Luxand, Inc.

http://www.luxand.com

Luxand Mirror Reality SDK 1.0

Developer's Guide

Table of Contents

Overview

Mirror Reality SDK is an iOS, Android and Windows library that allows one to put masks into faces in real time.

Library Activation

Mirror Reality SDK is a copy-protected library, and must be activated with a license key before its use. You need to pass the license key received from Luxand, Inc. to the MR_ActivateLibrary function.

To get a temporary evaluation key, fill in the following form:

http://luxand.com/mirror-reality/requestkey/ .

To get a permanent license key and ordering information, fill in the following form:

http://luxand.com/mirror-reality/order/ .

MR_ActivateLibrary Function

Activates the Mirror Reality SDK library.

iOS, Objective-C Syntax:

```
int MR_ActivateLibrary (char* LicenseKey);
```

Android, Java Syntax:

```
int MR.ActivateLibrary (char* LicenseKey);
```

Parameters:

LicenseKey – License key you received from Luxand, Inc.

Return Value:

Returns FSDKE_OK if successful.

Sample Applications

To create your app, use our sample applications.

iOS Sample

Android Sample

Windows Sample

They also use Luxand FaceSDK to work with images and detect faces and facial features, and OpenGL ES 1.x to draw.

Working with Masks

Every Mask consists of three files:

- filename.png – image file of a mask layer that is applied in multiply mode. You should use this layer to change skin color or add some translucent elements. When this layer is applied, the original skin texture is more visible compared to normal mode. This file may be missing.
- filename_normal.png – image file of a mask layer that is applied in normal mode. You should use this layer to draw something on the face or to make some parts lighter. When this layer is applied, the original skin texture is less visible compared to multiply mode. This file may be missing.
- filename.grd – text file of facial feature coordinates of the mask. This file is not in the Photoshop format. We recommend using our default .grd file. This file is not allowed to be missing.

Use our sample file in the Photoshop format when creating new masks:
http://luxand.com/download/mrsample.psd

It has two layers (one is applied in multiply mode; the second one is applied in normal mode). These layers must be square and their dimensions must be equal. Both layers have to be exported to separate .png files. The layer with the face must not be exported. This layer is for convenience only while creating a mask.

If the mask name is “filename,” layer in multiply mode is exported to “filename.png,” layer in normal mode to “filename_normal.png.”

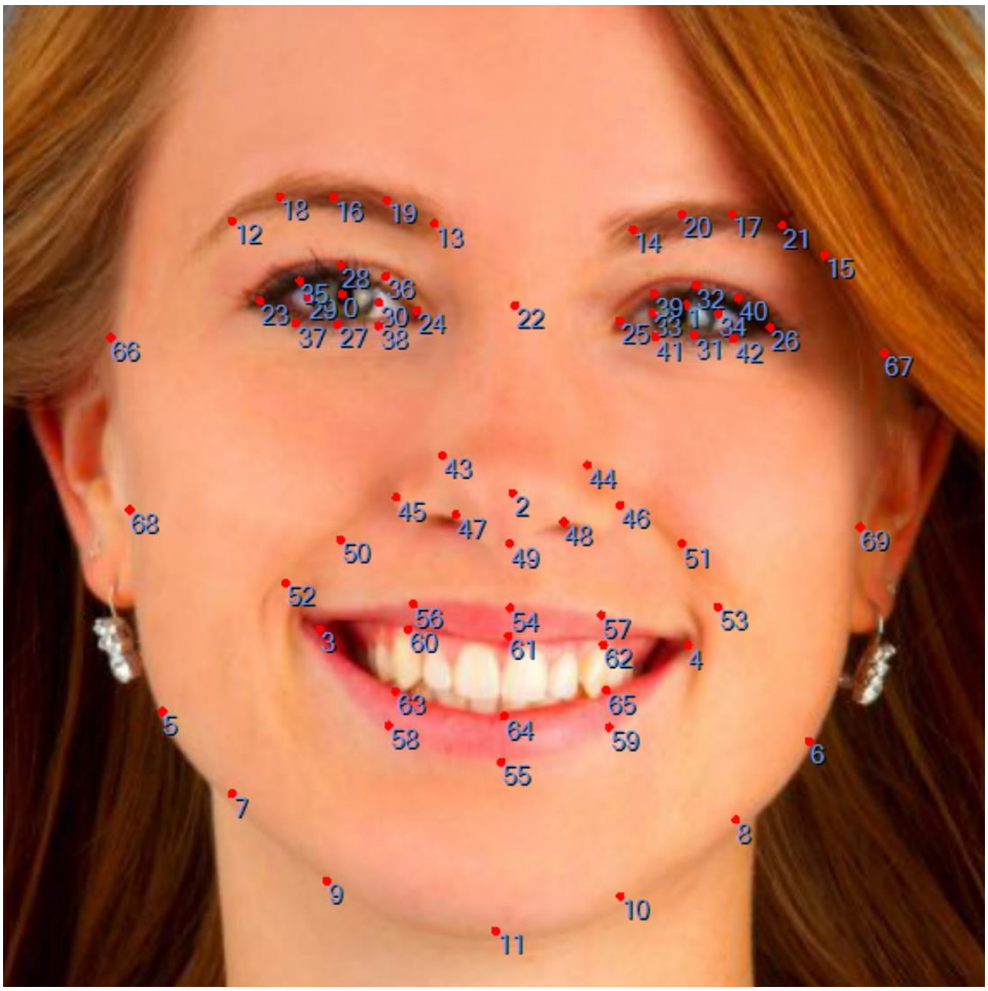
After that, you can use them with the SDK.

Facial Feature Coordinates of the Mask

Each mask must include a .grd file that contains its facial feature coordinates. Masks included in Mirror Reality SDK use the same .grd file. We recommend using this .grd file for your own masks as well. Therefore, if you create a new mask it must be aligned with the face in our mrsample.psd file. In this case, you do not need to edit the .grd file.

If you need to change some of your mask’s facial feature coordinates, you can edit the .grd file. Its structure is the following. The first line contains the number of facial features (points). Every next line contains the coordinates of the corresponding feature. If you need to change some coordinates, you can edit the corresponding line of the .grd file.

Mask facial feature coordinates should be matched with the corresponding facial feature coordinates (the default markup used in FaceSDK):



Please note that FaceSDK returns 70 facial features when detecting the facial features. Therefore, the custom .grd file must contain 70 facial features too.

Usage Scenario

1. Create an OpenGL textures array containing three elements: one for the original video frames and two for two mask files.
2. Load mask facial feature coordinates from the .grd file using the MR_LoadMaskCoordsFromFile or the MR_LoadMaskCoordsFromStream functions.
3. Load mask layers from .png files to the internal handles using the FSDK_LoadImageFromFileWithAlpha function. If one of the .png files is missing, then an empty internal handle for this file must be created using the FSDK_CreateEmptyImage function.
4. Load mask layers to the OpenGL textures using the MR_LoadMask function.
5. Detect faces and facial features on the camera frames using FaceSDK Tracker API functions (FSDK_FeedFrame and FSDK_GetTrackerFacialFeatures).
6. Add the mask to detected faces and display it on the screen using the MR_DrawGLScene function.

The FSDK_LoadImageFromFileWithAlpha, FSDK_CreateEmptyImage, FSDK_FeedFrame, and FSDK_GetTrackerFacialFeatures functions are Luxand FaceSDK functions. FaceSDK documentation is available here: luxand.com/facesdk/documentation.

Mirror Reality SDK Functions

The facial feature coordinates of the mask are stored in the MR_MaskFeatures data structure. MR_MaskFeatures is an array data type containing FSDK_FACIAL_FEATURE_COUNT points.

iOS, Objective-C Declaration:

```
typedef struct {
    float x, y;
} TPointf;

typedef TPointf MR_MaskFeatures [FSDK_FACIAL_FEATURE_COUNT];
```

Android, Java Declaration:

The class TPointf has the following properties:

```
  public float x, y;
```

The class MR_MaskFeatures has the following property:

```
  public TPointf features[];
```

C++ Declaration:

```
typedef struct {
    float x, y;
} TPointf;

typedef TPointf MR_MaskFeatures [FSDK_FACIAL_FEATURE_COUNT];
```

MR_LoadMaskCoordsFromFile Function

Loads facial feature coordinates of the mask from .grd file.

iOS, Objective-C Syntax:

```
int MR_LoadMaskCoordsFromFile (const char * filename,
MR_MaskFeatures maskCoords);
```

C++ Syntax:

```
int MR_LoadMaskCoordsFromFile (const char * filename,
MR_MaskFeatures maskCoords);
```

Parameters:

filename – pointer to the null-terminated string containing the name of a .grd file from which the mask coordinates will be loaded.

maskCoords – pointer to the array containing mask facial feature coordinates.

Return Value:

Returns FSDKE_OK if successful.

MR_LoadMaskCoordsFromStream Function

Loads facial feature coordinates of the mask from a stream.

Android, Java Syntax:

```
int MR.LoadMaskCoordsFromStream (InputStream stream,
MaskFeatures maskCoords);
```

Parameters:

stream – pointer to InputStream opened for .grd file's resource.

maskCoords – pointer to the array containing mask facial feature coordinates.

Return Value:

Returns FSDKE_OK if successful.

MR_LoadMask Function

Loads mask layers from files and loads them into OpenGL Textures.

iOS, Objective-C Syntax:

```
int MR_LoadMask (HImage maskImage1, HImage maskImage2, GLuint
maskTexture1, GLuint maskTexture2, int * isTexture1Created, int
* isTexture2Created);
```

Android, Java Syntax:

```
int MR.LoadMask (HImage maskImage1, HImage maskImage2, GLuint
maskTexture1, GLuint maskTexture2, int * isTexture1Created, int
* isTexture2Created);
```

C++ Syntax:

```
int MR_LoadMask (HImage maskImage1, HImage maskImage2, GLuint
maskTexture1, GLuint maskTexture2, int * isTexture1Created, int
* isTexture2Created);
```

Parameters:

maskImage1 – handle of the mask layer that is applied in multiply mode.

maskImage2 – handle of the mask layer that is applied in normal mode.

maskTexture1 – OpenGL texture of mask layer that is applied in multiply mode.

maskTexture2 – OpenGL texture of mask layer that is applied in normal mode.

isTexture1Created – flag indicating if the layer that is applied in multiply mode is copied to the OpenGL texture. The flag is equal to 0 if maskImage1 is empty, and is equal to 1 if maskImage1 is not empty.

isTexture2Created – flag indicating if the layer that is applied in normal mode is copied to the OpenGL texture. The flag is equal to 0 if maskImage2 is empty, and is equal to 1 if maskImage2 is not empty.

Return Value:

Returns FSDKE_OK if successful.

MR_DrawGLScene Function

Draws the faces with the masks on the screen.

iOS, Objective-C Syntax:

```
int MR_DrawGLScene (GLuint facesTexture, int facesDetected,
FSDK_Features features[MR_MAX_FACES], int
rotationAngle90Multiplier, int shiftType, GLuint maskTexture1,
GLuint maskTexture2, MR_MaskFeatures maskCoords, int
isTexture1Created, int isTexture2Created, int width, int
height);
```

Android, Java Syntax:

```
int MR.DrawGLScene (GLuint facesTexture, int facesDetected,
FSDK_Features features[MR_MAX_FACES], int
rotationAngle90Multiplier, int shiftType, GLuint maskTexture1,
GLuint maskTexture2, MR_MaskFeatures maskCoords, int
isTexture1Created, int isTexture2Created, int width, int
height);
```

C++ Syntax:

```
int MR_DrawGLScene (GLuint facesTexture, int facesDetected,
FSDK_Features features[MR_MAX_FACES], int
rotationAngle90Multiplier, int shiftType, GLuint maskTexture1,
GLuint maskTexture2, MR_MaskFeatures maskCoords, int
isTexture1Created, int isTexture2Created, int width, int
height);
```

Parameters:

facesTexture – OpenGL texture containing original frame from the camera.

facesDetected – number of detected faces that FSDK_FeedFrame returns.

features – arrays of facial features that FSDK_GetTrackerFacialFeatures returns.

rotationAngle90Multiplier – rotation angle (iOS: -1 for portrait, 1 for portrait upside down, 0 for landscape right, and 2 for landscape left).

shiftType – shift of facial feature coordinates (0 if coordinates are not shifted, 1 if the face should be shorter after transformation, 2 if the face should be longer after transformation).

maskTexture1 – OpenGL texture of mask layer that is applied in multiply mode.

maskTexture2 – OpenGL texture of mask layer that is applied in normal mode.

maskCoords – pointer to the array containing mask facial feature coordinates.

isTexture1Created – flag indicating if the layer that is applied in multiply mode is copied to the OpenGL texture.

isTexture2Created – flag indicating if the layer that is applied in normal mode is copied to the OpenGL texture.

width – width of the output frame.

height – height of the output frame.

Return Value:

Returns FSDKE_OK if successful.

Library Information

The Mirror Reality SDK library uses libjpeg-turbo © Miyasaka Masaru, TigerVNC and VirtualGL projects, libcurl © Daniel Stenberg; libpng © Glenn Randers-Pehrson; easybmp © The EasyBMP Project (http://easybmp.sourceforge.net); RSA Data Security, Inc. MD5 Message-Digest Algorithm.

