MegEye-L9K Linux SDK API User Guide V1.0.9

This document is the Linux SDK integration manual for the Shituo Technology L9K module, targeting end users. It provides detailed descriptions of the SDK's classes, member variables, member functions, and partial usage guidelines. For actual integration, please use it in conjunction with the provided Demo project for complete functionality.

Document Table of Contents

MegEye-L9K Linux SDK API User Guide V1.0.9

- Document Table of Contents
- CHANGELOG
- Reference
  - facePassInit
    - Overview
    - Parameter Description
    - Usage Example
  - facePassDeInit
    - Overview
    - Parameter Description
    - Usage Example
  - facePassGetVersion
    - Overview
    - Parameter Description
    - Usage Example
  - facePassStartRecog
    - Overview
    - Parameter Description
    - Usage Example
  - facePassPauseRecog
    - Overview
    - Parameter Description
    - Usage Example
  - facePassStartRegister
    - Overview
    - Parameter Description
    - Usage Example
  - facePassCreateGroup
    - Overview
    - Parameter Description
    - Usage Example
  - facePassInsertFeature
    - Overview
    - Parameter Description
    - Usage Example
  - facePassBatchInsertFeatures
    - Overview

- Parameter Description
- Usage Example

facePassSetLivessConfig
- Overview
- Parameter Description
- Usage Example

facePassSetRecogConfig
- Overview
- Parameter Description
- Usage Example

facePassRemoveUserFromGroupByfaceID
- Overview
- Parameter Description
- Usage Example

facePassRemoveUsersFromGroupByGroupId
- Overview
- Parameter Description
- Usage Example

facePassRemoveAllUsers
- Overview
- Parameter Description
- Usage Example

facePassGetUserCnt
- Overview
- Parameter Description
- Usage Example

facePassSetSnapShotPush
- Overview
- Parameter Description
- Usage Example

facePassSetEcoMode
- Overview
- Parameter Description
- Usage Example

facePassAddUserByJpeg
- Overview
- Parameter Description
- Usage Example

facePassOtaInit
- Overview
- Parameter Description
- Usage Example

facePassOtaStart
- Overview
- Parameter Description
- Usage Example

facePassOtaSendPackets
- Overview
- Parameter Description
- Usage Example

facePassSetSystemTime
- Overview
- Parameter Description

- Usage Example

- facePassVideoSetUvcResolution
  - Overview
  - Parameter Description
  - Usage Example
- facePassVideoSetUvcCodeType
  - Overview
  - Parameter Description
  - Usage Example
- facePassVideoSetUvcVideoSt
  - Overview
  - Parameter Description
  - Usage Example
- facePassVideoSetUvcVideoHmirror
  - Overview
  - Parameter Description
  - Usage Example
- facePassEnableQRCodeDetect
  - Overview
  - Parameter Description
  - Usage Example
- facePassEnableFaceRepeat
  - Overview
  - Parameter Description
  - Usage Example
- facePassSetOnlyDetectMode
  - Overview
  - Parameter Description
  - Usage Example
- facePassSetMaxDistance
  - Overview
  - Parameter Description
  - Usage Example
- facePassSetNoTargetDetPushTime
  - Overview
  - Parameter Description
  - Usage Example
- facePassCompare1v1
  - Overview
  - Parameter Description
  - Usage Example
- facePassPicCompare1v1
  - Overview
  - Parameter Description
  - Usage Example
- facePassCompare1vn
  - Overview
  - Parameter Description
  - Usage Example
- UVC_initVideoCapture
  - Overview
  - Parameter Description
  - Usage Example

- UVC_releaseVideoCapture
  - Overview
  - Parameter Description
  - Usage Example
- UVC_startCaptureFrames
  - Overview
  - Parameter Description
  - Usage Example
- UVC_stopCaptureFrames
  - Overview
  - Parameter Description
  - Usage Example

Data structures

- FACE_PASS_CALLBACK
  - Overview
- FACE_PASS_GROUP_E
  - Overview
- FACE_PASS_UVC_RES_E
  - Overview
- FACE_PASS_UVC_PLAYLOAD_TYPE_E
  - Overview
- FACE_PASS_ATTR_INFO_S
  - Overview
- FACE_PASS_DET_INFO_S
  - Overview
- FACE_PASS_IDEN_INFO_S
  - Overview
- FACE_PASS_REG_INFO_S
  - Overview
- FACE_PASS_SYSTIME
  - Overview
- FACE_PASS_OTA_INFO_S
  - Overview
- FACE_PASS_QRCODE_INFO_S
  - Overview
- FACE_PASS_DATA_INFO_S
  - Overview
- FACE_PASS_PPLCODE_E
  - Overview
- FACE_PASS_FACE_ERR_E
  - Overview
- FrameCallback
- Example

CHANGELOG

- **V1.0.9**

    1. Added UVC video stream horizontal mirror function

- **V1.0.8**

    1. Added feature 1VN recognition function
    2. Added recognition distance setting function
    3. Added no-target detection push time setting function
    4. Added ID photo feature extraction without database storage function

- **V1.0.7**

    1. Adjusted deinitialization interface, added restart state control
    2. Added image 1V1 comparison function

- **V1.0.6**

    1. Removed adaptive enrollment function
    2. Added database reset function

- **V1.0.5**

    1. Added log recording function for upper-level machines
    2. Added target detection status notification
    3. Optimized test item input

- **V1.0.4**

    1. Corrected partial structure definitions
    2. Added UVC streaming decoding function

- **V1.0.3**

    1. Added QR code recognition function
    2. Added UVC frame rate setting function
    3. Added UVC stream encoding type setting function
    4. Completed error type definition

- **V1.0.2**

    1. Added UVC frame rate setting function
    2. Redefined algorithm data reporting type
    3. Added system time synchronization function
    4. Added error type definition

- **v1.0.1**
    1. Added power-saving mode
    2. Added snapshot push switch
    3. Added OTA upgrade function
    4. Separated recognition threshold ranges for face and palm
    5. Completed user deletion function
    6. Added database quantity acquisition function
- **v1.0.0**
    1. Initial version, completed basic enrollment and recognition function development
    2. Completed initial version of main device communication interaction and message definition
    3. Supported image enrollment, single-device enrollment, and feature-only enrollment

Reference

facePassInit

```
int32_t facePassInit(FACE_PASS_CALLBACK *face_pass_cb)
```

Overview

Module initialization

Parameter Description

- face_pass_cb: Message receiving callback function set, refer to **FACE_PASS_CALLBACK** for specific definition
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
FACE_PASS_CALLBACK face_pass_cb = {0};
face_pass_cb.serialMsgRead = read_msg;
face_pass_cb.serialMsgWrite = send_msg;
face_pass_cb.recog_process = recog_result_process;
ret = facePassInit(&face_pass_cb);
```

facePassDeInit

```
int32_t facePassDeInit(int reboot)
```

Overview

Module deinitialization. Clears face and palm features from memory and reboots the module.

Parameter Description

- reboot: 0 - Module does not reboot; 1 - Module reboots
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
ret = facePassDeInit(0);
```

facePassGetVersion

```
void facePassGetVersion(char *soft_ver,char *sdk_ver,char *sn,int32_t *feature_id)
```

Overview

Get module software version

Parameter Description

- soft_ver: Software version
- sdk_ver: Algorithm version
- sn: Device serial number
- feature_id: Feature ID

Usage Example

```
char *soft_ver[128];
char *sdk_ver[128];
char *sn[128];
int32_t feature_id;
facePassGetVersion(soft_ver,sdk_ver,sn,&feature_id);
```

facePassStartRecog

```
int32_t facePassStartRecog(void)
```

Overview

Start recognition

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret;
ret = facePassStartRecog();
```

facePassPauseRecog

```
int32_t facePassPauseRecog(void)
```

Overview

Pause algorithm recognition

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret;
ret = facePassPauseRecog();
```

facePassStartRegister

```
int32_t facePassStartRegister(int32_t face_id,int32_t type,int32_t time_out)
```

Overview

Start user enrollment according to the specified target type

Parameter Description

- face_id: Unique identifier for the user being enrolled
- type: Enrollment target type. 0 - Face enrollment; 1 - Palm enrollment
- time_out: Enrollment timeout period in ms
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret;
ret = facePassStartRegister(1, 0, 15000);
```

facePassCreateGroup

```
int32_t facePassCreateGroup(int32_t *group_id,int32_t group_type)
```

Overview

Create different feature management group IDs according to the target type

Parameter Description

- group_id: Group ID returned by the module
- group_type: Group type, refer to **FACE_PASS_GROUP_E** for details
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t face_group_id = -1;
int32_t ret = 0;
ret = facePassCreateGroup(&face_group_id, FACE_PASS_GROUP_FACE);
```

facePassInsertFeature

```
int32_t facePassInsertFeature(int32_t group_id,unsigned int face_id,const char *ft,int32_t
ft_len)
```

Overview

Create a model, insert existing user features into the module, build a database for search

Parameter Description

- group_id: Group ID. Face and palm features are managed using different group IDs
- face_id: Unique user identifier
- ft: Feature data
- ft_len: Feature length
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t face_group_id = -1;
int32_t ret = 0;
int32_t ft_len = 1048;
char face_ft[1048] = {0};
ret = facePassCreateGroup(&face_group_id, FACE_PASS_GROUP_FACE);
if(ret < 0)
{
    perror(ret);
}
ret = facePassInsertFeature(face_group_id, 1, ft, ft_len);
```

facePassBatchInsertFeatures

```
int32_t facePassBatchInsertFeatures(int32_t user_cnts,FACE_PASS_FT_INFO_S *fts)
```

Overview

Create a model, batch insert existing user features into the module, build a database for search. It is recommended to batch 600 face data entries or 300 palm data entries each time.

Parameter Description

- user_cnts: Number of users being inserted currently
- fts: Batch feature collection
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t face_group_id = -1;
int32_t ret = 0;
int32_t ft_len = 1048;
char face_ft[1048] = {0};
ret = facePassCreateGroup(&face_group_id, FACE_PASS_GROUP_FACE);
if(ret < 0)
{
    perror(ret);
}
FACE_PASS_FT_INFO_S fts[1];
ret = facePassBatchInsertFeatures(1, fts);
```

facePassSetLivessConfig

```
int32_t facePassSetLivessConfig(int32_t en_live,int32_t live_th,int32_t retry_cnts)
```

Overview

Set liveness function configuration

Parameter Description

- en_live: 0 - Disable liveness function; 1 - Enable liveness function
- live_th: Liveness threshold, range 0-100
- retry_cnts: Liveness retest count, range 0-5
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassSetLivessConfig(0, 80, 3);
```

facePassSetRecogConfig

```
int32_t facePassSetRecogConfig(int32_t face_th,int32_t palm_th,int32_t retry_cnts,int32_t
push_interval)
```

Overview

Set recognition parameter configuration

Parameter Description

- face_th: Face recognition threshold, range 0-100, recommended 90
- palm_th: Palm recognition threshold, recommended 43
- retry_cnts: Stranger retest count
- push_interval: Interval between image pushes in seconds
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassSetRecogConfig(90, 43, 3, 2);
```

facePassRemoveUserFromGroupByfaceID

```
int32_t facePassRemoveUserFromGroupByfaceID(int32_t group_id, uint64_t face_id)
```

Overview

Delete specified user information from the database

Parameter Description

- group_id: User group ID
- face_id: Unique user identifier
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassRemoveUserFromGroupByfaceID(FACE_PASS_GROUP_FACE, 1);
```

facePassRemoveUsersFromGroupByGroupId

```
int32_t facePassRemoveUsersFromGroupByGroupId(int32_t group_id)
```

Overview

Delete all objects of the specified target type from the database

Parameter Description

- group_id: User group ID
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassRemoveUsersFromGroupByGroupId(FACE_PASS_GROUP_FACE);
```

facePassRemoveAllUsers

```
int32_t facePassRemoveAllUsers(void)
```

Overview

Delete all targets from the database

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassRemoveAllUsers();
```

facePassGetUserCnt

```
int32_t facePassGetUserCnt(int32_t obj_type,int32_t *obj_cnt)
```

Overview

Get the number of specified target types in the module database

Parameter Description

- obj_type: Target type, refer to **FACE_PASS_OBJ_TYPE_E**
- obj_cnt: Returns the number of corresponding target types
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t face_cnt = 0;
ret = facePassGetUserCnt(FACE_PASS_OBJ_TYPE_FACE, &face_cnt);
```

facePassSetSnapShotPush

```
int32_t facePassSetSnapShotPush(int32_t push_st,int32_t crop_st)
```

Overview

Set whether to push snapshots during enrollment and recognition. (If database re-extraction is required, the snapshots enrolled by the device must be saved. If the local storage space is limited, the cropping function can be enabled. It is not recommended to crop palm images.)

Parameter Description

- push_st: 0 - Do not push snapshots; 1 - Push snapshots
- crop_st: 0 - Do not crop snapshots; 1 - Crop snapshots
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassSetSnapShotPush(0, 0);
```

facePassSetEcoMode

```
int32_t facePassSetEcoMode(int32_t eco_st)
```

Overview

Enable/disable power-saving mode. (If no target is detected for a long time, the infrared light will be turned off to reduce power consumption.)

Parameter Description

- eco_st: 0 - Disable power-saving mode; 1 - Enable power-saving mode
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassSetEcoMode(0);
```

facePassAddUserByJpeg

```
int32_t facePassAddUserByJpeg(int32_t face_id,const char *pic,int32_t pic_len,char
*ft,int32_t *ft_len,int32_t *otype,int32_t *fid,int32_t *err_info)
```

Overview

Face image enrollment

Parameter Description

- face_id: Unique user identifier
- pic: Image content
- pic_len: Image length
- ft: Returned feature data
- ft_len: Feature length
- otype: Returned target type
- fid: face_id in the database, must match the input face_id
- err_info: Error information, refer to **FACE_PASS_FACE_ERR_E** definition
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassAddUserByJpeg(1, pic, pic_len, ft, &ft_len, &otype, &fid, &err_info);
```

facePassOtaInit

```
int32_t facePassOtaInit(void)
```

Overview

Notify the module to enter OTA initialization

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassOtaInit();
```

facePassOtaStart

```
int32_t facePassOtaStart(void)
```

Overview

After the upgrade firmware is transmitted, notify the module to perform flash burning

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassOtaStart(0);
```

facePassOtaSendPackets

```
int32_t facePassOtaSendPackets(char *data,int32_data_size)
```

Overview

Send upgrade firmware to the module. It is recommended to send in 128K packets.

Parameter Description

- data: Upgrade firmware data
- data_size: Upgrade firmware data size
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassOtaInit();
if(ret > 0)
{
    ret = facePassOtaSendPackets(data, data_size);
    if(ret > 0)
    {
        ret = facePassOtaStart();
    }
}
```

facePassSetSystemTime

```
int32_t facePassSetSystemTime(FACE_PASS_SYSTIME system_time)
```

Overview

Set module system time

Parameter Description

- FACE_PASS_SYSTIME: System time definition type, refer to FACE_PASS_SYSTIME
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
FACE_PASS_SYSTIME time = {0};
time.year = system_time.year;
time.month = system_time.month;
time.day = system_time.day;
time.hour = system_time.hour;
time.min = system_time.min;
time.sec = system_time.sec;
ret = facePassSetSystemTime(&time);
```

facePassVideoSetUvcResolution

```
int32_t facePassVideoSetUvcResolution(FACE_PASS_UVC_RES_E resolution)
```

Overview

Set UVC output resolution

Parameter Description

- FACE_PASS_UVC_RES_E: Resolution type, refer to FACE_PASS_UVC_RES_E
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassVideoSetUvcResolution(FACE_PASS_UVC_RES_720P);
```

facePassVideoSetUvcCodeType

```
int32_t facePassVideoSetUvcCodeType(FACE_PASS_UVC_PLAYLOAD_TYPE_E code_type)
```

Overview

Set UVC video stream encoding type. Currently only supports three formats: mjpeg, H264, and static mjpeg images.

Parameter Description

- FACE_PASS_UVC_PLAYLOAD_TYPE_E: Resolution type, refer to **FACE_PASS_UVC_PLAYLOAD_TYPE_E**
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassVideoSetUvcCodeType(FACE_PASS_UVC_PAYLOAD_TYPE_MJPEG);
```

facePassVideoSetUvcVideoSt

```
int32_t facePassVideoSetUvcVideoSt(bool en_video)
```

Overview

Set UVC video stream switch. 0 - Disable video transmission; 1 - Enable video transmission. (For low-performance platform integration, video feedback is not required, and the effect will take effect after restart.)

Parameter Description

- en_video: false - Disable video stream; true - Enable video stream
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassVideoSetUvcVideoSt(false);
```

facePassVideoSetUvcVideoHmirror

```
int32_t facePassVideoSetUvcVideoHmirror(bool en_mirror)
```

Overview

Set UVC video stream horizontal mirror switch. 0 - Disable horizontal mirror; 1 - Enable horizontal mirror. (Set according to some user requirements, the effect will take effect after restart.)

Parameter Description

- en_mirror: false - Disable horizontal mirror; true - Enable horizontal mirror
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassVideoSetUvcVideoHmirror(false);
```

facePassEnableQRCodeDetect

```
int32_t facePassEnableQRCodeDetect(bool en_qrcode)
```

Overview

Enable/disable QR code detection function. Conflicts with face and palm recognition algorithms. After enabling QR code detection, face and palm recognition will be automatically paused. There is a default timeout internally. If no QR code is detected within a certain time, QR code detection will be automatically paused.

Parameter Description

- en_qrcode: false - Disable QR code detection; true - Start QR code detection
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassEnableQRCodeDetect(false);
```

facePassEnableFaceRepeat

```
int32_t facePassEnableFaceRepeat(bool en_repeat)
```

Overview

Enable/disable database deduplication function

Parameter Description

- en_repeat: false - Disable database deduplication; true - Enable database deduplication
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
ret = facePassEnableFaceRepeat(false);
```

facePassSetOnlyDetectMode

```
int32_t facePassSetOnlyDetectMode(int32_t det_st)
```

Overview

Enable/disable detection-only mode. After enabling, the module only pushes detection data, not recognition and liveness data. (Function customized by some users.)

Parameter Description

- det_st: 1 - Enable detection-only function; 0 - Disable
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t det_st = 1;
ret = facePassSetOnlyDetectMode(det_st);
```

facePassSetMaxDistance

```
int32_t facePassSetMaxDistance(int32_t distance);
```

Overview

Set maximum recognition distance in cm. The maximum configurable distance is 2m.

Parameter Description

- distance: Recognition distance in cm, recommended to set between 60-200
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t distance = 120;
ret = facePassSetMaxDistance(distance);
```

facePassSetNoTargetDetPushTime

```
int32_t facePassSetNoTargetDetPushTime(int32_t time_ms);
```

Overview

Set push time when no target is detected. To avoid frequent reports, the minimum setting time is 200ms. (Customized by some users.)

Parameter Description

- time_ms: Push time in milliseconds
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t time_ms = 1000;
ret = facePassSetNoTargetDetPushTime(time_ms);
```

facePassCompare1v1

```
int32_t facePassCompare1v1(const char *fta, int16_t fta_len, char *ftb,int16_t
ftb_len,int32_t ft_type,int32_t *score)
```

Overview

Feature 1v1 comparison function (customized by some users)

Parameter Description

- fta: Original feature A data
- fta_len: Original feature A data length
- ftb: Original feature B data
- ftb_len: Original feature B data length
- ft_type: Feature type. 0 - Face; 1 - Palm
- score: Comparison score, returned
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t score = 0;
int32_t ft_type = 0;
int32_t fta_len = ftb_len = 280;
char *fta[1024];
char *ftb[1024];
ret = facePassCompare1v1(fta, fta_len, ftb, ftb_len, ft_type, &score);
```

facePassPicCompare1v1

```
FACE_API int32_t facePassPicCompare1v1(const char *pic_a, int32_t pica_len, char
*pic_b,int32_t picb_len,int32_t *score,int32_t *err_info)
```

Overview

Image 1v1 comparison function (customized by some users)

Parameter Description

- pic_a: Image A data
- pica_len: Image A data length
- pic_b: Image B data
- picb_len: Image B data length
- score: Comparison score, returned
- err_info: Error information, returned
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t score = 0;
int32_t err_info = 0;
ret = facePassPicCompare1v1(pic_a, pica_len, pic_b, picb_len, &score, &err_info);
```

facePassCompare1vn

```
int32_t facePassCompare1vn(const char *ft, int16_t ft_len,int32_t ft_type,int32_t
*top_id,int32_t *score)
```

Overview

Feature 1vn comparison function (customized by some users)

Parameter Description

- ft: Original feature data
- ft_len: Original feature data length
- ft_type: Feature type. 0 - Face; 1 - Palm
- top_id: User ID with the highest similarity
- score: Comparison score, returned
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
int32_t ret = 0;
int32_t score = 0;
int32_t top_id = 0;
int32_t ft_type = 0;
int32_t ft_len = 280;
char *ft[1024];
ret = facePassCompare1vn(ft, ft_len, ft_type, &top_id, &score);
```

UVC_initVideoCapture

```
int UVC_initVideoCapture(const char *deviceName, int width, int height, int fps,
FrameCallback callback, void *userData)
```

Overview

Initialize UVC streaming function

Parameter Description

- deviceName: Device node name. Linux: /dev/video0; Windows: MJPG Camera
- width: Video stream width
- height: Video stream height
- fps: Frame rate
- FrameCallback: Video frame callback
- userData: User data
- Return value: Returns 0 on success, -1 on failure

Usage Example

```
void get_uvc_stream1(const uint8_t *data, int width, int height, int stride, int *format,
void *userData);
if(UVC_initVideoCapture("MJPG Camera", 720, 1280, 15, get_uvc_stream1, NULL) < 0)
{
    PRINTF(LOG_ERROR, "UVC_initVideoCapture failed\n");
}
else
{
    UVC_startCaptureFrames();
}
/*exit*/
UVC_releaseVideoCapture();
```

UVC_releaseVideoCapture

```
void UVC_releaseVideoCapture(void)
```

Overview

Release video capture resources

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

Refer to **UVC_initVideoCapture Usage Example**

UVC_startCaptureFrames

```
void UVC_startCaptureFrames(void)
```

Overview

Start video frame capture

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

Refer to **UVC_initVideoCapture Usage Example**

UVC_stopCaptureFrames

```
void UVC_stopCaptureFrames(void)
```

Overview

Stop video frame capture

Parameter Description

- Return value: Returns 0 on success, -1 on failure

Usage Example

Refer to **UVC_initVideoCapture Usage Example**

Data structures

FACE_PASS_CALLBACK

Overview

Callback related

```
typedef int32_t (*SERIAL_MSG_READ)(unsigned char *data, unsigned int len);
// Receive message from UART
typedef int32_t (*SERIAL_MSG_WRITE)(unsigned char *data, unsigned int len);
// Send data via UART
typedef int32_t (*REPORT_DATA_INFO)(FACE_PASS_DATA_INFO_S* module_data);
// Upper layer main program needs to process recognition results all in this callback
typedef struct
{
    SERIAL_MSG_READ serialMsgRead; // ACM interface data reception
    SERIAL_MSG_WRITE serialMsgWrite; // ACM interface data transmission
    REPORT_DATA_INFO reportDataInfo; // Module upload data information
} FACE_PASS_CALLBACK;
```

FACE_PASS_GROUP_E

Overview

Target group type

```
typedef enum
{
    FACE_PASS_GROUP_FACE = 0, // Face group
    FACE_PASS_GROUP_PALM,     // Palm group
    FACE_PASS_GROUP_MAX = 10
} FACE_PASS_GROUP_E;
```

FACE_PASS_UVC_RES_E

Overview

Resolution type

```
typedef enum
{
    FACE_PASS_UVC_RES_1080P = 0,
    FACE_PASS_UVC_RES_720P,
    FACE_PASS_UVC_RES_480P,
    FACE_PASS_UVC_RES_320P,
    FACE_PASS_UVC_RES_480_800,
    FACE_PASS_UVC_RES_600_1024,
    FACE_PASS_UVC_RES_640_480
} FACE_PASS_UVC_RES_E;
```

FACE_PASS_UVC_PLAYLOAD_TYPE_E

Overview

UVC encoding type. Currently only supports three formats: H264, MJPEG, and static images.

```
typedef enum
{
    FACE_PASS_UVC_PAYLOAD_TYPE_H264 = 0,
    FACE_PASS_UVC_PAYLOAD_TYPE_JPEG,
    FACE_PASS_UVC_PAYLOAD_TYPE_MJPEG,
    FACE_PASS_UVC_PAYLOAD_TYPE_H265,
    FACE_PASS_UVC_PAYLOAD_TYPE_STATIC_JPEG,
    FACE_PASS_UVC_PAYLOAD_TYPE_YUYV,
    FACE_PASS_UVC_PAYLOAD_TYPE_NV21,
    FACE_PASS_UVC_PAYLOAD_TYPE_BUTT
} FACE_PASS_UVC_PLAYLOAD_TYPE_E;
```

FACE_PASS_ATTR_INFO_S

Overview

Pose and sharpness information

```
typedef struct
{
    int32_t blur;
    int32_t roll;
    int32_t pitch;
    int32_t yaw;
} FACE_PASS_ATTR_INFO_S;
```

FACE_PASS_DET_INFO_S

Overview

Detection related information

```
typedef struct
{
    int32_t rect_x; // Target frame x-axis coordinate
    int32_t rect_y; // Target frame y-axis coordinate
    int32_t rect_w; // Target frame width
    int32_t rect_h; // Target frame height
    int32_t det_st; // For internal debugging, users can ignore
    int32_t fail_index; // For internal debugging, users can ignore
} FACE_PASS_DET_INFO_S;
```

FACE_PASS_IDEN_INFO_S

Overview

```
typedef struct
{
    int8_t iden_ges; // Holding state
    int32_t top1_id; // Top1 ID
    int8_t iden_score; // Recognition score 0-100
    int8_t stranger; // Stranger identifier
} FACE_PASS_IDEN_INFO_S;
```

FACE_PASS_REG_INFO_S

Overview

```
typedef struct
{
    int32_t obj_type;   // Target type. 0 - Face; 1 - Palm
    int32_t face_id;    // Unique user identifier
    int8_t id_existed;  // Whether duplicate enrollment
    int16_t ft_len;
    char ft[1048];
} FACE_PASS_REG_INFO_S;
```

FACE_PASS_SYSTIME

Overview

System time

```
typedef struct {
unsigned int sec : 6;   //Second: 0~63
unsigned int min : 6;   // Minute: 0~63
unsigned int hour : 5;  //Hour: 0~23
unsigned int day : 5;   //Day: 1~31
unsigned int month : 4; //Month: 1~12
unsigned int year;      //Year: 0~63 (years since SYSTIME_FORM_YEAR)
} FACE_PASS_SYSTIME;
```

FACE_PASS_OTA_INFO_S

Overview

OTA upgrade related information

```
typedef struct
{
    int32_t process; // Upgrade progress
    int8_t err_info; // Error information
} FACE_PASS_OTA_INFO_S;
```

FACE_PASS_QRCODE_INFO_S

Overview

QR code detection information

```
typedef struct
{
    char code_str[512]; // QR code content
    int32_t str_len; // QR code length
} FACE_PASS_QRCODE_INFO_S;
```

FACE_PASS_DATA_INFO_S

Overview

Algorithm frame processing results, upgrade information, and QR code detection information

```
typedef struct
{
    int32_t ret;
    int32_t ppl_code; // Algorithm PPL return value FACE_PASS_PPLCODE
    int8_t obj_type;  // Face/palm
    int32_t fid;      // Current frame ID
    int16_t ft_len;   // Feature length
    char ft[1048];    // Feature data
    int32_t rgb_size; // Visible light image size
    char rgb_buff;    // Visible light image data
    int32_t ir_size;  // Infrared image size
    char ir_buff;     // Infrared image data
    FACE_PASS_DET_INFO_S det_info;   // Detection related information
    FACE_PASS_ATTR_INFO_S attr_info; // Attribute related information
    FACE_PASS_IDEN_INFO_S iden_info; // Recognition related information
    FACE_PASS_REG_INFO_S reg_info;   // Enrollment related information
    FACE_PASS_OTA_INFO_S ota_info;
    FACE_PASS_QRCODE_INFO_S qrcode_info;
} FACE_PASS_DATA_INFO_S;
```

FACE_PASS_PPLCODE_E

Overview

Algorithm PPL processing state return information

```
typedef enum FACE_PASS_PPLCODE_EU {
    OBJECT_STATE_NOOBJECT, // No target detected: face or palm
    OBJECT_STATE_TOOUP,    // Target too close to the top edge of the image
    OBJECT_STATE_TOODOWN,  // Target too close to the bottom edge of the image
    OBJECT_STATE_TOOLEFT,  // Target too close to the left edge of the image
    OBJECT_STATE_TOORIGHT, // Target too close to the right edge of the image
    OBJECT_STATE_FAR,      // Target too far
    OBJECT_STATE_CLOSE,    // Target too close
    OBJECT_STATE_BLUR_UNQUALIFIED,   // Target clarity unqualified
    OBJECT_STATE_POSE_UNQUALIFIED,   // Target pose unqualified
    OBJECT_STATE_PAIR_NOOBJECT,      // No target detected in paired frame
    OBJECT_STATE_EYE_OCCLUSION = 30, // Eye occlusion
    OBJECT_STATE_FACE_OCCLUSION,     // Face occlusion
    OBJECT_STATE_MULTI_OCCLUSION,    // Multiple targets detected in the image, all do not
meet the occ requirements
    OBJECT_STATE_LUMA_OVEREXPOSED,   // Image overexposed
    OBJECT_STATE_LUMA_OVERDARK,      // Image underexposed
    OBJECT_STATE_LUMA_MULTI_ERR,     // Multiple targets in the image, all overexposed or
underexposed
    OBJECT_STATE_EXTRACT_FT_ERR,            // Feature extraction failed
    OBJECT_STATE_FACES_NOT_MATCH = 50,      // IR and RGB target matching failed
    OBJECT_STATE_LIVENESS_SCORE_CHECK_FAIL, // Liveness verification failed
    OBJECT_STATE_CLARITY_SCORE_CHECK_FAIL,  // Palm clarity verification failed
    OBJECT_STATE_GESTURE_SCORE_CHECK_FAIL,  // Holding gesture verification failed
(reserved)
    OBJECT_STATE_W_H_RATIO_CHECK_FAIL,   // Palm width-height ratio detection abnormal
    OBJECT_STATE_ENROLL_ERROR,           // Local enrollment abnormal
    OBJECT_STATE_ENROLL_ID_EXIST,        // This ID already exists in the database during
enrollment
    OBJECT_STATE_ENROLL_TIME_OUT,        // Enrollment timeout
    OBJECT_STATE_UNLOCK_ERROR = 70,      // Comparison error
    OBJECT_STATE_UNLOCK_COMPARE_FAIL,    // 1:N comparison failed
    OBJECT_STATE_UNLOCK_STRANGER,        // Comparison determined as stranger
    OBJECT_STATE_INVALID_PARAMETER = 90, // Invalid parameter
    OBJECT_STATE_INTERNAL_ERROR,         // SDK internal error
    OBJECT_STATE_AUTHRIZE_FAILED,        // Authorization failed
    OBJECT_STATE_GET_IMG_ERROR,          // Main frame acquisition abnormal
    OBJECT_STATE_GET_PAIR_IMG_ERROR,     // Paired frame acquisition abnormal
    OBJECT_STATE_RUNNING,                 // Running state
    OBJECT_STATE_SUCCESS,                 // Comparison/enrollment successful
    OBJECT_STATE_LOAD_MODEL_ERROR,        // Model loading error
    OBJECT_STATE_CALI_ERROR,              // Device uncalibrated
    OBJECT_STATE_DETECTED,                // Target detected after no target detected for a
period of time
    OBJECT_STATE_NOOBJECT_LONG_TIME,      // No target detected for a long period of time
    OBJECT_STATE_OTA_RUNNING = 120,       // Firmware upgrade in progress
    OBJECT_STATE_DET_QRCODE = 130,        // QR code detected
    OBJECT_STATE_DET_QRCODE_TIME_OUT = 131, // QR code detection timeout
} FACE_PASS_PPLCODE_E;
```

FACE_PASS_FACE_ERR_E

Overview

Face image enrollment error information

```
typedef enum
{
    FACE_PASS_FACE_OK= 0x0,
    FACE_PASS_FACE_ERR_FORMAT= 0x1,       // Image preprocessing error
    FACE_PASS_FACE_ERR_SIZE= 0x2,
    FACE_PASS_FACE_ERR_SIZE_SMALL= 0x3,
    FACE_PASS_FACE_ERR_RESOLUTION= 0x4,
    FACE_PASS_FACE_ERR_NO_MEM= 0x5,
    FACE_PASS_FACE_ERR_NO_FEATURE= 0x10,  // Face detection/enrollment error
    FACE_PASS_FACE_ERR_NO_PIC= 0x11,
    FACE_PASS_FACE_ERR_NO_FACE= 0x12,
    FACE_PASS_FACE_ERR_MUTI_FACE= 0x13,
    FACE_PASS_FACE_ERR_MINI_FACE= 0x14,
    FACE_PASS_FACE_ERR_BRIGHT= 0x15,
    FACE_PASS_FACE_ERR_DARK= 0x16,
    FACE_PASS_FACE_ERR_BLUR= 0x17,
    FACE_PASS_FACE_ERR_POSE= 0x18,
    /* Enrollment processing error */
    FACE_PASS_FACE_ERR_FULL= 0x20,           // Database full
    FACE_PASS_FACE_ERR_REPEAT_IMAGE= 0x21,   // Duplicate image
    FACE_PASS_FACE_ERR_REPEAT_FACENO= 0x22,  // Duplicate number
    FACE_PASS_FACE_ERR_REPEAT_CARDNO= 0x23,  // Duplicate card number
    FACE_PASS_FACE_ERR_REPEAT_FACE_ID= 0x24,
    FACE_PASS_FACE_ERR_REPEAT_INVALID_ID= 0x25,
    /* Other errors */
    FACE_PASS_FACE_ERR_FACE_NAME= 0x30,      // Image naming error during import
    FACE_PASS_FACE_ERR_IMAGE_PORCESS= 0x31,  // Image conversion error
    /* Authentication error processing */
    FACE_PASS_FACE_ERR_NO_OBJ= 0x40,         // No target detected
    FACE_PASS_FACE_ERR_EXTRACT= 0x41,        // Feature extraction failed
    FACE_PASS_FACE_ERR_LIVENESS= 0x42,       // Liveness detection failed
    FACE_PASS_FACE_ERR_OBJ_TYPE= 0x43,       // Type mismatch
    FACE_PASS_FACE_ERR_MATCH= 0x44,          // Not matched
    FACE_PASS_FACE_ERR_HANDLEINVAL= 0x50,    // Invalid object type
    FACE_PASS_FACE_ERR_HANDLEOPEND= 0x51,   // Algorithm aborted
    FACE_PASS_FACE_ERR_NO_CMD= 0x52,        // Unsupported command
    FACE_PASS_FACE_ERR_PARAM= 0x53,         // Parameter error
    FACE_PASS_FACE_ERR_TIMEOUT= 0x54,       // Enrollment timeout
    FACE_PASS_FACE_ERR_CMD_NOT_MATCH= 0x55,
    FACE_PASS_FACE_ERR_FEATURE_LEN= 0x56,
    FACE_PASS_FACE_ERR_BATCH_INSERT_CNT= 0x57,
    FACE_PASS_FACE_ERR_UNSUPPORT_ALG_TYPE  = 0x58,
    FACE_PASS_FACE_ERR_OUT_OF_MAX_USERS_LIMIT  = 0x59,
    FACE_PASS_FACE_ERR_CHECK_FACE_CNTS= 0x60,
    FACE_PASS_FACE_ERR_CHECK_PALM_CNTS= 0x61,
    FACE_PASS_FACE_ERR_ALG_ADD_USER= 0x63,  // Algorithm internal user addition abnormal
    FACE_PASS_FACE_ERR_UNKNOW= 0xFF,
} FACE_PASS_FACE_ERR_E;
```

FrameCallback

```
typedef void (*FrameCallback)(const uint8_t *data, int width, int height, int stride, void
*userData);
```

Description: Frame data callback function type

Example

Refer to face_pass_test.c