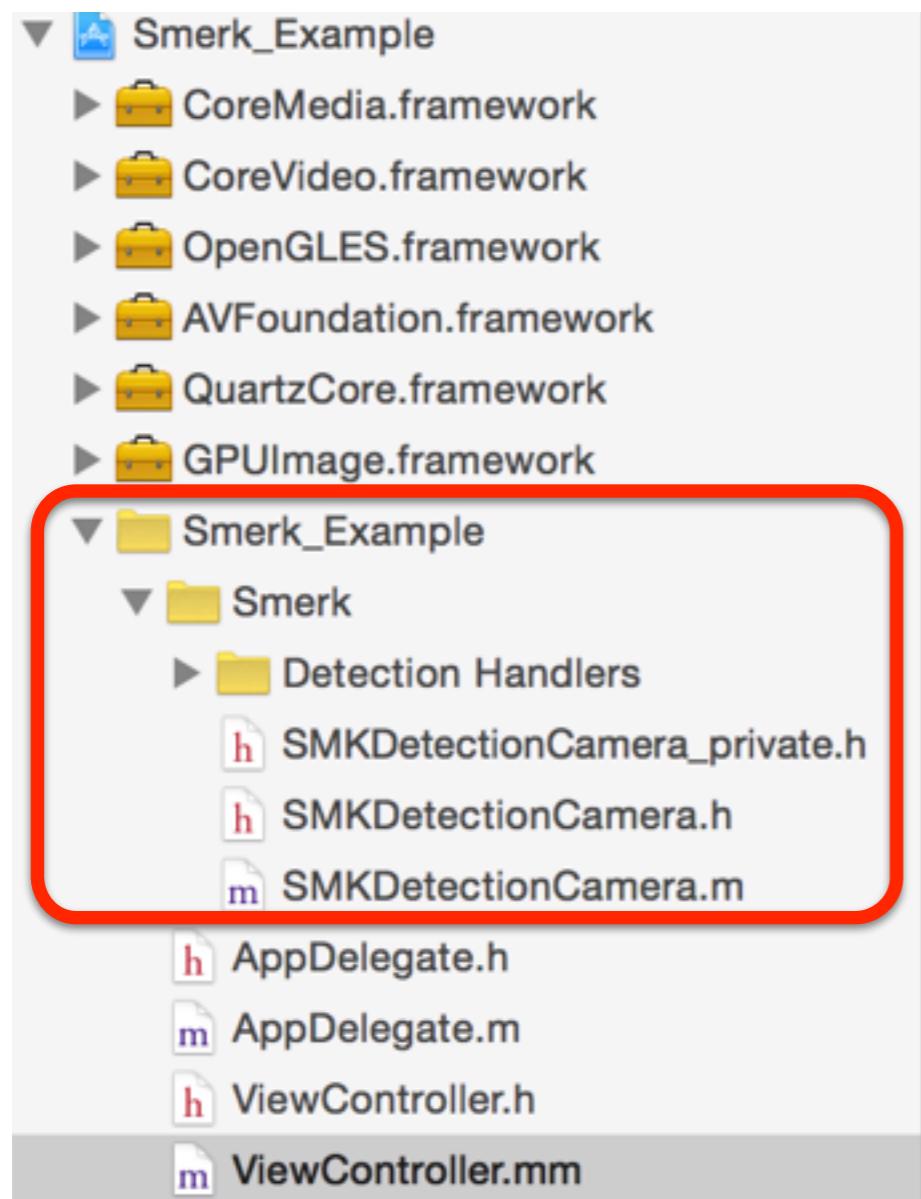
// Setup the face box view
[self setupFaceTrackingViews];
[self calculateTransformations];

// Set the block for running face detection
[detector_ beginDetecting:kFaceFeature
                      codeTypes:@[kCodeTypeFace]
               withDetectionBlock:^(SMKFace *face)
{
    // Check if the kFaceFeature is detected
    if (detectionType == kFaceFeature)
        [self updateFaceWithFace:face];
}
];
}

// Finally start the camera
[detector_ 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>



```
// Set the face detector to be used
detector_ = [[SMKDetectionCamera alloc] initWithCaptureDevice:[AVCaptureDevice deviceWithPosition:AVCaptureDevicePositionFront]];
[detector_ setOutputImageOrientation:cameraView_.fillMode];
[cameraView_.fillMode = kGPUImageFillNone];
[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
[detector_ beginDetecting:kFaceFeature
                      codeTypes:@[kCodeTypeFace]
withDetectionBlock:^(SMKFace *face)
{
    // Check if the kFaceFeature is detected
    if (detectionType == kFaceFeature)
        [self updateFaceWithFace:face];
}];

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

# Smerk Example

---

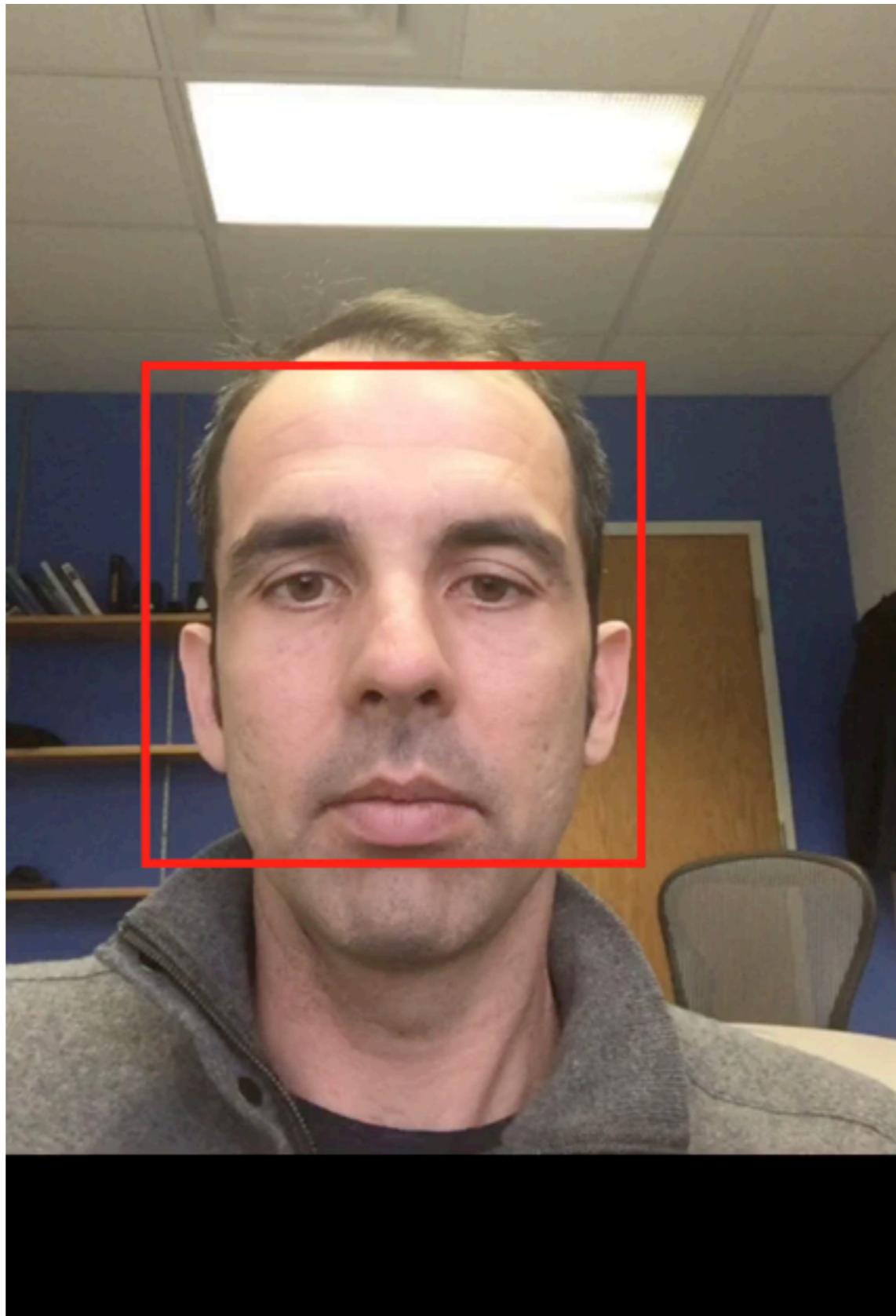
- We are now going to demonstrate how we can perform real-time face tracking through GPUImage.
- On your browser please go to the address,

[https://github.com/slucey-cs-cmu-edu/Smerk\\_Example](https://github.com/slucey-cs-cmu-edu/Smerk_Example)

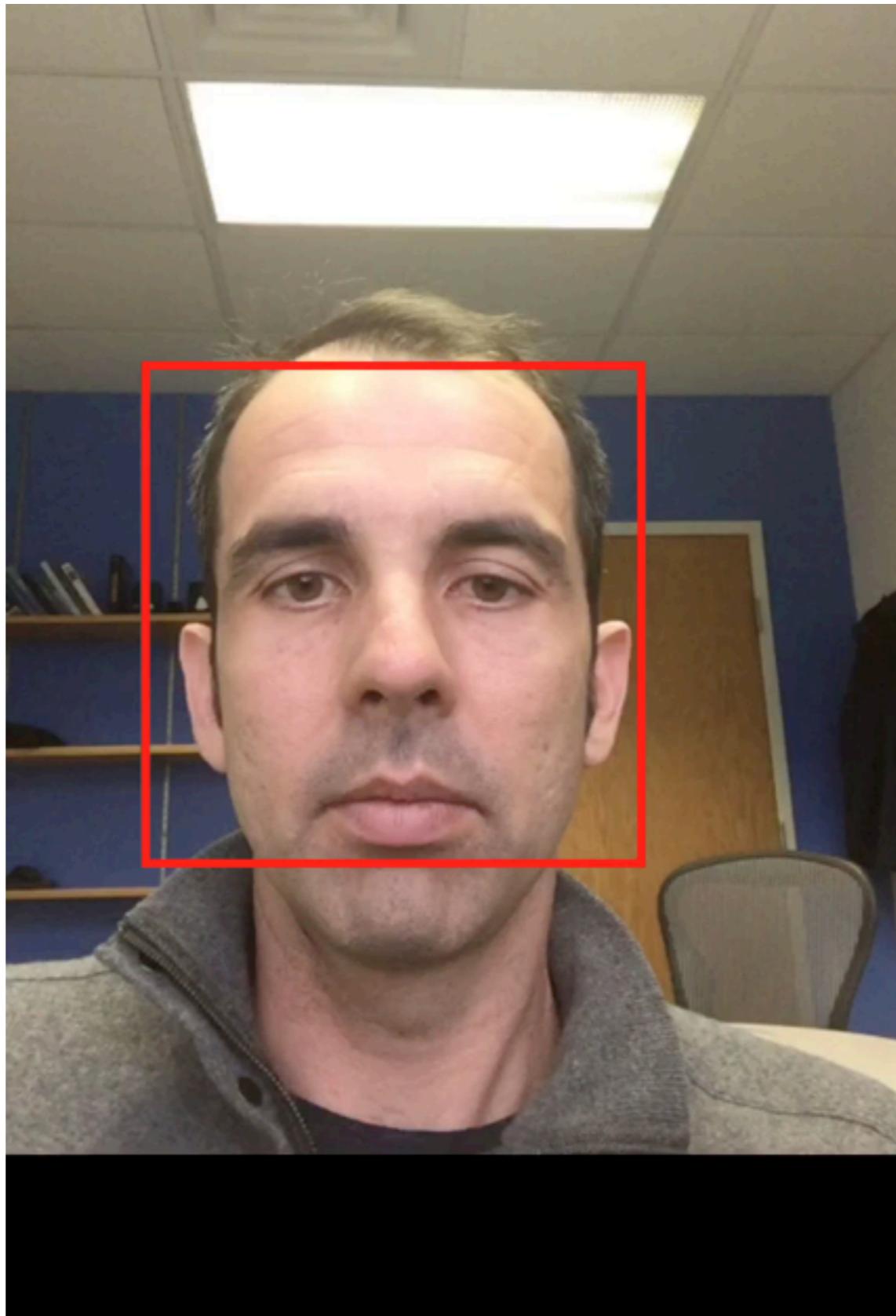
- Or better yet, if you have git installed you can type from the command line.

```
$ git clone https://github.com/slucey-cs-cmu-edu/Smerk\_Example.git
```

# Smerk Example



# Smerk Example



# Smerk Example

The screenshot shows the Xcode Instruments application interface. At the top, there are navigation buttons and a title bar indicating the target is "Smerk\_Example" running on "Simon Lucey's iPhone". Below the title bar is a toolbar with various icons. The main area is divided into two sections: a left sidebar and a right content view. The sidebar lists performance metrics for the process "Smerk\_Example" (PID 1200): CPU usage at 106%, Memory usage at 11.6 MB, Disk usage at Zero KB/s, Network usage at Zero KB/s, and FPS at 30 FPS. The content view displays the source code for the file "ViewController.m" under the "Smerk\_Example" folder. The code includes imports for "ViewController.h", "SMKDetectionCamera.h", and "GPUImage/GPUImage.h", and defines an interface for the ViewController class.

```
// 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 : UIViewController
{
    GPUImageView *cameraView_;
    SMKDetectionCamera *detector_; // Detector that
    UIView *faceFeatureTrackingView_; // View for showing
    CGAffineTransform cameraOutputToPreviewFrameTransform_;
    CGAffineTransform portraitRotationTransform_;
    CGAffineTransform texelToPixelTransform_;
}

@end

@implementation ViewController
```

# Smerk Example

The screenshot shows the Xcode Instruments application interface. At the top, there are navigation buttons and a title bar indicating the target is "Smerk\_Example" running on "Simon Lucey's iPhone". Below the title bar is a toolbar with various icons. The main area is divided into two sections: a left sidebar and a right content area.

**Left Sidebar:** Shows monitoring metrics for the process "Smerk\_Example" (PID 1200). The metrics listed are CPU (106%), Memory (11.6 MB), Disk (Zero KB/s), Network (Zero KB/s), and FPS (30 FPS, highlighted with a red box).

**Right Content Area:** Displays the source code for the "ViewController.m" file. The code includes comments about the creation date (11/30/15) and copyright (© 2015 CMU\_16432). It also imports "ViewController.h", "SMKDetectionCamera.h", and "GPUImage/GPUImage.h". The implementation section starts with an @interface declaration for ViewController.

```
// 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 : UIViewController
    // Setup the view (this time using GPUImageView)
    GPUImageView *cameraView_;
    SMKDetectionCamera *detector_; // Detector that ...
    UIView *faceFeatureTrackingView_; // View for sho...
    CGAffineTransform cameraOutputToPreviewFrameTrans...
    CGAffineTransform portraitRotationTransform_;
    CGAffineTransform texelToPixelTransform_;
}

@end

@implementation ViewController
```

# Smerk Example

```
1 //  
2 // ViewController.m  
3 // Smerk_Example  
4 //  
5 // Created by Simon Lucey on 11/30/15.  
6 // Copyright © 2015 CMU_16432. All rights reserved.  
7 //  
8  
9 #import "ViewController.h"  
10 #import "SMKDetectionCamera.h"  
11 #import <GPUImage/GPUImage.h>  
12  
13 @interface ViewController () {  
14     // Setup the view (this time using GPUImageView)  
15     GPUImageView *cameraView_;  
16     SMKDetectionCamera *detector_; // Detector that should be used  
17     UIView *faceFeatureTrackingView_; // View for showing bounding box around the face  
18     CGAffineTransform cameraOutputToPreviewFrameTransform_;  
19     CGAffineTransform portraitRotationTransform_;  
20     CGAffineTransform texelToPixelTransform_;  
21 }  
22
```

# Smerk Example

```
1 //  
2 // ViewController.m  
3 // Smerk_Example  
4 //  
5 // Created by Simon Lucey on 11/30/15.  
6 // Copyright © 2015 CMU_16432. All rights reserved.  
7 //  
8  
9 #import "ViewController.h"  
10 #import "SMKDetectionCamera.h"  
11 #import <GPUImage/GPUImage.h>  
12  
13 @interface ViewController () {  
14     // Setup the view (this time using GPUImageView)  
15     GPUImageView *cameraView_;  
16     SMKDetectionCamera *detector_; // Detector that should be used  
17     UIView *faceFeatureTrackingView_; // View for showing bounding box around the face  
18     CGAffineTransform cameraOutputToPreviewFrameTransform_;  
19     CGAffineTransform portraitRotationTransform_;  
20     CGAffineTransform texelToPixelTransform_;  
21 }  
22
```

# Smerk Example

```
1 //  
2 // ViewController.m  
3 // Smerk_Example  
4 //  
5 // Created by Simon Lucey on 11/30/15.  
6 // Copyright © 2015 CMU_16432. All rights reserved.  
7 //  
8  
9 #import "ViewController.h"  
10 #import "SMKDetectionCamera.h"  
11 #import <GPUImage/GPUImage.h>  
12  
13 @interface ViewController () {  
14     // Setup the view (this time using GPUImageView)  
15     GPUImageView *cameraView_;  
16     SMKDetectionCamera *detector_; // Detector that should be used  
17     UIView *faceFeatureTrackingView_; // View for showing bounding box around the face  
18     CGAffineTransform cameraOutputToPreviewFrameTransform_;  
19     CGAffineTransform portraitRotationTransform_;  
20     CGAffineTransform texelToPixelTransform_;  
21 }  
22
```

# Smerk Example

```
32 // Setup GPUImageView (not we are not using UIImageView here).....
33 cameraView_ = [[GPUImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width,
34
35 // Set the face detector to be used
36 detector_ = [[SMKDetectionCamera alloc] initWithSessionPreset:AVCaptureSessionPreset640x480 cameraP
37 [detector_ setOutputImageOrientation:UIInterfaceOrientationPortrait]; // Set to portrait
38 cameraView_.fillMode = kGPUImageFillModePreserveAspectRatio;
39 [detector_ addTarget:cameraView_];
40
41 // Important: add as a subview
42 [self.view addSubview:cameraView_];
43
44 // Setup the face box view
45 [self setupFaceTrackingViews];
46 [self calculateTransformations];
47
48 // Set the block for running face detector
49 [detector_ beginDetecting:kFaceFeatures | kMachineAndFaceMetaData
50             codeTypes:@[AVMetadataObjectTypeQRCode]
51             withDetectionBlock:^(SMKDetectionOptions detectionType, NSArray *detectedObjects, CGRect
52                             // Check if the kFaceFeatures have been discovered
53                             if (detectionType & kFaceFeatures) {
54                                 [self updateFaceFeatureTrackingViewWithObjects:detectedObjects];
55                             }
56                         }];
57
58 // Finally start the camera
59 [detector_ startCameraCapture];
```

# Smerk Example

```
32 // Setup GPUImageView (not we are not using UIImageView here).....
33 cameraView_ = [[GPUImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width,
34
35 // Set the face detector to be used
36 detector_ = [[SMKDetectionCamera alloc] initWithSessionPreset:AVCaptureSessionPreset640x480 cameraP
37 [detector_ setOutputImageOrientation:UIInterfaceOrientationPortrait]; // Set to portrait
38 cameraView_.fillMode = kGPUImageFillModePreserveAspectRatio;
39 [detector_ addTarget:cameraView_];
40
41 // Important: add as a subview
42 [self.view addSubview:cameraView_];
43
44 // Setup the face box view
45 [self setupFaceTrackingViews];
46 [self calculateTransformations];
47
48 // Set the block for running face detector
49 [detector_ beginDetecting:kFaceFeatures | kMachineAndFaceMetaData
50             codeTypes:@[AVMetadataObjectTypeQRCode]
51             withDetectionBlock:^(SMKDetectionOptions detectionType, NSArray *detectedObjects, CGRect
52                             // Check if the kFaceFeatures have been discovered
53                             if (detectionType & kFaceFeatures) {
54                                 [self updateFaceFeatureTrackingViewWithObjects:detectedObjects];
55                             }
56                         }];
57
58 // Finally start the camera
59 [detector_ startCameraCapture];
```

# Smerk Example

```
32 // Setup GPUImageView (not we are not using UIImageView here).....
33 cameraView_ = [[GPUImageView alloc] initWithFrame:CGRectMake(0.0, 0.0, self.view.frame.size.width,
34
35 // Set the face detector to be used
36 detector_ = [[SMKDetectionCamera alloc] initWithSessionPreset:AVCaptureSessionPreset640x480 cameraP
37 [detector_ setOutputImageOrientation:UIInterfaceOrientationPortrait]; // Set to portrait
38 cameraView_.fillMode = kGPUImageFillModePreserveAspectRatio;
39 [detector_ addTarget:cameraView_];
40
41 // Important: add as a subview
42 [self.view addSubview:cameraView_];
43
44 // Setup the face box view
45 [self setupFaceTrackingViews];
46 [self calculateTransformations];
47
48 // Set the block for running face detector
49 [detector_ beginDetecting:kFaceFeatures | kMachineAndFaceMetaData
50             codeTypes:@[AVMetadataObjectTypeQRCode]
51             withDetectionBlock:^(SMKDetectionOptions detectionType, NSArray *detectedObjects, CGRect
52                             // Check if the kFaceFeatures have been discovered
53                             if (detectionType & kFaceFeatures) {
54                                 [self updateFaceFeatureTrackingViewWithObjects:detectedObjects];
55                             }
56                         }];
57
58 // Finally start the camera
59 [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,



translation



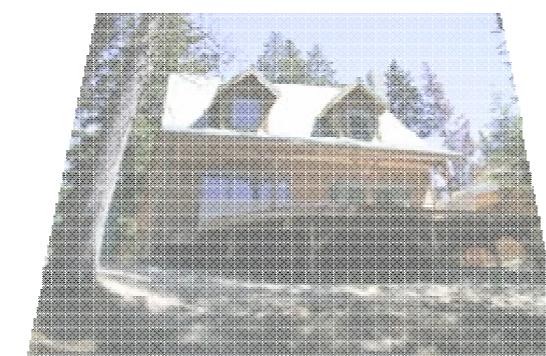
rotation



aspect



affine



perspective

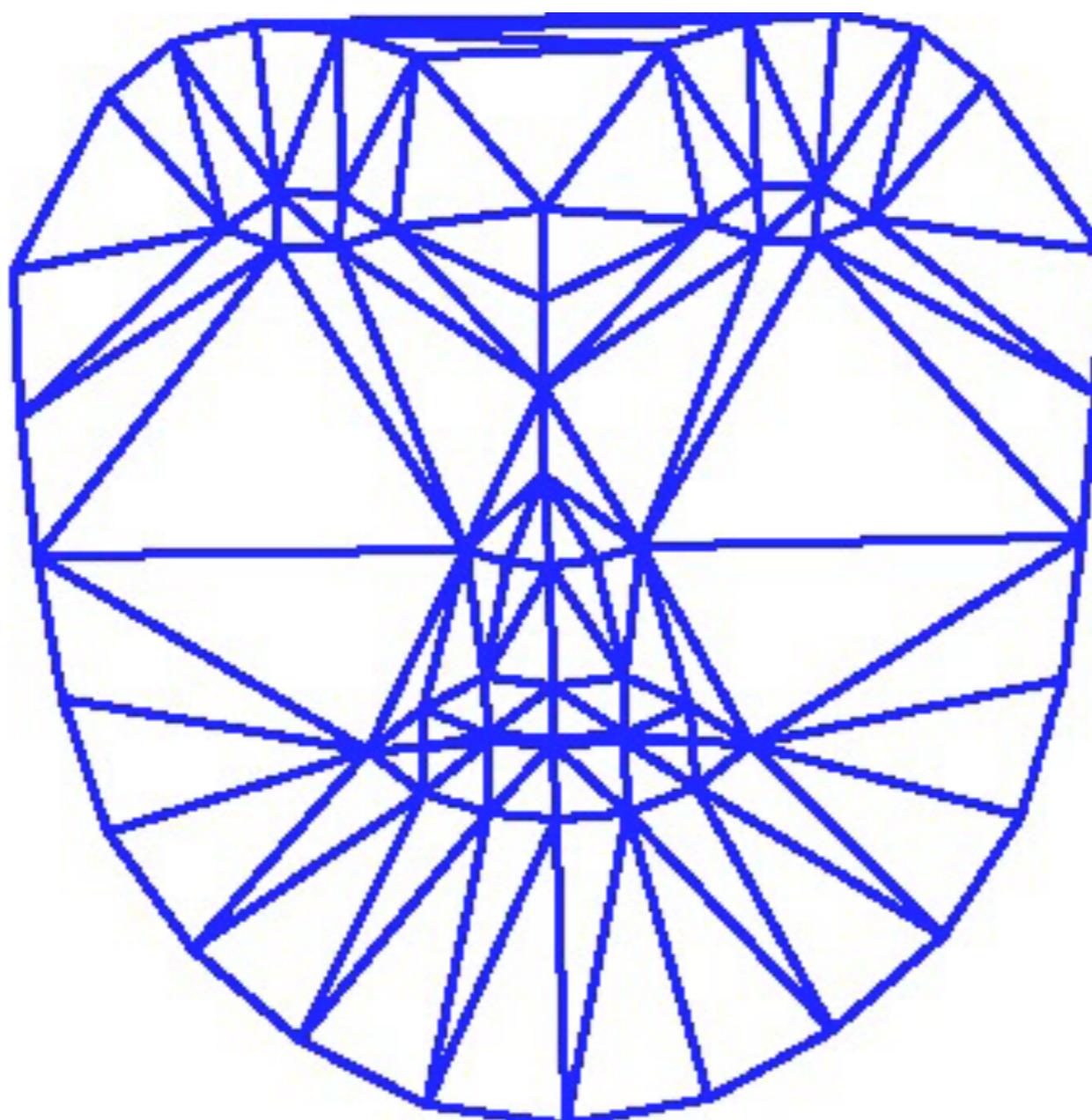


cylindrical

# Non-Rigid Warps

---

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

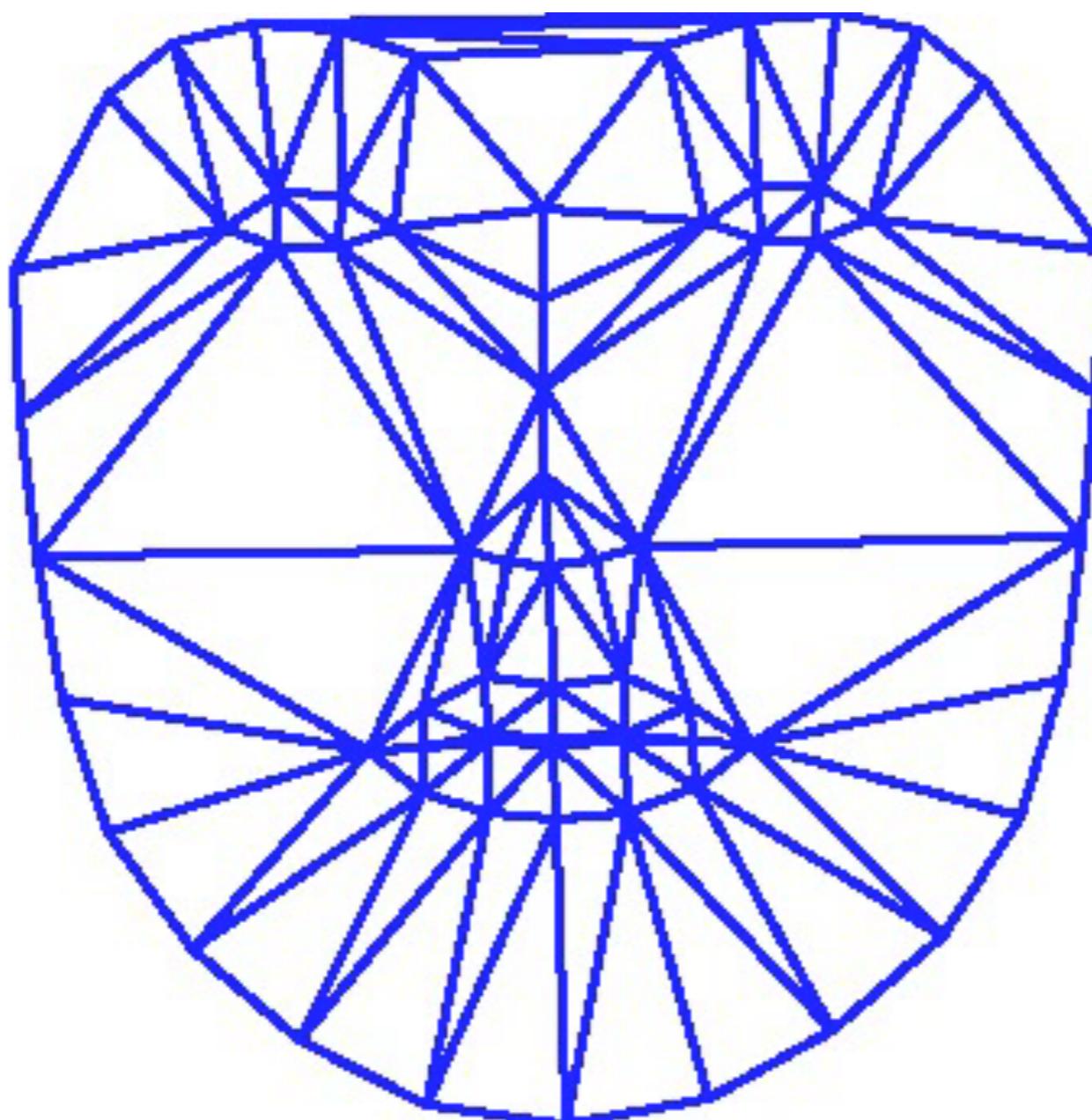


(Szeliski and Fleet)

# Non-Rigid Warps

---

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



(Szeliski and Fleet)

# Active Appearance Models

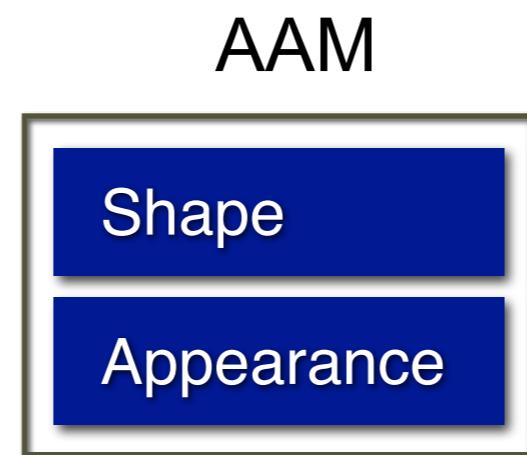
---

- 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



(Matthews and Gross)

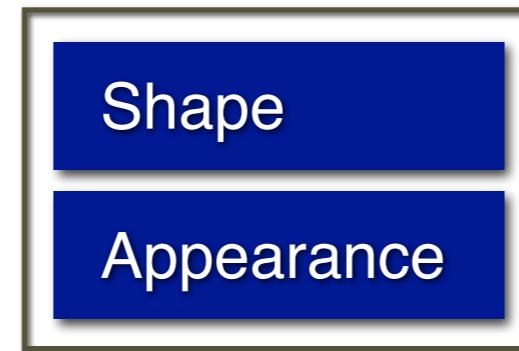
# Active Appearance Models

- Separately model object shape and appearance

Parameters

0.12  
-0.34  
6.78  
-12.2  
0.01

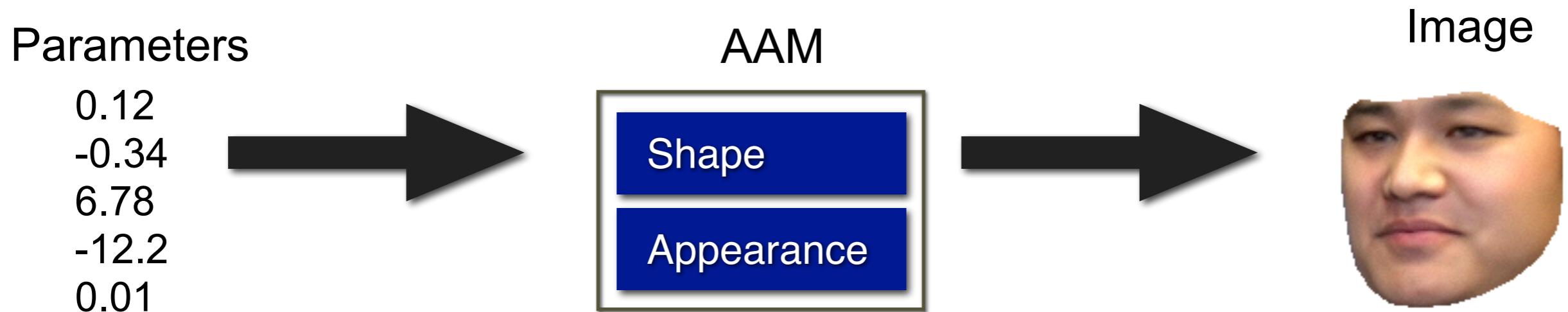
AAM



(Matthews and Gross)

# Active Appearance Models

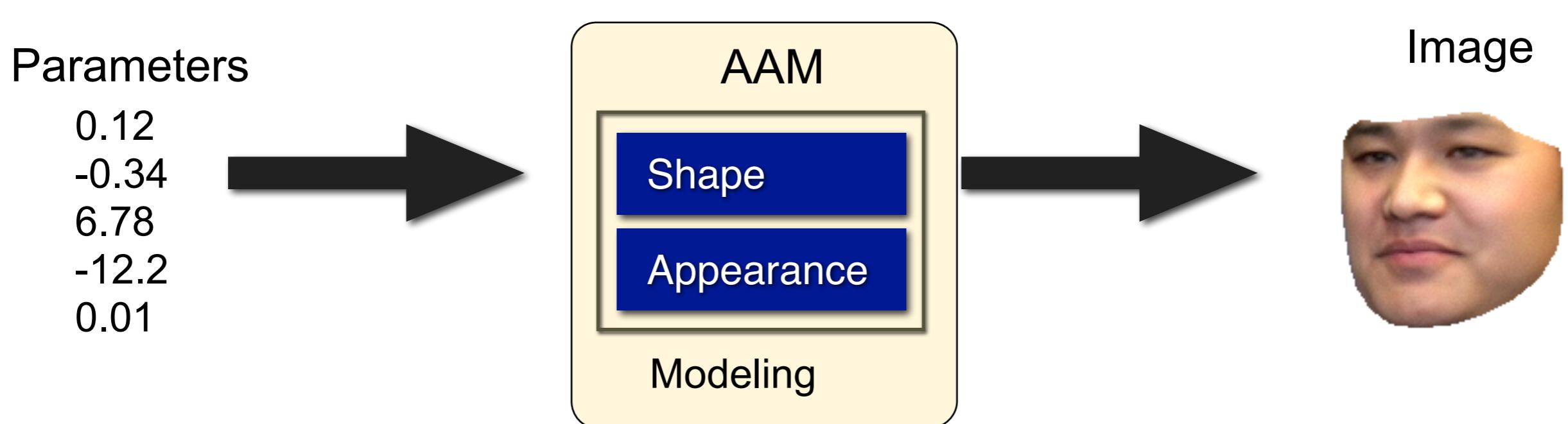
- Separately model object shape and appearance



(Matthews and Gross)

# Active Appearance Models

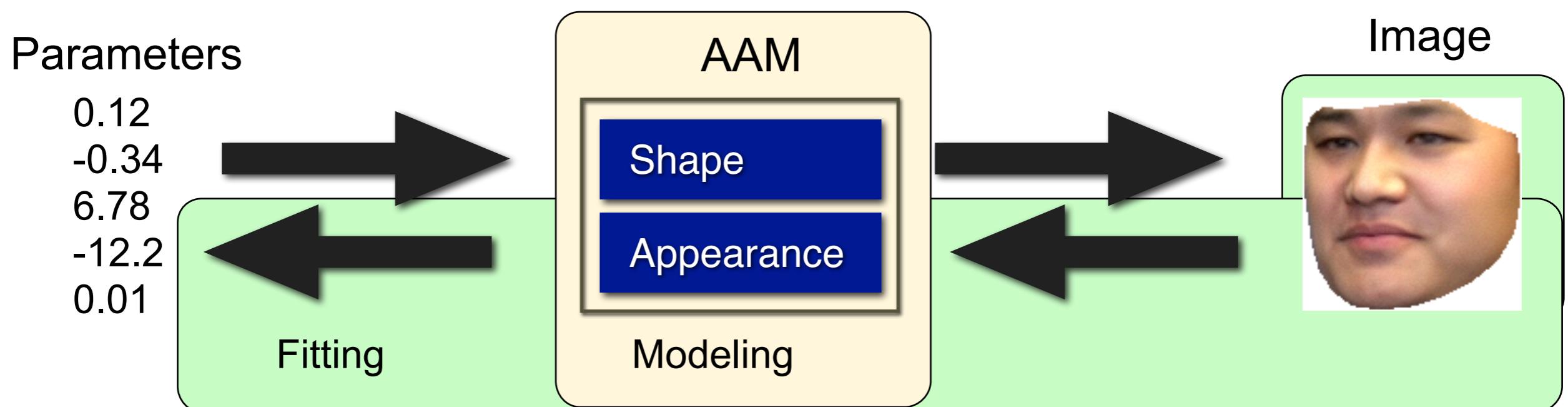
- Separately model object shape and appearance



(Matthews and Gross)

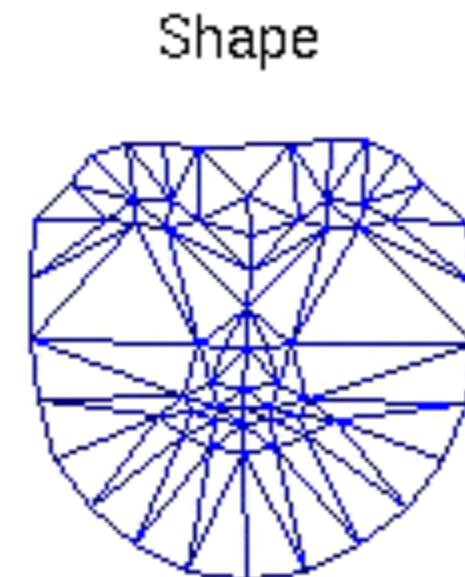
# Active Appearance Models

- Separately model object shape and appearance



(Matthews and Gross)

# Active Appearance Models

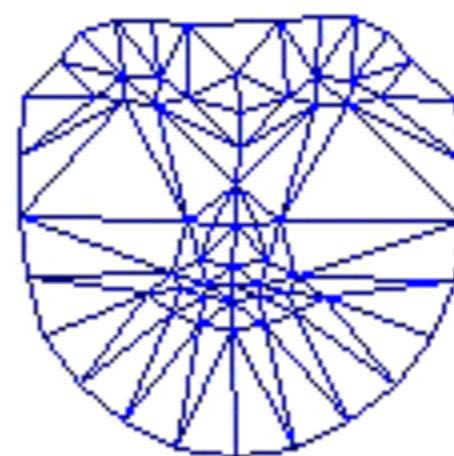


(Matthews and Gross)

# Active Appearance Models



Shape



Combined



(Matthews and Gross)

# AAM Overview

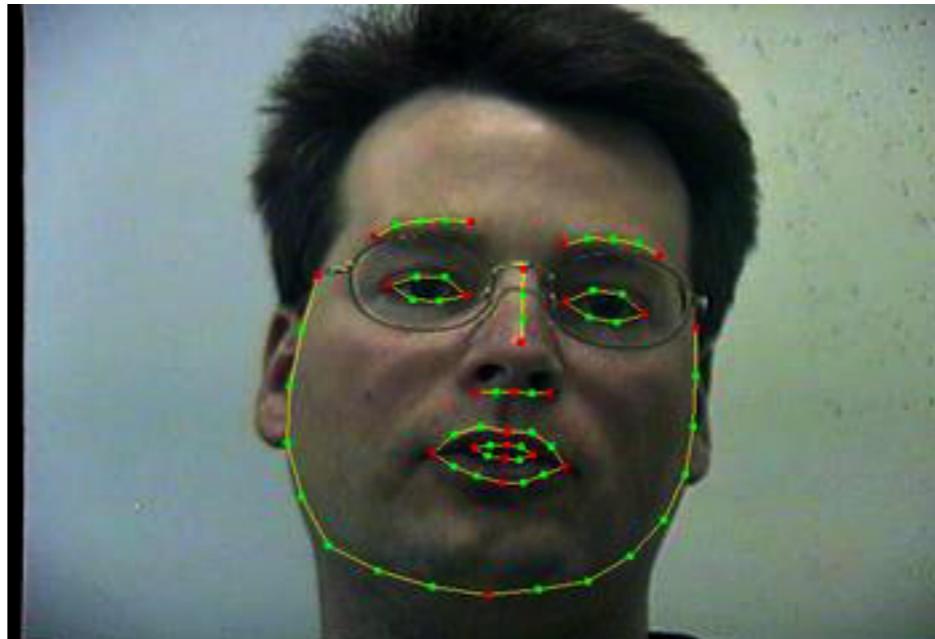
---

- 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

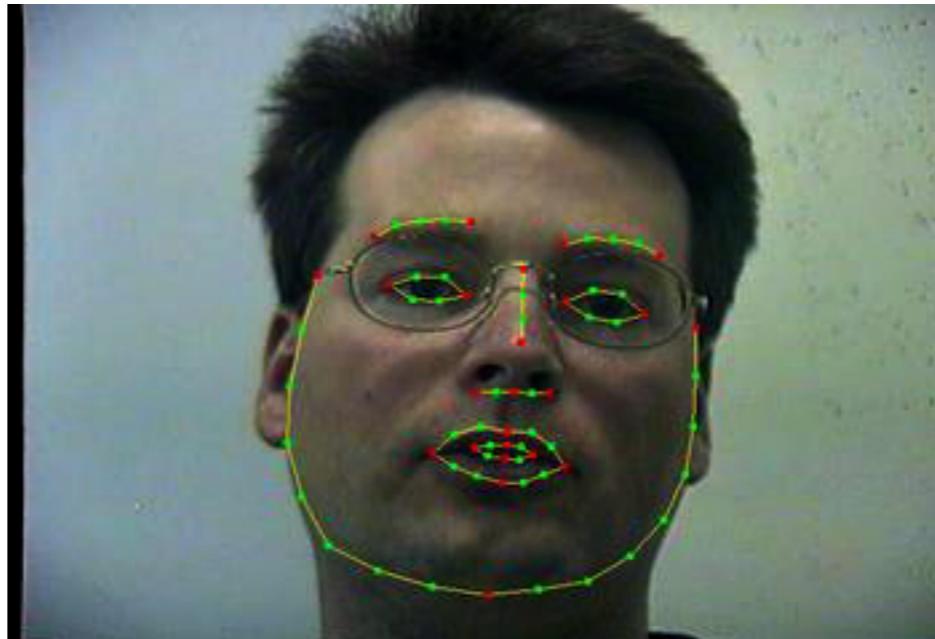
- Define a set of landmarks on a face that can be reliably located in an image
- Still open questions:
  - How many landmarks, which landmarks



(Matthews and Gross)

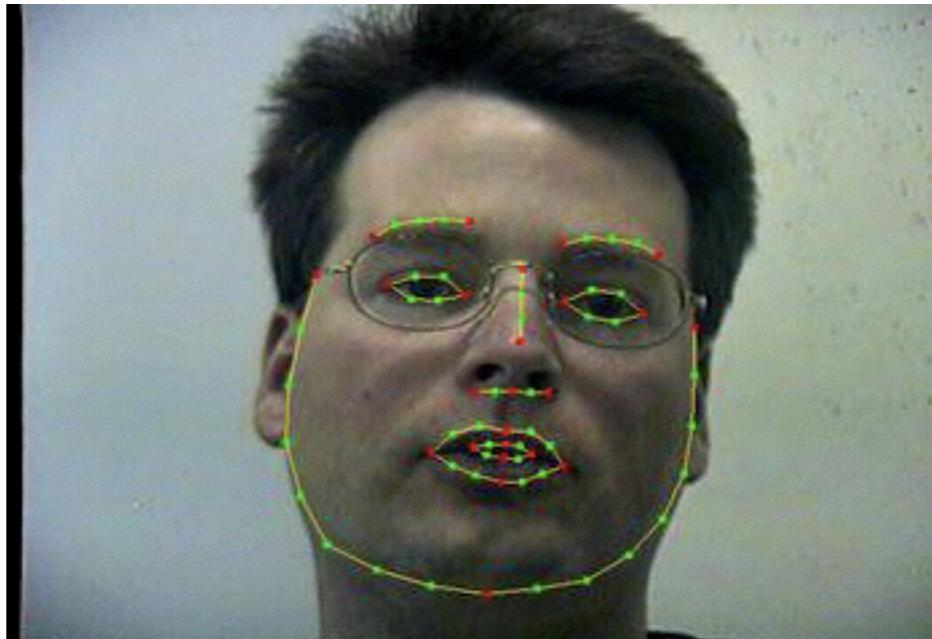
# Model Building - Definition

- 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



$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n)$$

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

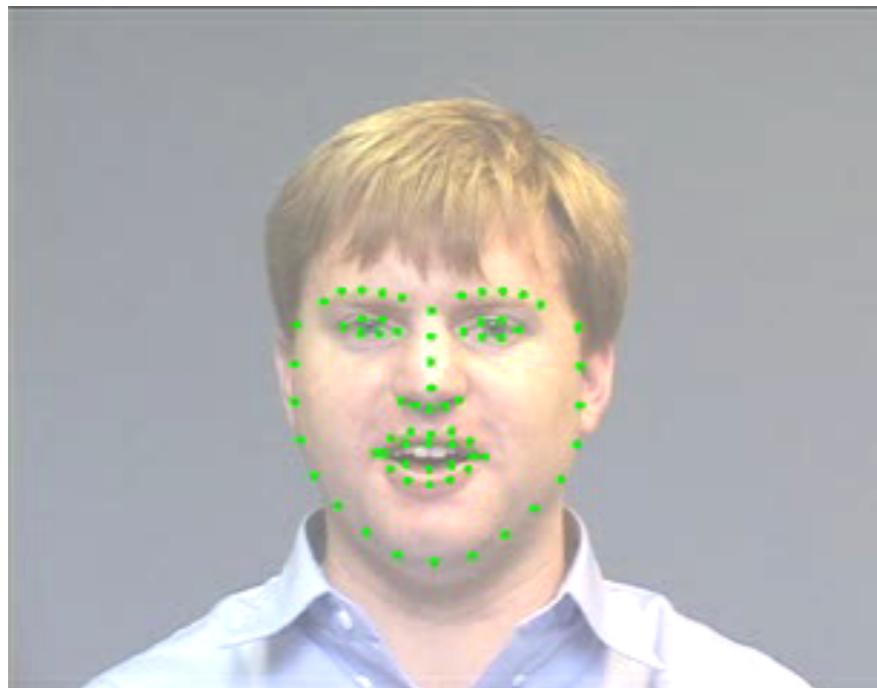
- Various sources of shape variations between individual images
  - Identity
  - Facial expression
  - Position in the image

Don't Care!

(Matthews and Gross)

# Shape Alignment

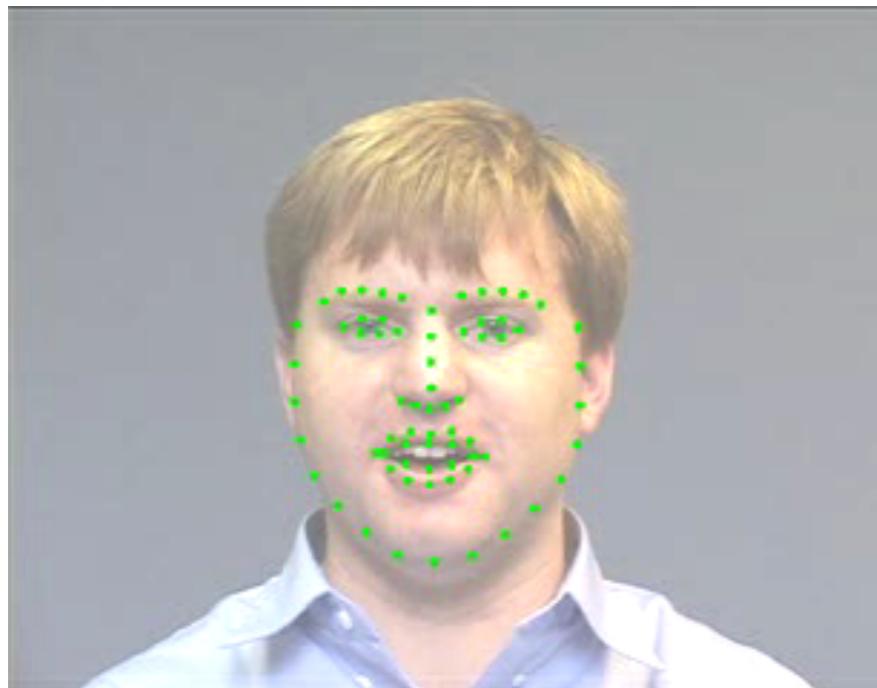
- Procrustes analysis
  - Removes variation due to predetermined shape transformation



(Matthews and Gross)

# Shape Alignment

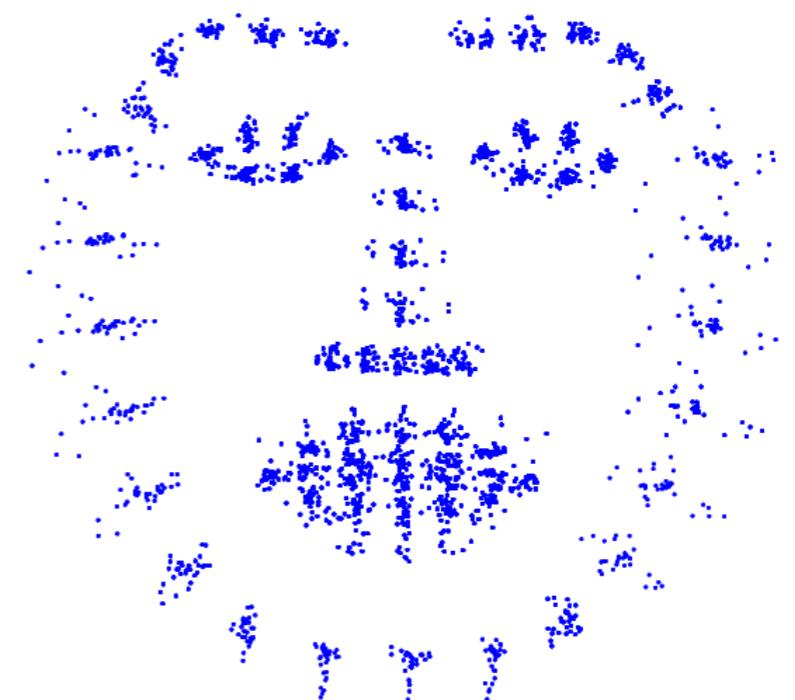
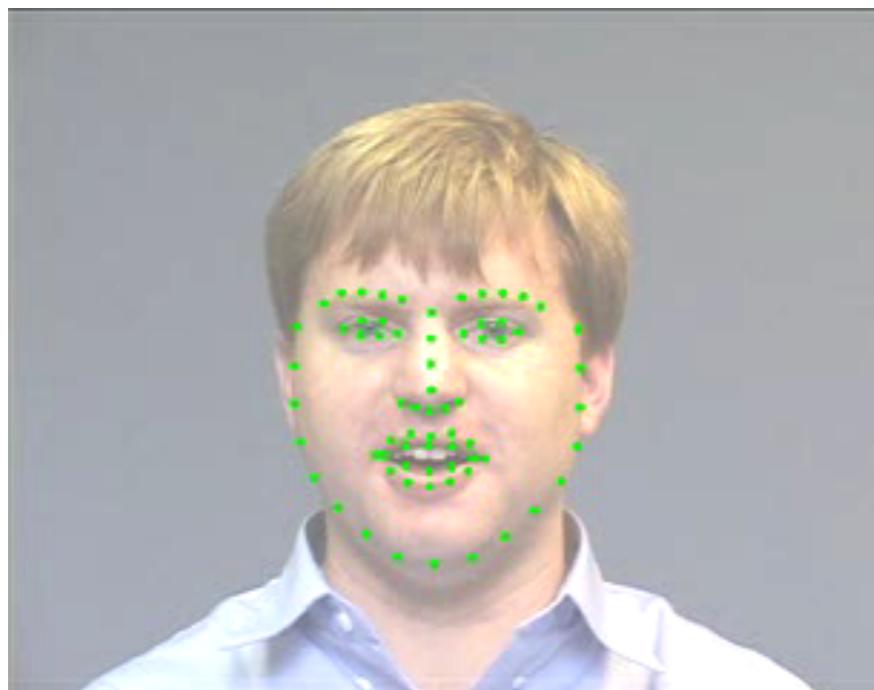
- Procrustes analysis
  - Removes variation due to predetermined shape transformation



(Matthews and Gross)

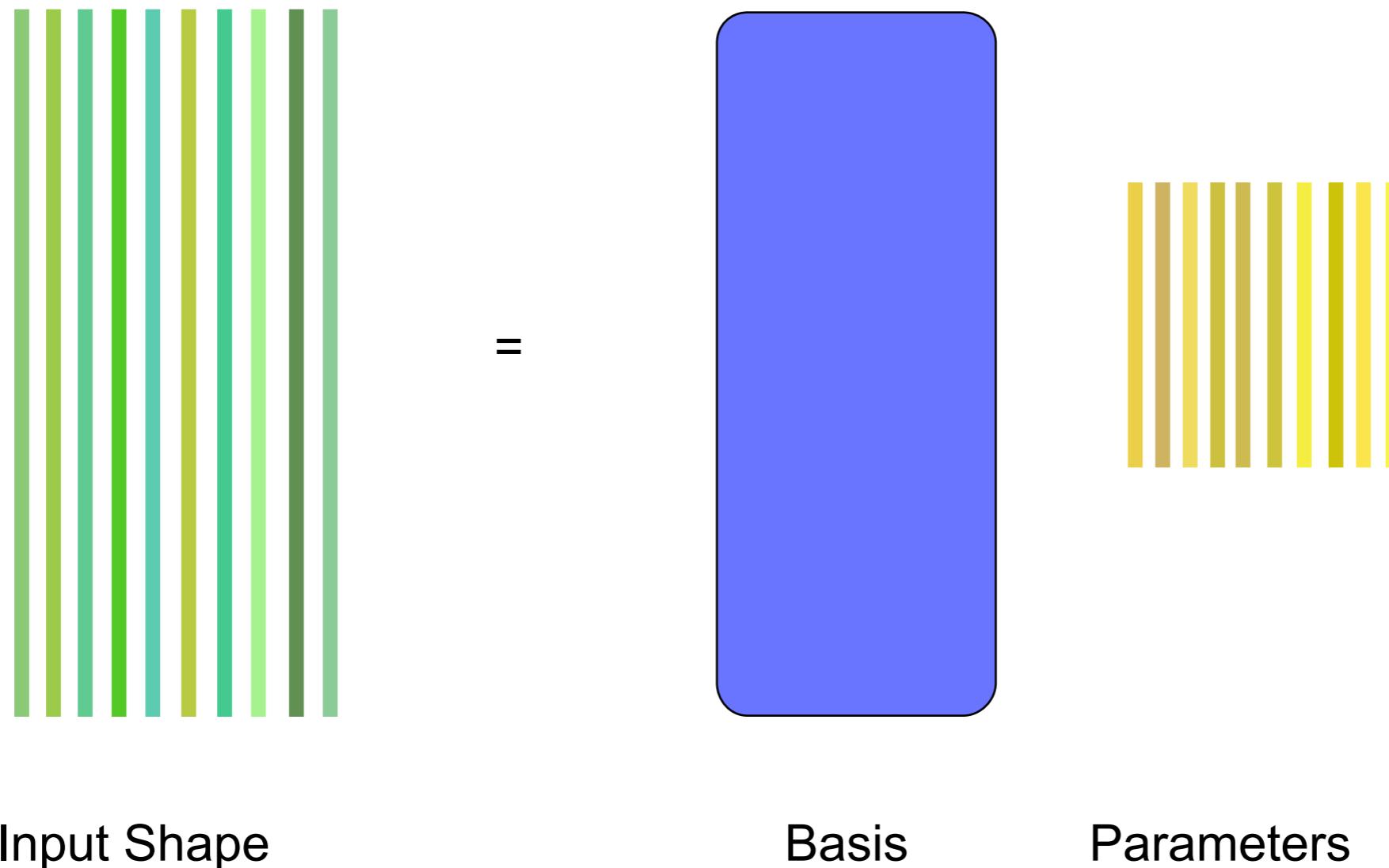
# Shape Alignment

- Procrustes analysis
  - Removes variation due to predetermined shape transformation



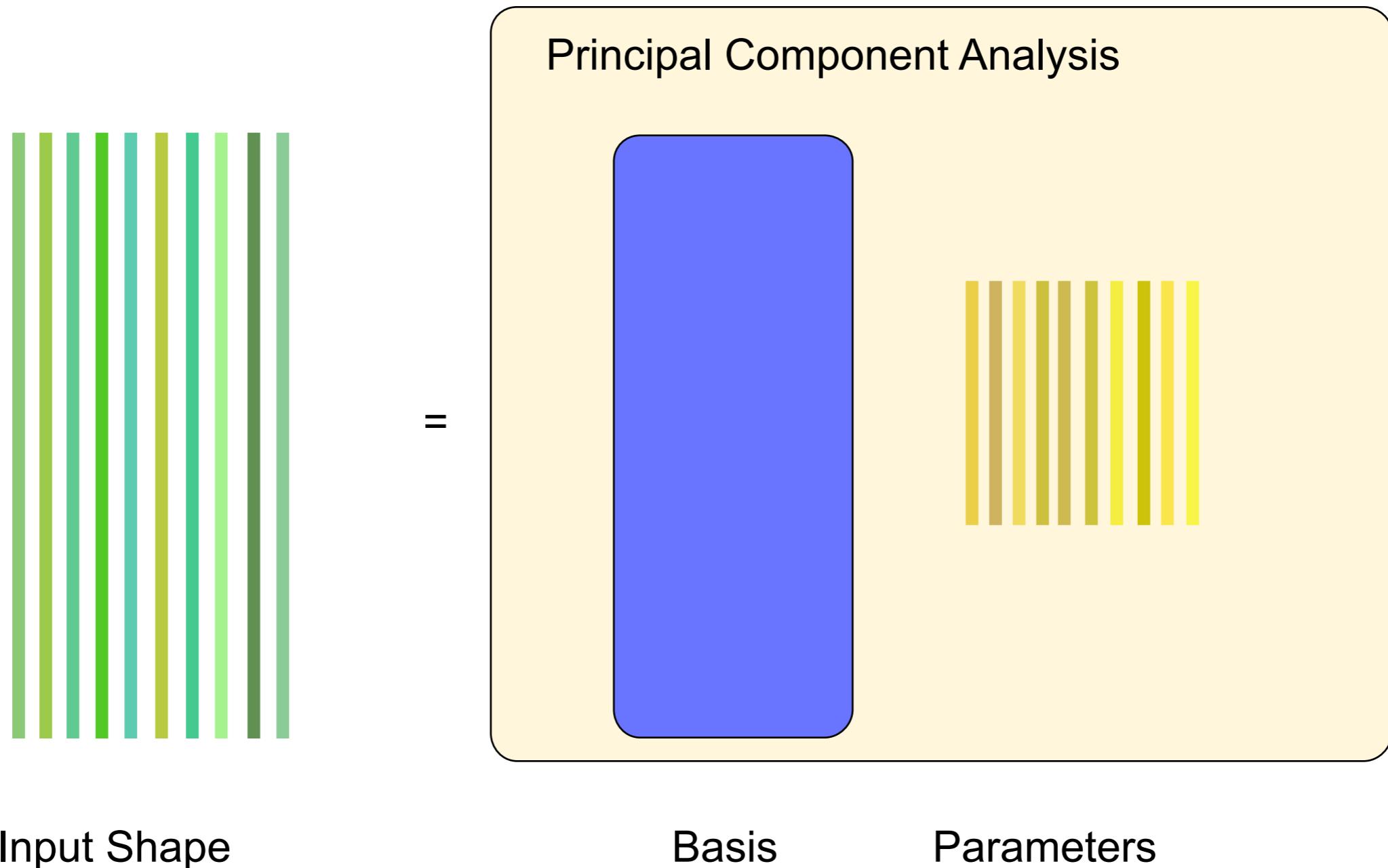
(Matthews and Gross)

# Building a Shape Model



(Matthews and Gross)

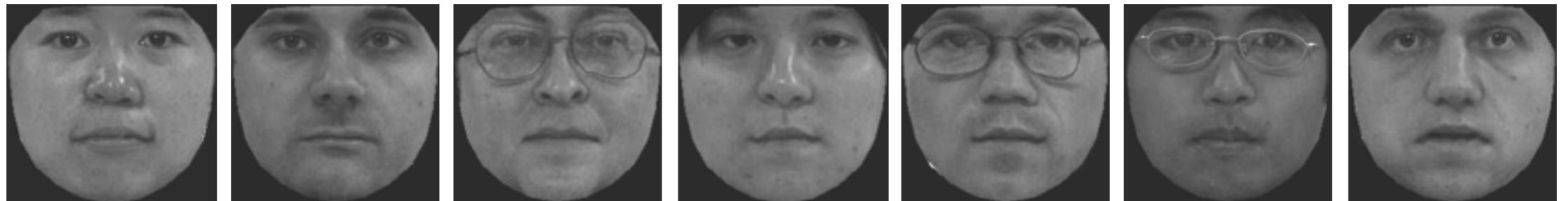
# Building a Shape Model



(Matthews and Gross)

# PCA on Appearance

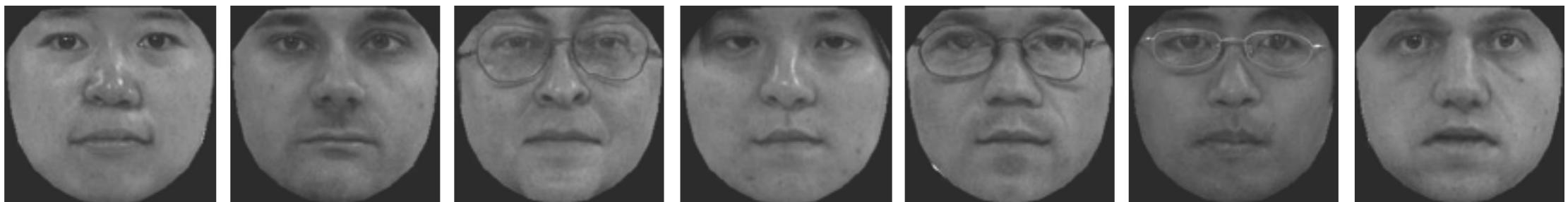
Original



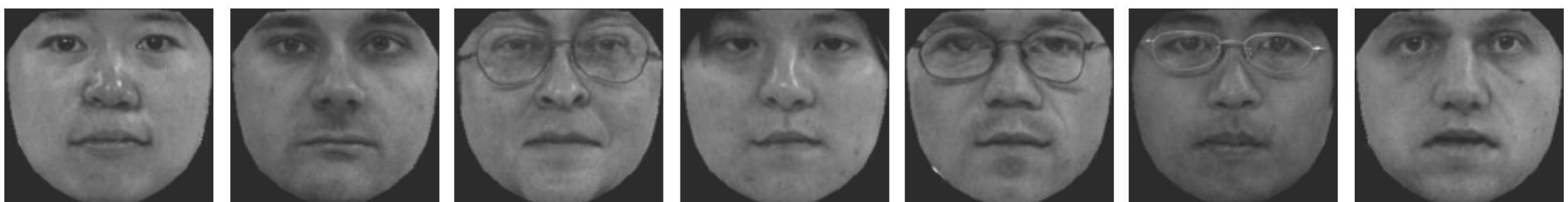
(Matthews and Gross)

# PCA on Appearance

Original



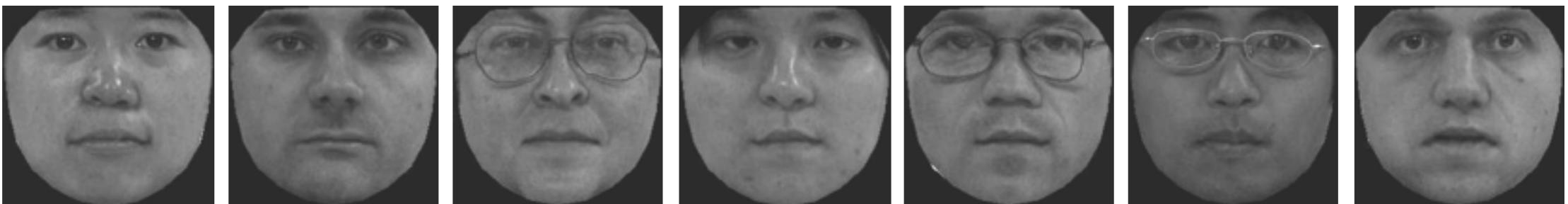
Full



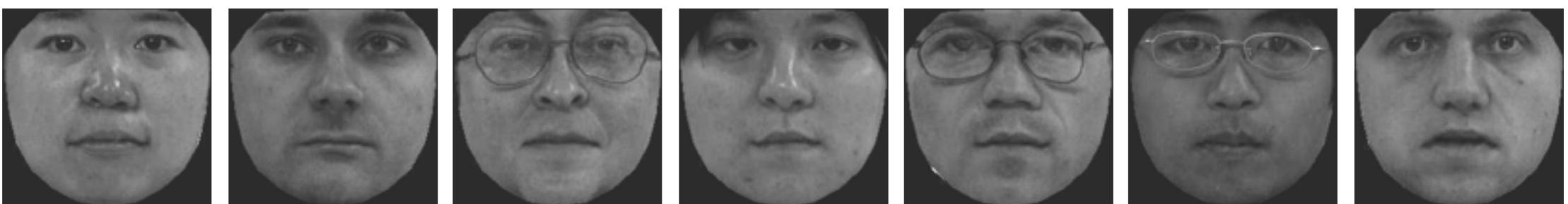
(Matthews and Gross)

# PCA on Appearance

Original



Full



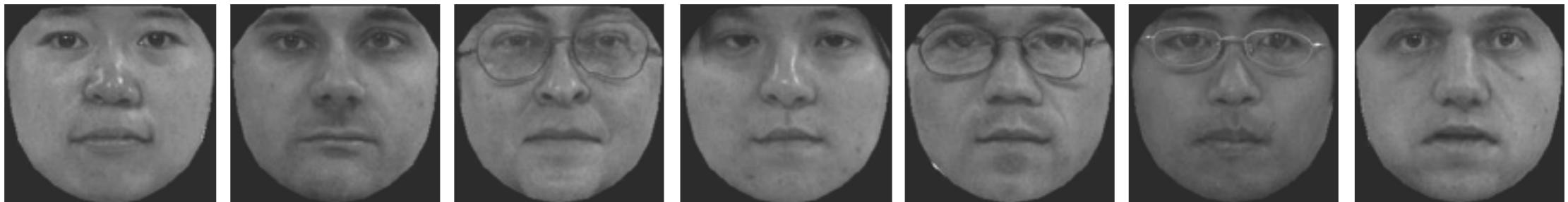
20 dim.



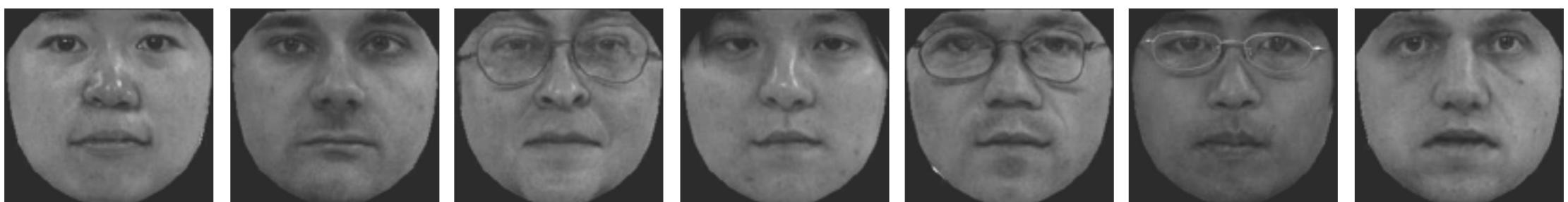
(Matthews and Gross)

# PCA on Appearance

Original



Full



20 dim.



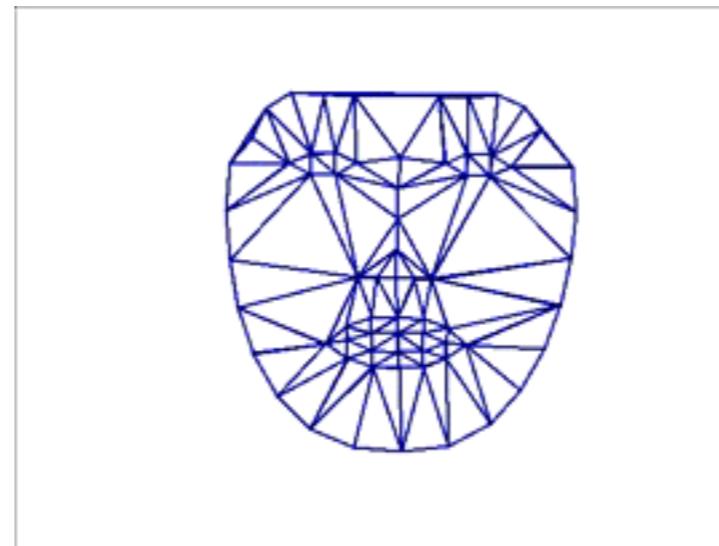
10 dim.



(Matthews and Gross)

# PCA Shape Model

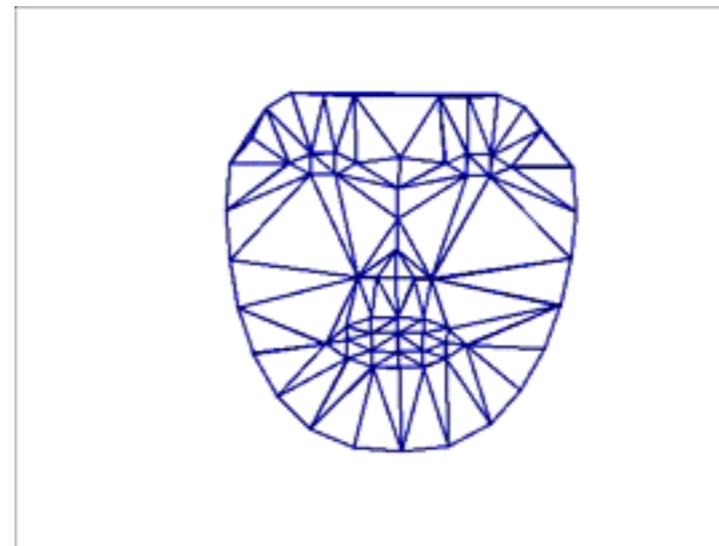
- Input:  
Bunch of aligned face shapes
- Output  
Mean shape and the direction of the largest variations  
(eigenvectors)



(Matthews and Gross)

# PCA Shape Model

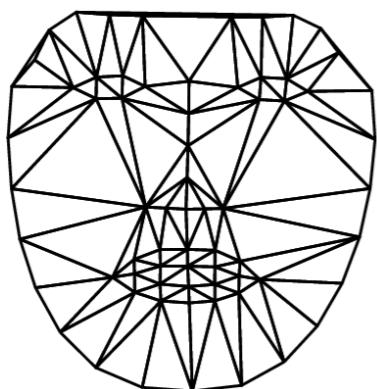
- Input:  
Bunch of aligned face shapes
- Output  
Mean shape and the direction of the largest variations  
(eigenvectors)



(Matthews and Gross)

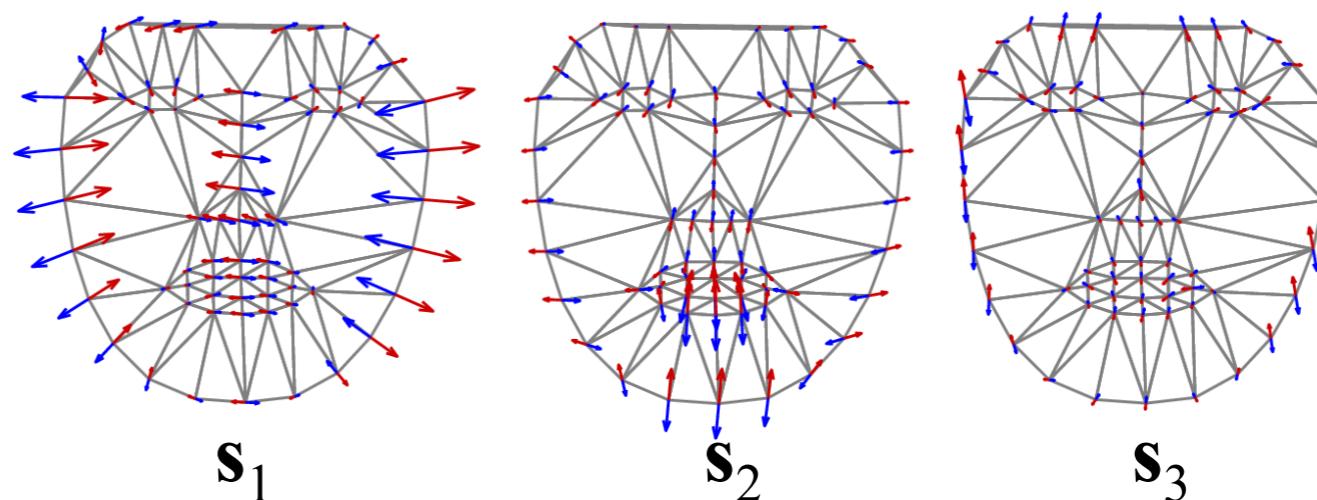
# PCA Shape Model

Mean Face



$\mathbf{s}_0$

First 3 Shape Modes



Shape Model

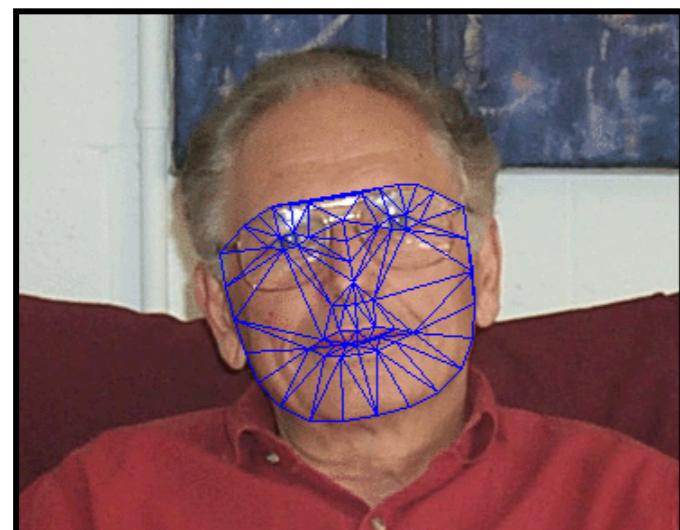
$$\mathcal{W}(\mathbf{z}; \mathbf{p}) = \mathbf{s}_0 + \sum_{k=1}^K p_k \mathbf{s}_k$$

Shape Parameters

$$\mathbf{p} = [p_1, p_2, \dots, p_K]^T$$

(Matthews and Gross)

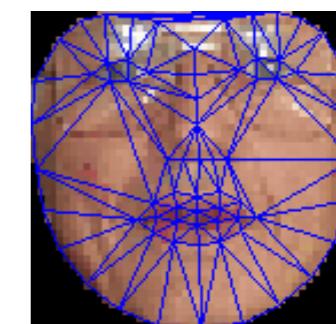
# Construction of Appearance Model



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)

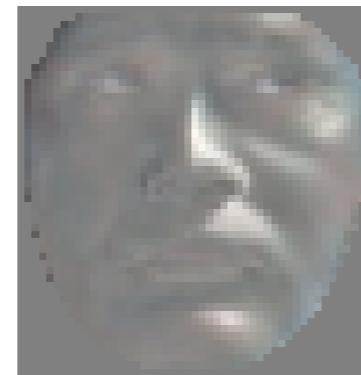
# AAM Appearance Model

Mean Face



$A_0(\mathbf{z})$

First 3 Appearance Modes



$A_1(\mathbf{z})$



$A_2(\mathbf{z})$



$A_3(\mathbf{z})$

Appearance Model

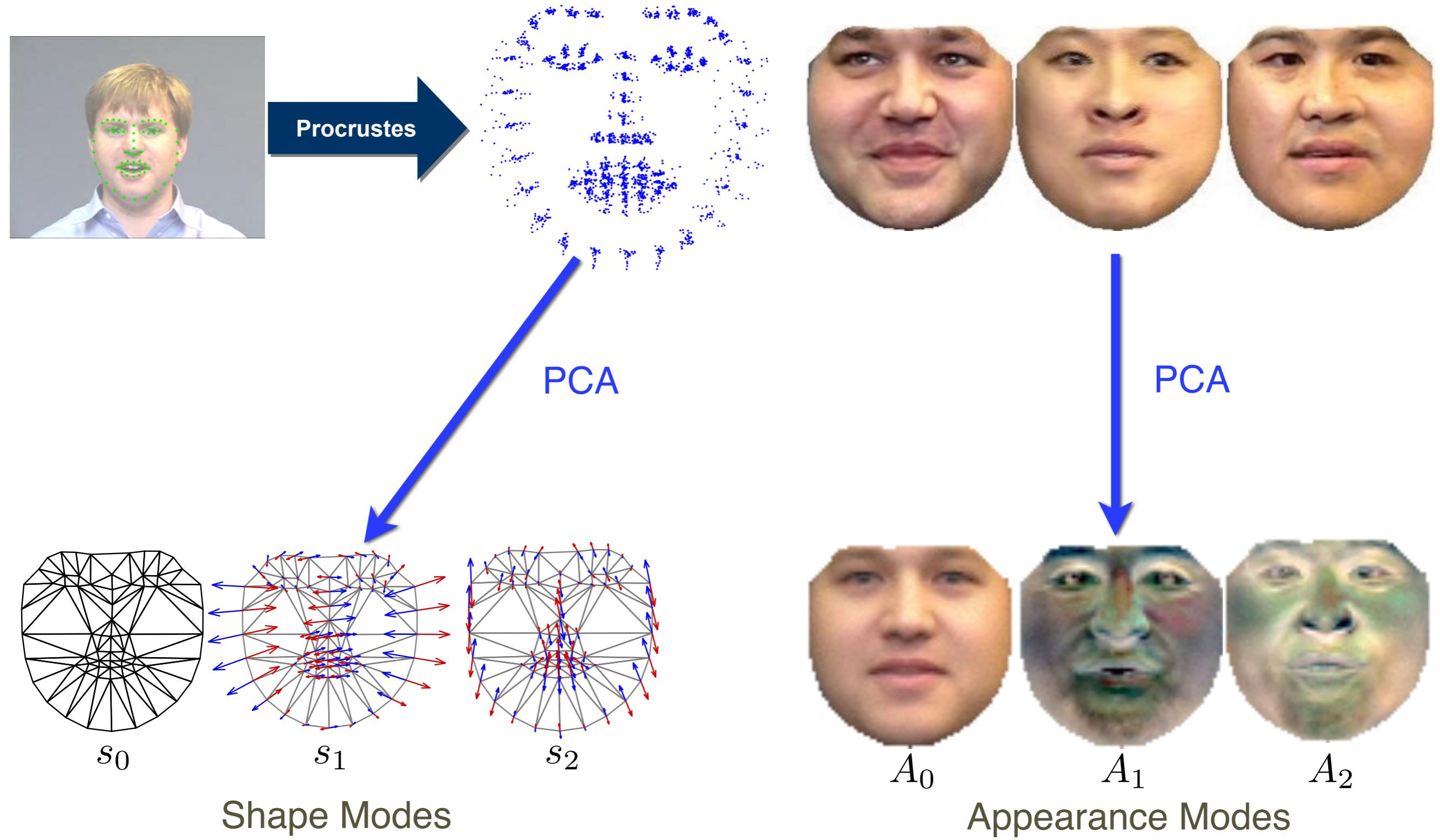
$$T(\mathbf{z}) = A_0(\mathbf{z}) + \sum_{m=1}^M \lambda_i A_m(\mathbf{z})$$

Appearance Parameters

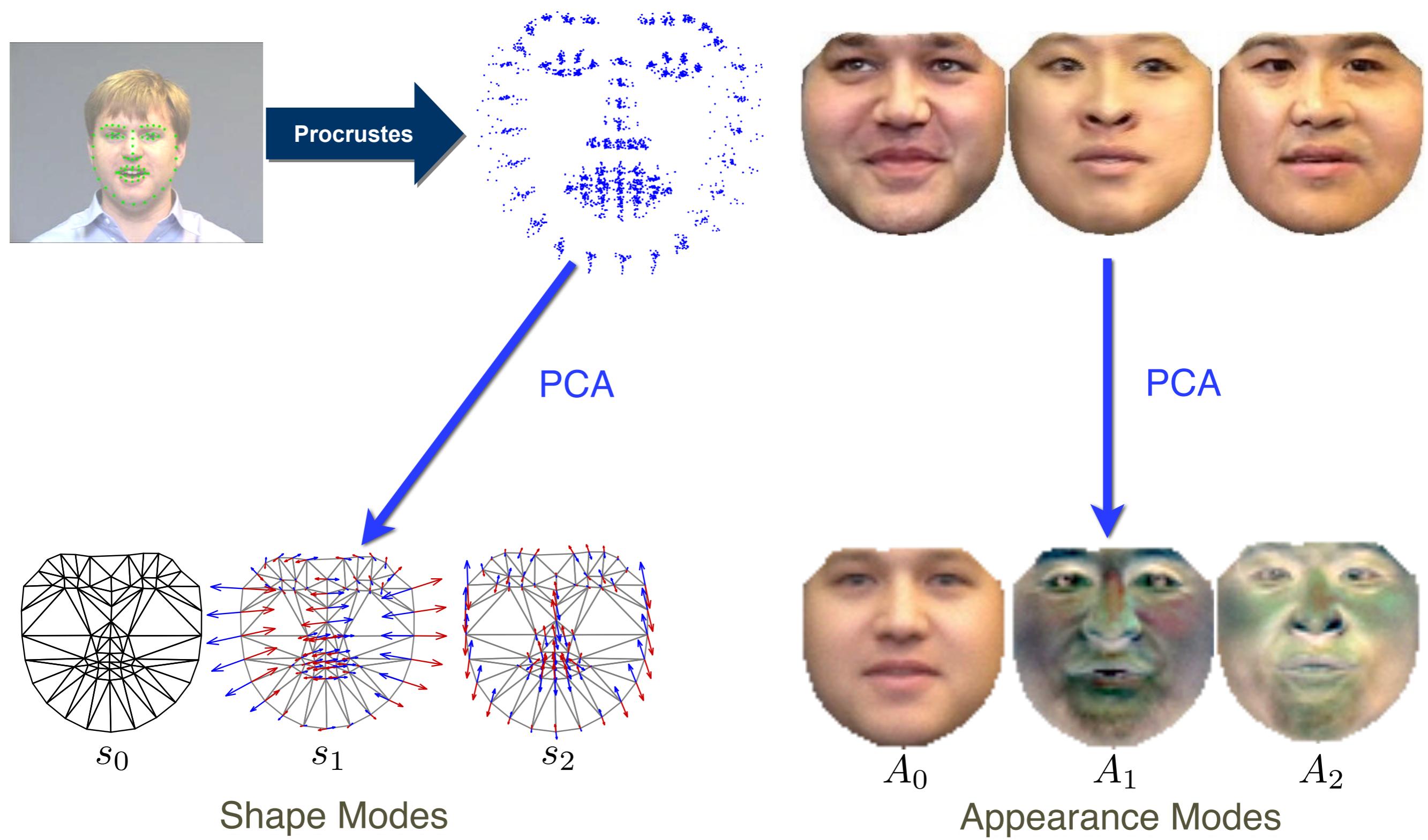
$$(\lambda_1, \lambda_2, \dots, \lambda_n)^T$$

(Matthews and Gross)

# AAM Model Building Overview



# AAM Model Building Overview



# Model Instantiation

$$\text{Appearance}, T(\mathbf{z}) = A_0(\mathbf{z}) + 3559A_1(\mathbf{z}) + 351A_2(\mathbf{z}) - 256A_3(\mathbf{z}) \dots$$

(Matthews and Gross)

# Model Instantiation

$$\text{Appearance}, T(\mathbf{z}) = A_0(\mathbf{z}) + 3559A_1(\mathbf{z}) + 351A_2(\mathbf{z}) - 256A_3(\mathbf{z}) \dots$$



$$\text{Shape, } \mathbf{z} = \mathbf{s}_0 - 54\mathbf{s}_1 + 10\mathbf{s}_2 - 9.1\mathbf{s}_3 \dots$$

(Matthews and Gross)

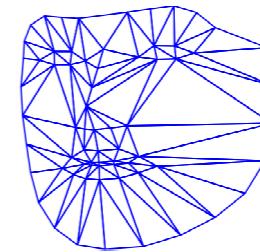
# Model Instantiation

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

$$\mathcal{W}(\mathbf{z}; \mathbf{p}) = \mathbf{s}_0 + \sum_{k=1}^K p_k \mathbf{s}_k$$



Warp



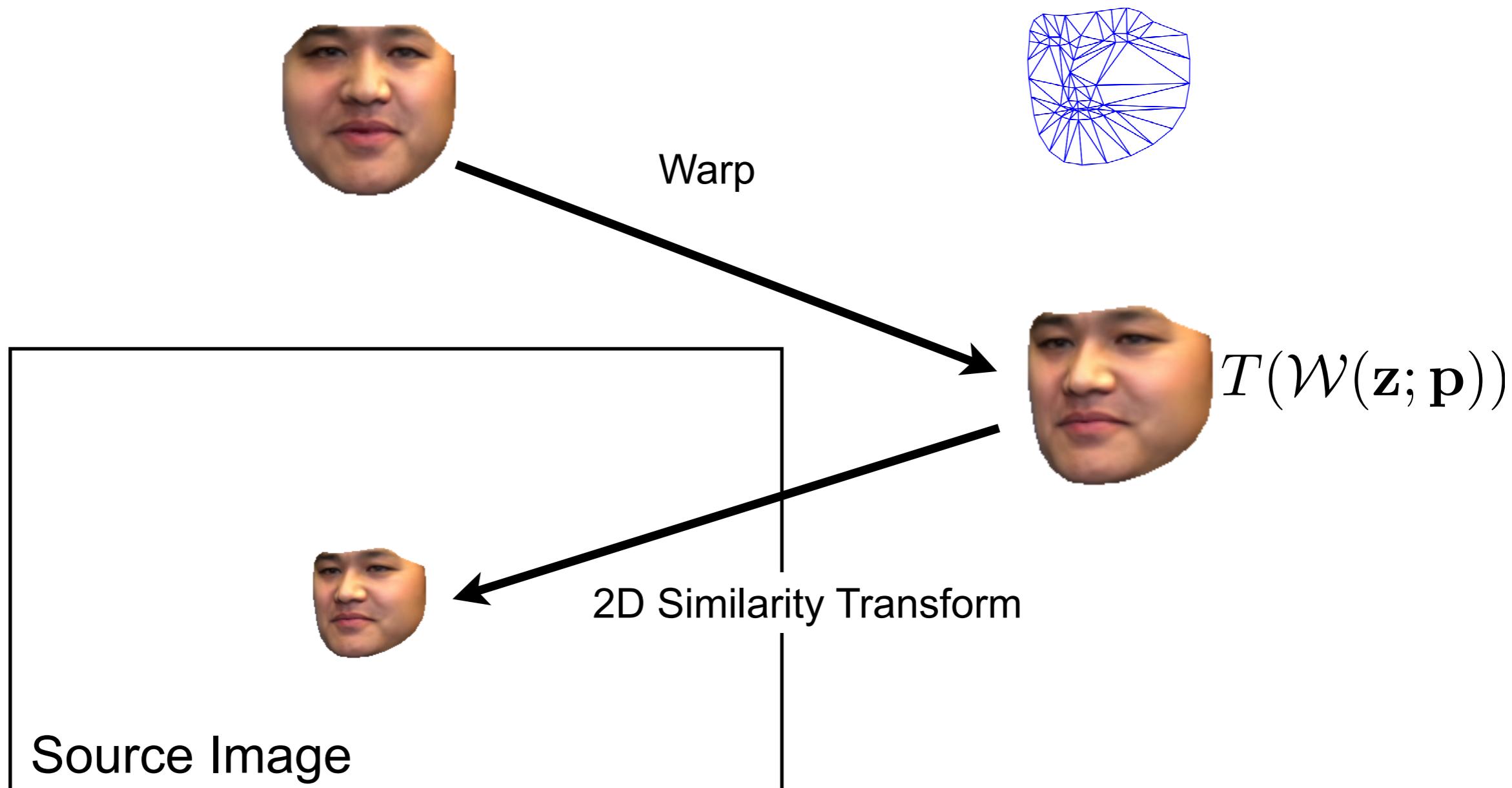
$T(\mathcal{W}(\mathbf{z}; \mathbf{p}))$

(Matthews and Gross)

# Model Instantiation

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

$$\mathcal{W}(\mathbf{z}; \mathbf{p}) = \mathbf{s}_0 + \sum_{k=1}^K p_k \mathbf{s}_k$$



(Matthews and Gross)

# A Parameterized Face

---

|



(Matthews and Gross)

# A Parameterized Face

---

|



(Matthews and Gross)

# Things to Note

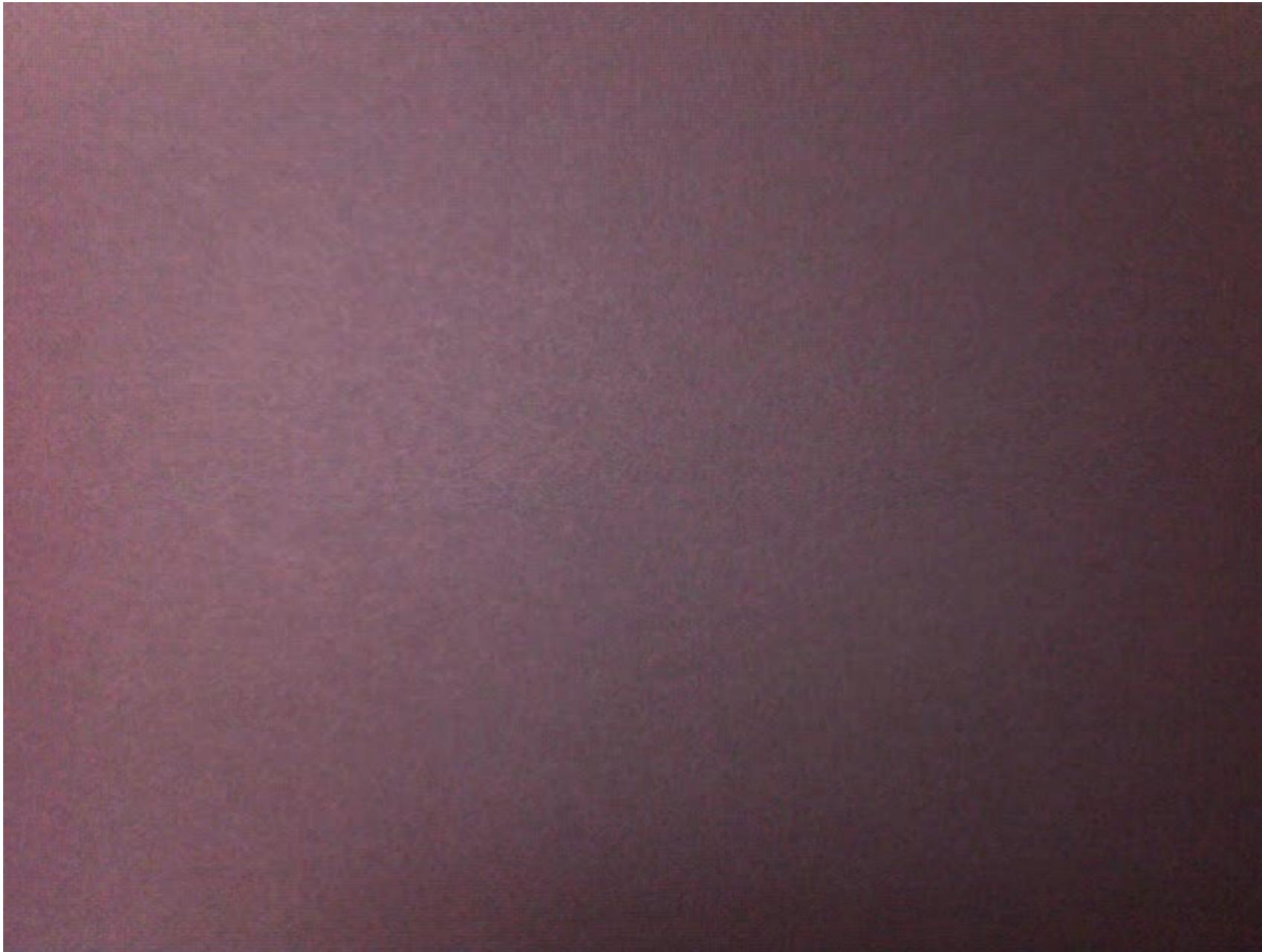
---

- 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

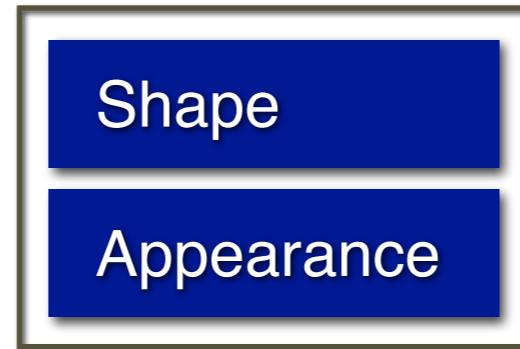
---

- 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

AAM



Determine the best model parameters to reconstruct  
the image

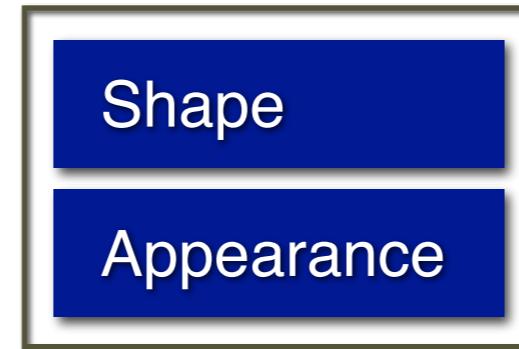
(Matthews and Gross)

# Active Appearance Models

Parameters

0.12  
-0.34  
6.78  
-12.2  
0.01

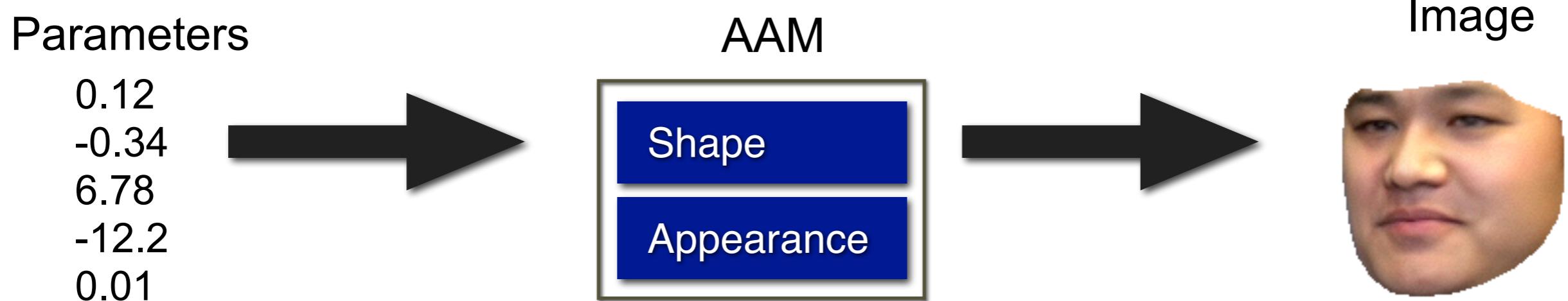
AAM



Determine the best model parameters to reconstruct  
the image

(Matthews and Gross)

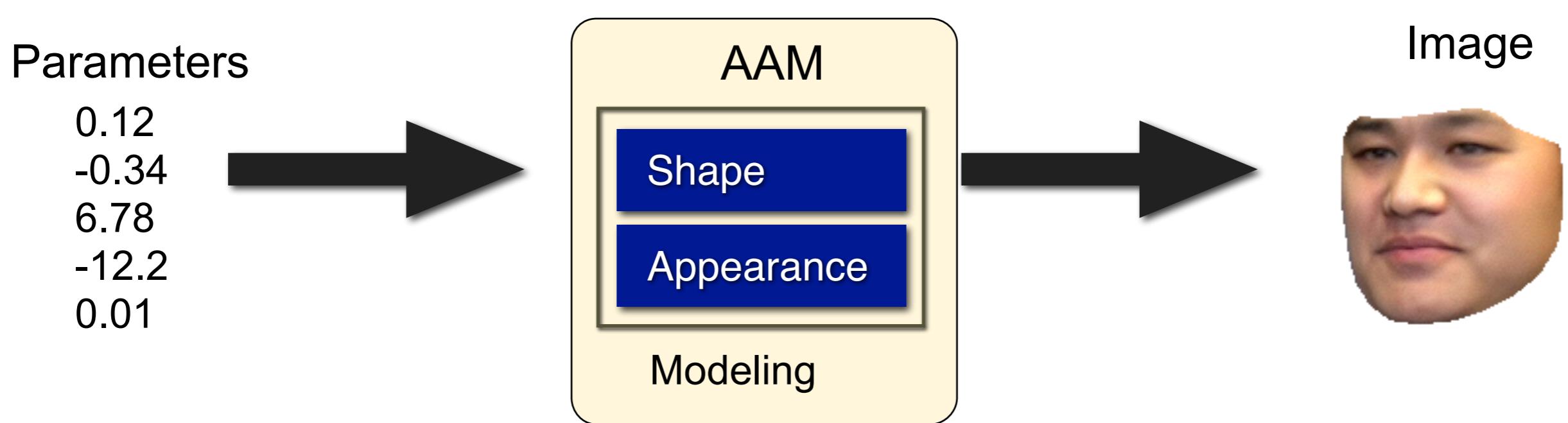
# Active Appearance Models



Determine the best model parameters to reconstruct  
the image

(Matthews and Gross)

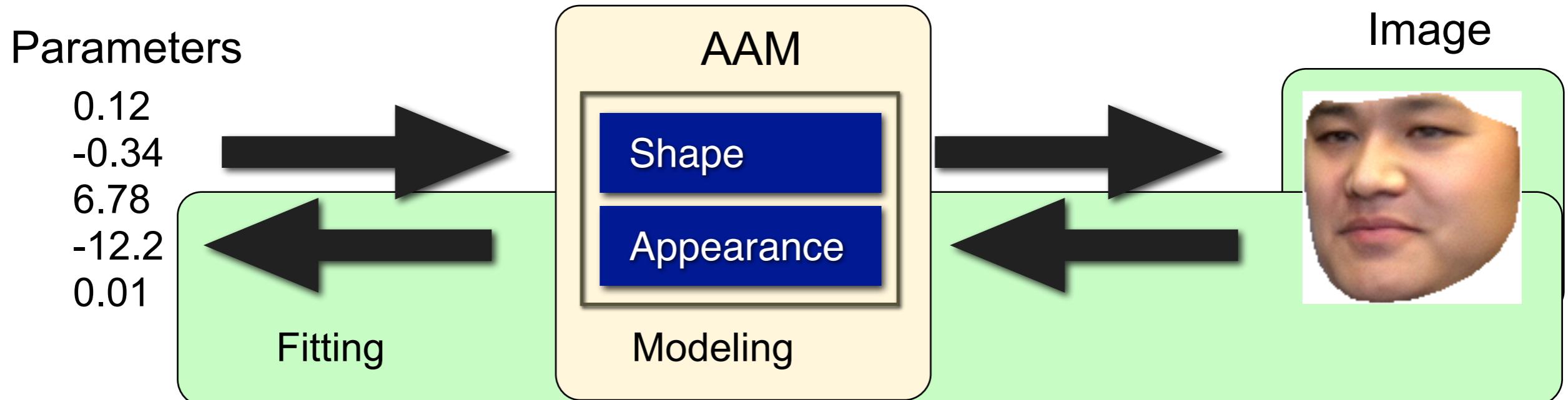
# Active Appearance Models



Determine the best model parameters to reconstruct  
the 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 do the fitting?
- Turns out, we can still use the “simultaneous” extension of the LK algorithm,

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

- where,

$$\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}} = [\mathbf{s}_1, \dots, \mathbf{s}_K] \quad \mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}}$$

# AAM - Fitting “Inverse Composition”

---

- Even faster if we use the inverse composition (IC) extension as well as “projecting out” the appearance,

$$\arg \min_{\Delta \mathbf{p}} \| T(\mathbf{z}) + \mathbf{J}_{ic} \Delta \mathbf{p} - I(\mathcal{W}(\mathbf{z}; \mathbf{p})) \|_{\text{null}(\mathbf{A})}^2$$

- where,

$$\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}} = [\mathbf{s}_1, \dots, \mathbf{s}_K] \quad \mathbf{J}_{ic} = \frac{\partial T(\mathcal{W}(\mathbf{z}; \mathbf{0})^T)}{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}}$$

# IC AAM Fitting

- Runs at 230 Hz on a 3.2GHz PC

Input



Overlaid  
Model  
Instance



Shape  
Overlaid



Rendered  
Model  
Instance



(Matthews and Gross)

# IC AAM Fitting

- Runs at 230 Hz on a 3.2GHz PC

Input



Overlaid  
Model  
Instance



Shape  
Overlaid



Rendered  
Model  
Instance



(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)

# Not All Is Peachy



Original model does not handle occlusion well

(Matthews and Gross)

# Not All Is Peachy



Original model does not handle occlusion well

(Matthews and Gross)

# AAM - Fitting “Robust Error Function”

---

- To handle occlusions we can employ the robust error function,

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

- where,

$$\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}} = [\mathbf{s}_1, \dots, \mathbf{s}_K] \quad \mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}}$$

# AAMs with Occlusion Modeling



(Matthews and Gross)

# AAMs with Occlusion Modeling



(Matthews and Gross)

# Applications

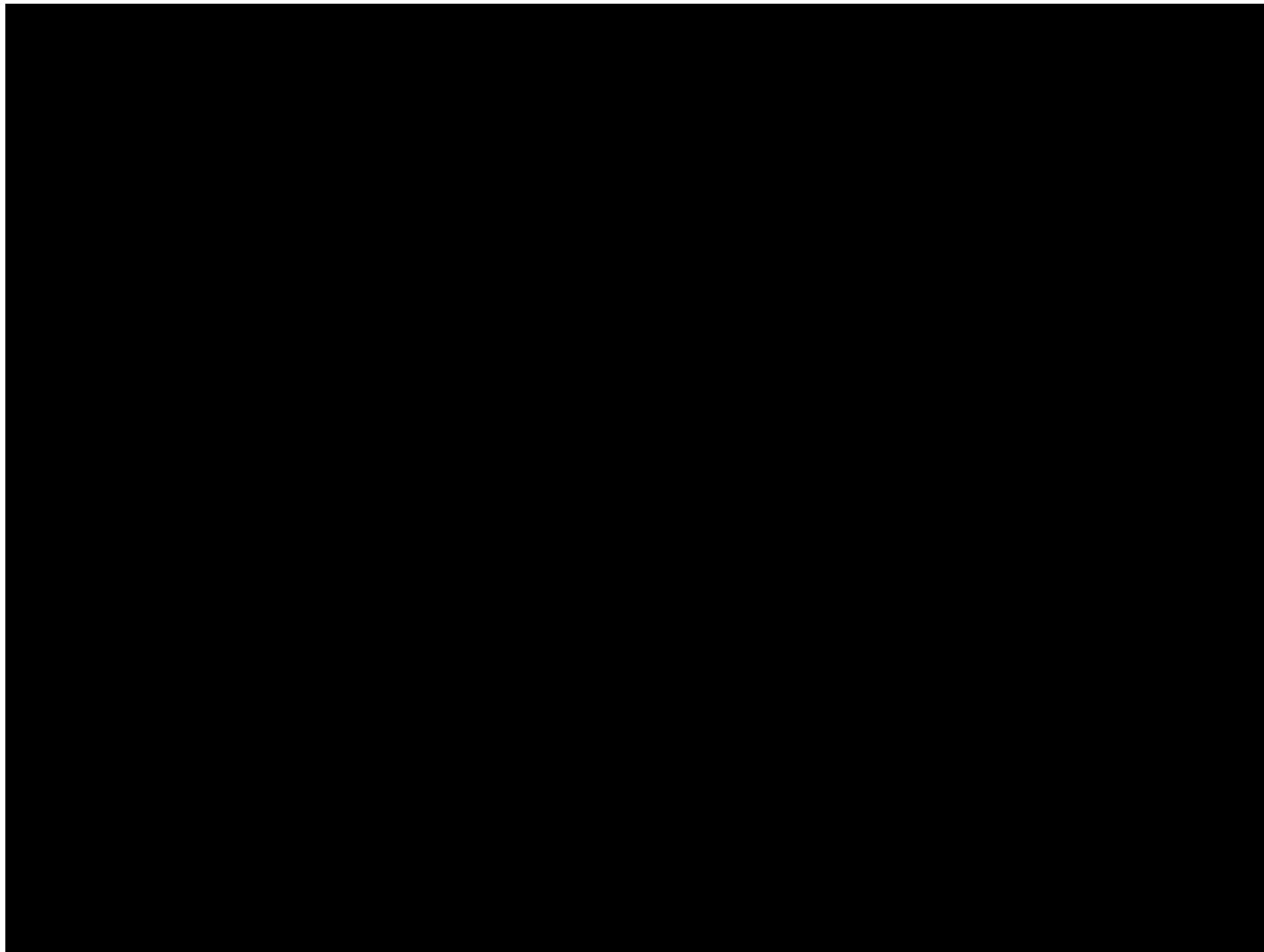
---

- **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

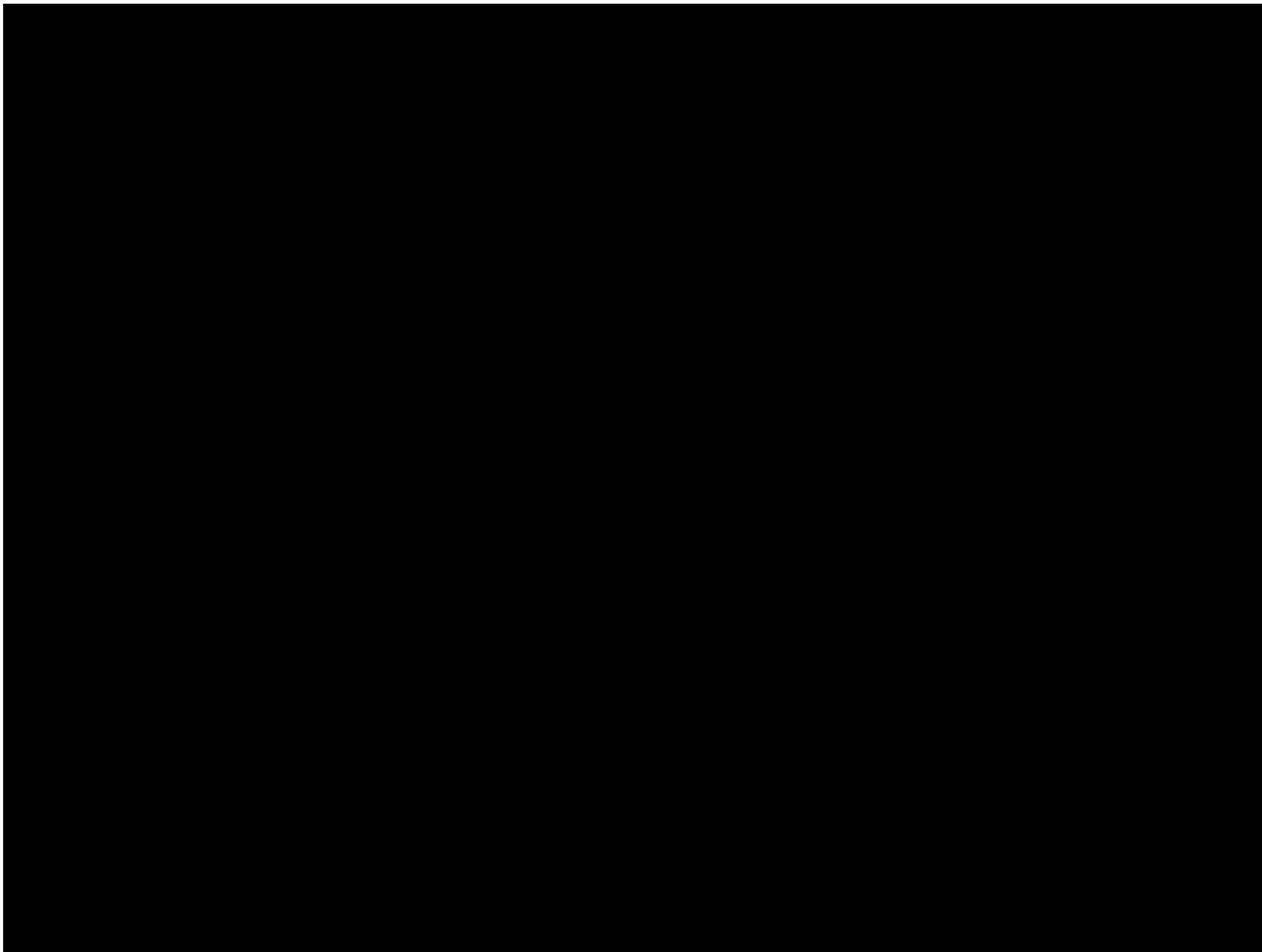
- Mouse replacement



(Matthews and Gross)

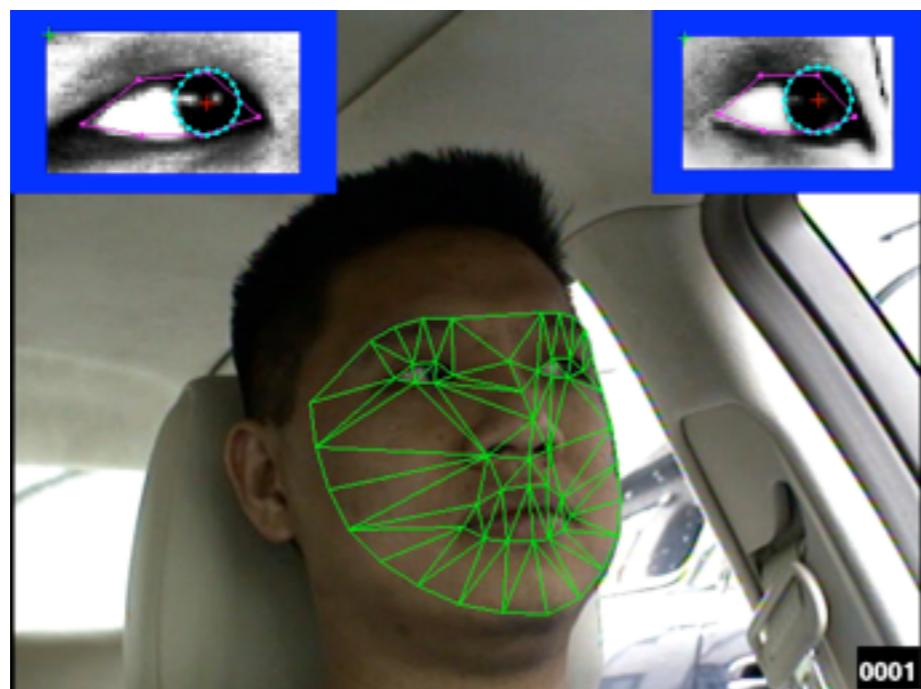
# User Interfaces: Head Pose

- Mouse replacement

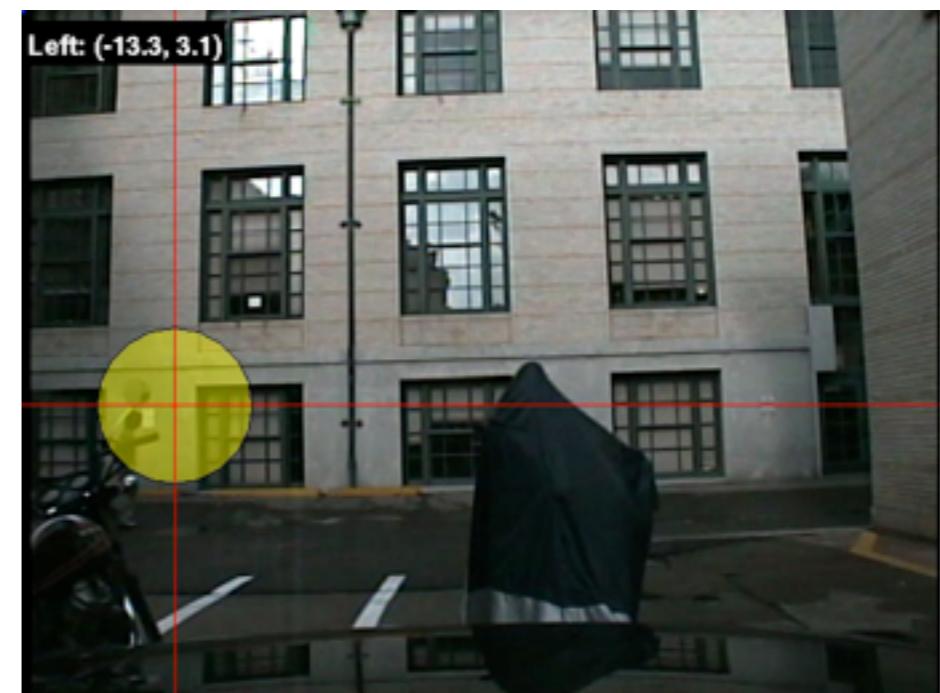


(Matthews and Gross)

# User Interfaces: Gaze Tracking



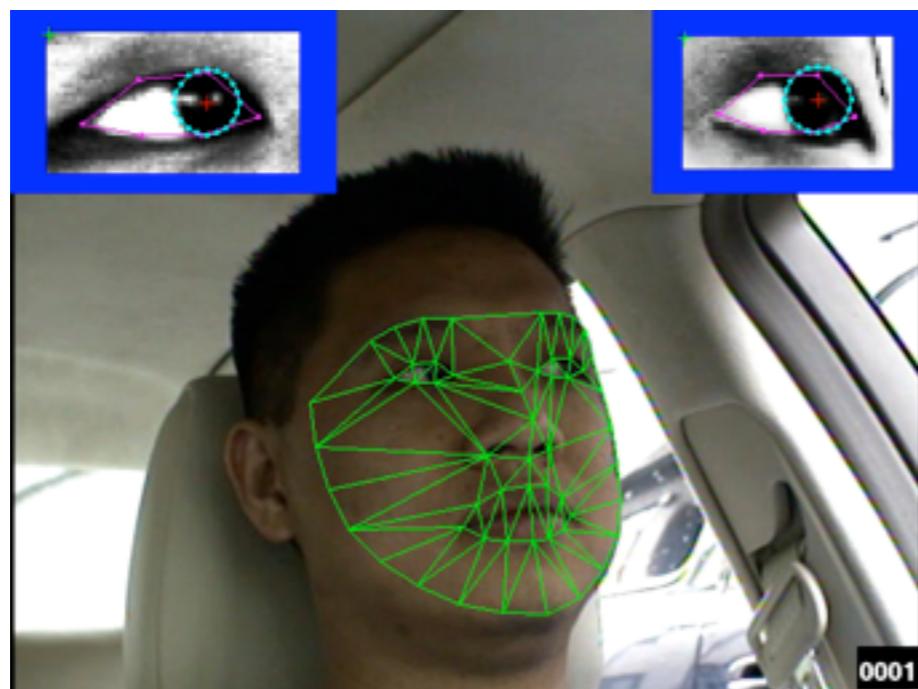
Driver Camera



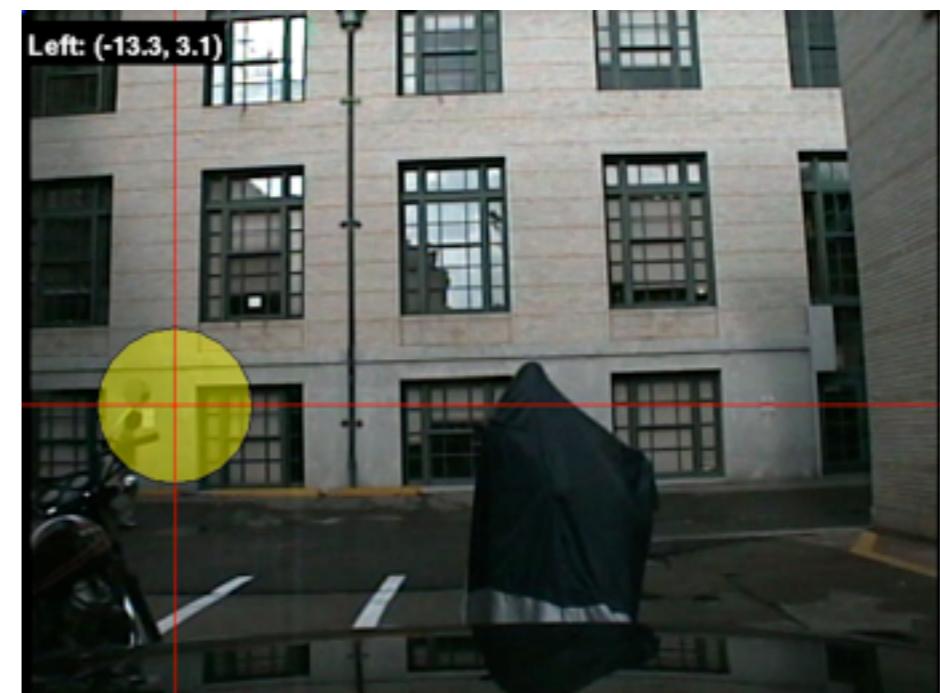
Exterior View Camera

(Matthews and Gross)

# User Interfaces: Gaze Tracking



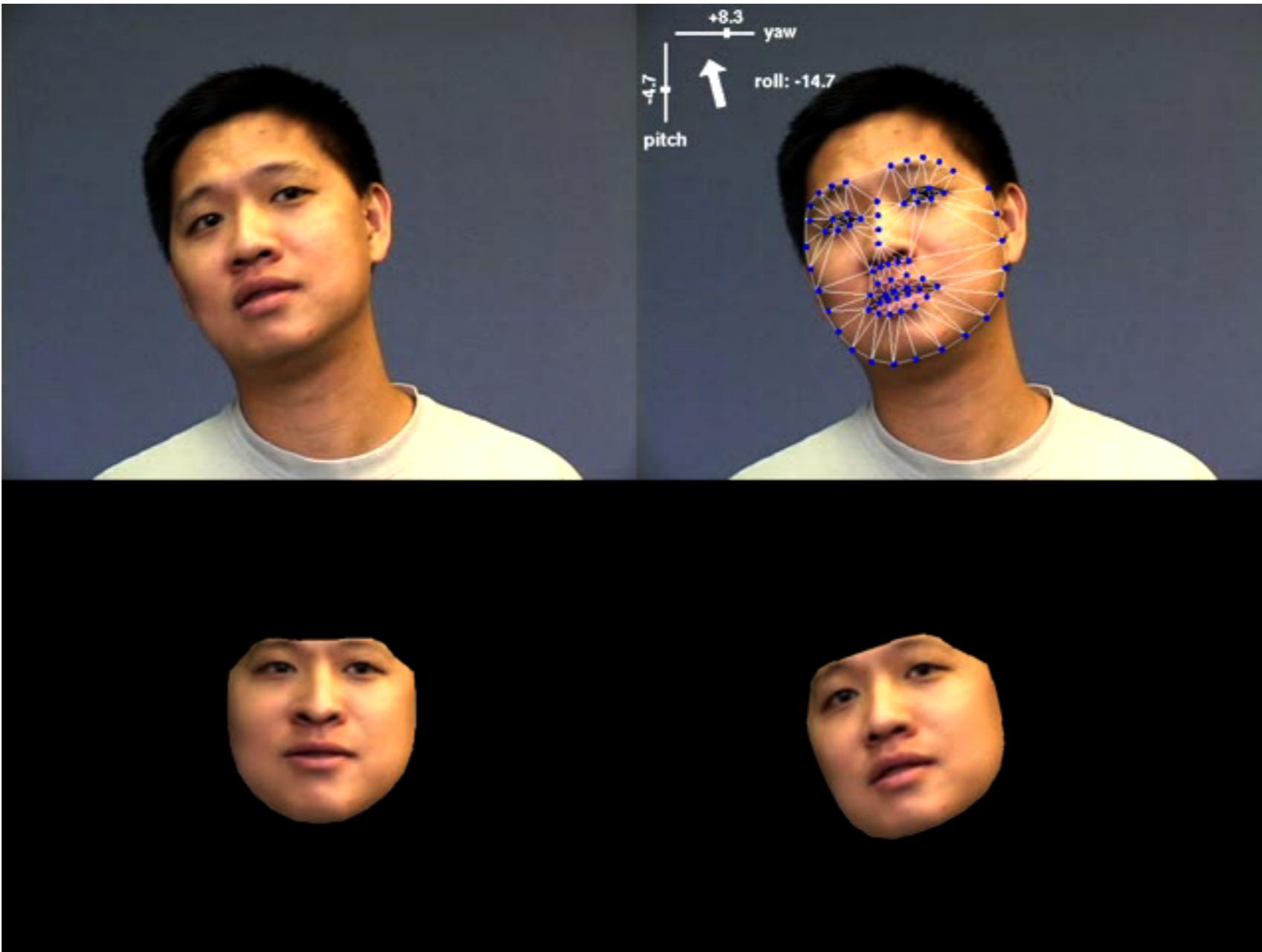
Driver Camera



Exterior View Camera

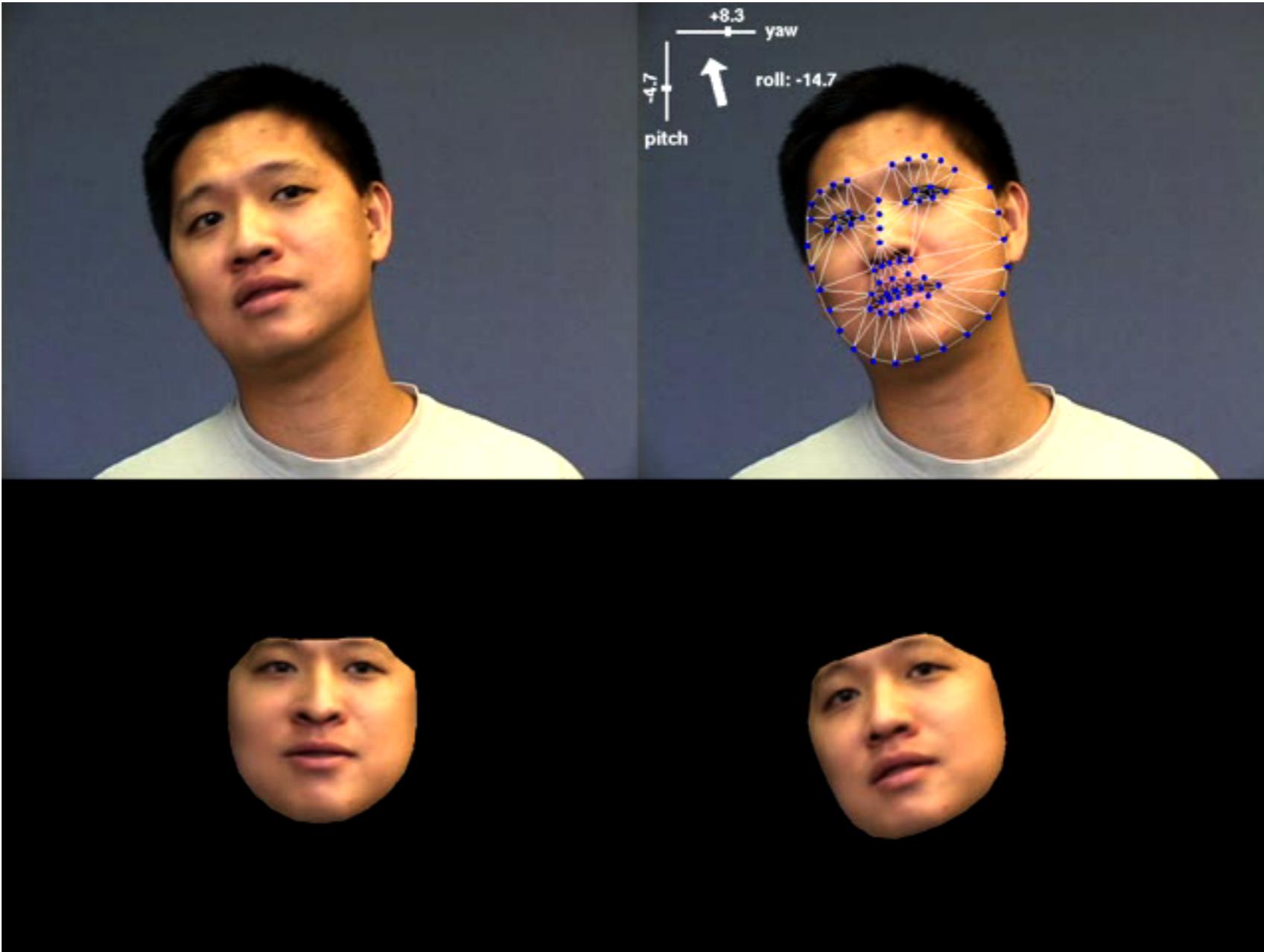
(Matthews and Gross)

# Face Recognition: Pose Normalization



(Matthews and Gross)

# Face Recognition: Pose Normalization



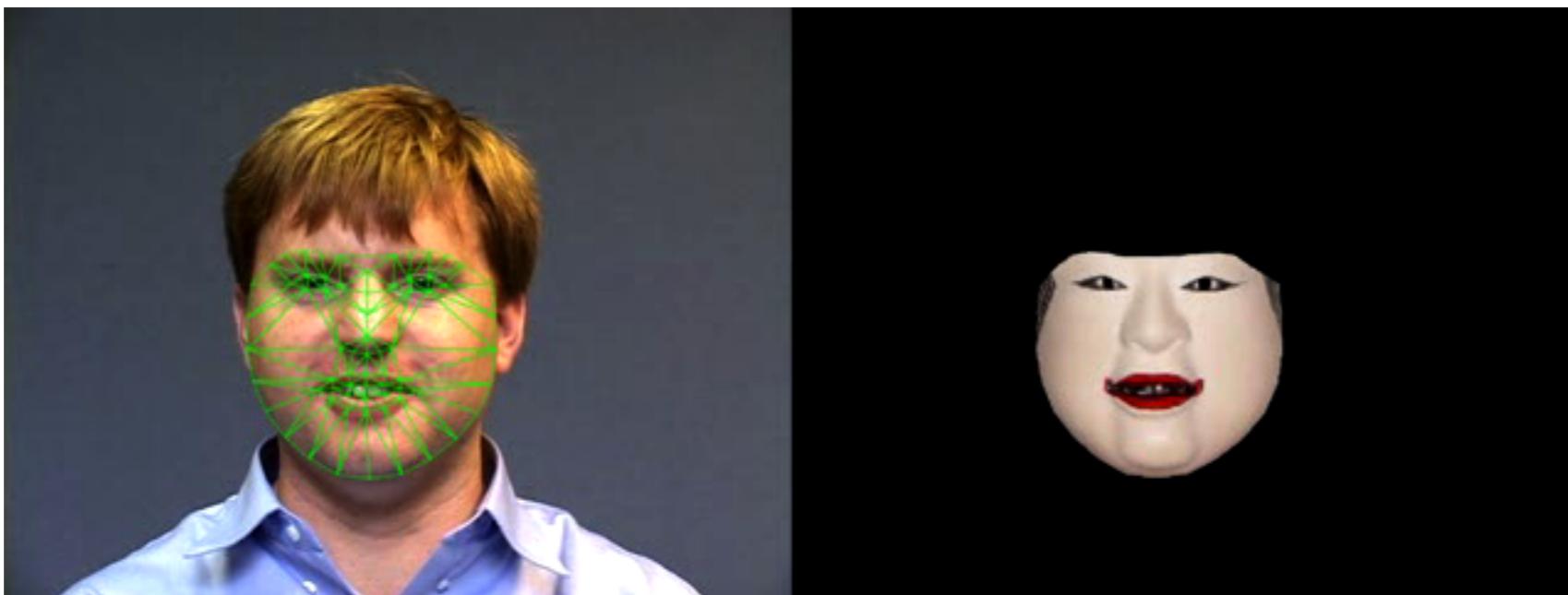
(Matthews and Gross)

# Animation Generation



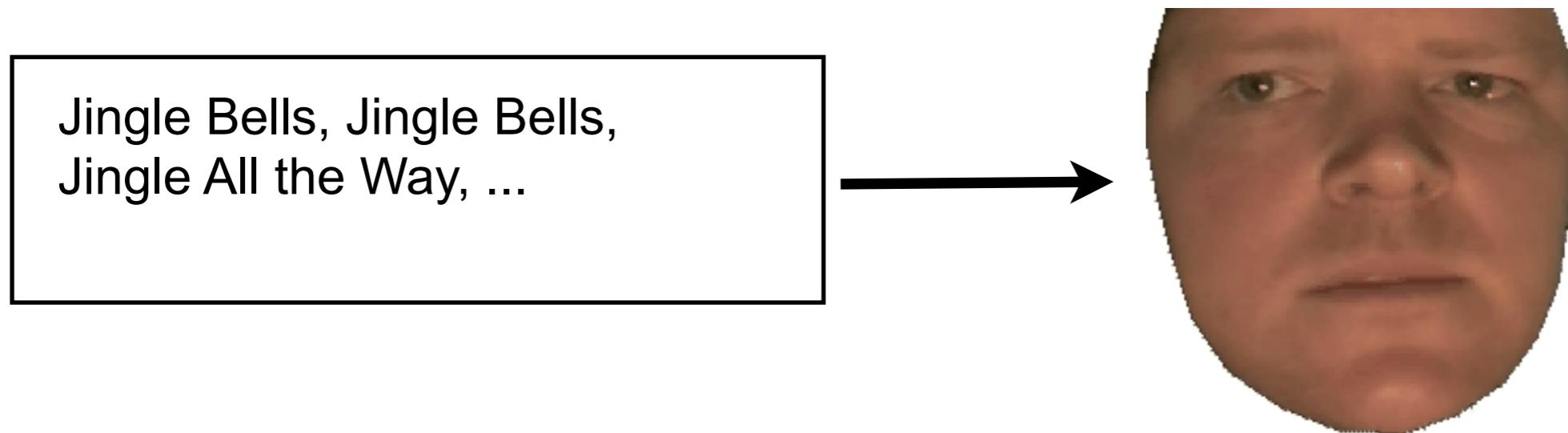
(Matthews and Gross)

# Animation Generation



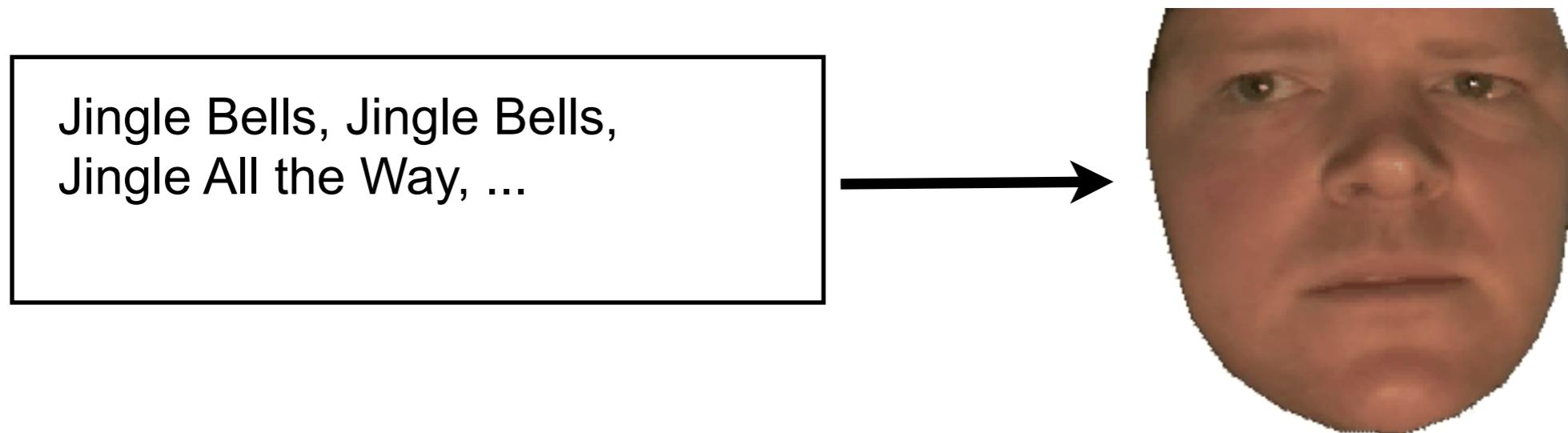
(Matthews and Gross)

# Audio-Visual Speech Synthesis



(Matthews and Gross)

# Audio-Visual Speech Synthesis



(Matthews and Gross)