Rockchip Rkfacial Introduction

ID: RK-SM-YF-363

Release Version: V2.0.3

Release Date: 2022-02-16

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document is going to introduce the interface of each module of rkfacil.

Product Version

Chipset	Kernel Version
RK1808, RK1806	Linux 4.4
RV1126, RV1109	Linux 4.19

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Date	Version	Author	Change Description
2020-05-21	V1.0.0	Zhihua Wang	Initial version
2020-06-08	V1.1.0	Zhihua Wang	Modify user information callback
2020-07-28	V2.0.0	Zhihua Wang	Update interface and flowchart
2020-10-19	V2.0.1	Zhihua Wang	Add RV1126, RV1109 support
2021-03-15	V2.0.2	Ruby Zhang	Update product version information
2022-02-16	V2.0.3	Ruby Zhang	Update the format of the document

Contents

1. Code module Introduction

1.1 rockface_control

1.1.1 int rockface_control_init(int face_cnt)

Description

It is used to initialize each algorithm of rockface and face database, and extract the facial feature values from the jpg file in the specified directory to the database.

Parameter

face_cnt: the maximum number of faces supported by the face database

Return

int 0 succeeds, -1 fails

1.1.2 void rockface_control_exit(void)

Description

It used to de-initialization of each algorithm of rockface.

Parameter

void

Return

void

1.1.3 int rockface_control_add_ui(int id, const char *name, void *feature)

Description

Register users through the UI

Parameter

id user id

name username

feature user's facial feature value

Return

int 0 success

1.1.4 int rockface_control_add_web(int id, const char *name)

Description

Register users through web server

Parameter

id　　user id

name　username

Return

int　0 success

1.1.5 int rockface_control_add_local(const char *name)

Description

Register users through local storage of pictures

Parameter

name　username

Return

int　0 success

1.1.6 int rockface_control_delete(int id, const char *pname, bool notify)

Description

Delete users

Parameter

id　user id

pname　user name, use to delete by web server

notify　whether to notify the web server

Return

int 0 success

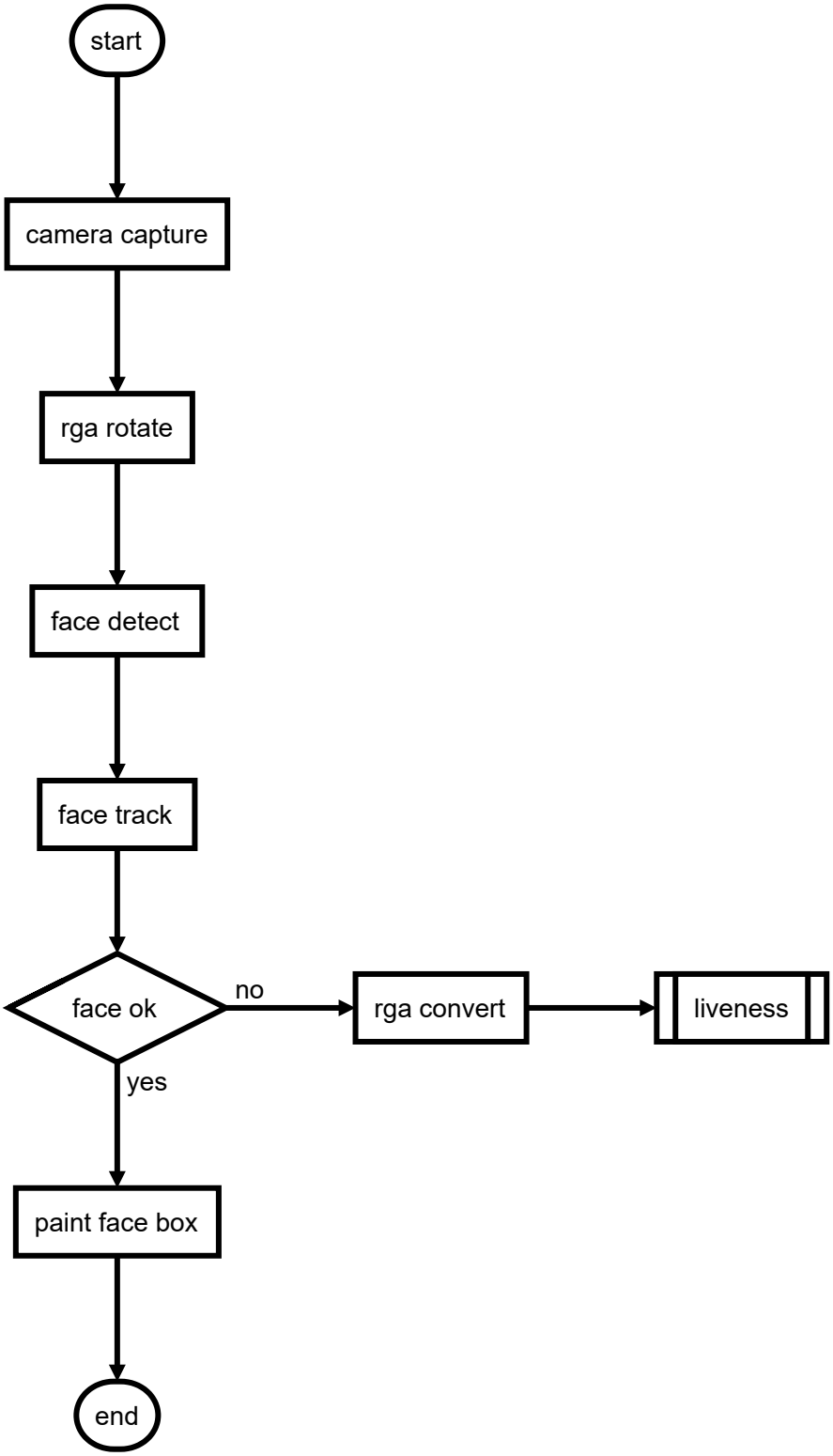
Default macro definition description:

```
#define DEFAULT_FACE_NUMBER 1000 //Indicates the maximum number of faces
supported by the default face database
#define DEFAULT_FACE_PATH "/userdata" //By default, load the jpg file from this
directory to obtain the feature value at boot
#define FACE_DETECT_SCORE 0.55 //The score of face detection, range from 0 to 1,
the larger the value the more stringent
#define FACE_SCORE_LANDMARK_RUNNING 0.9 //The score of face feature value using
RGB preview, range 0-1, the larger the value the more stringent
```

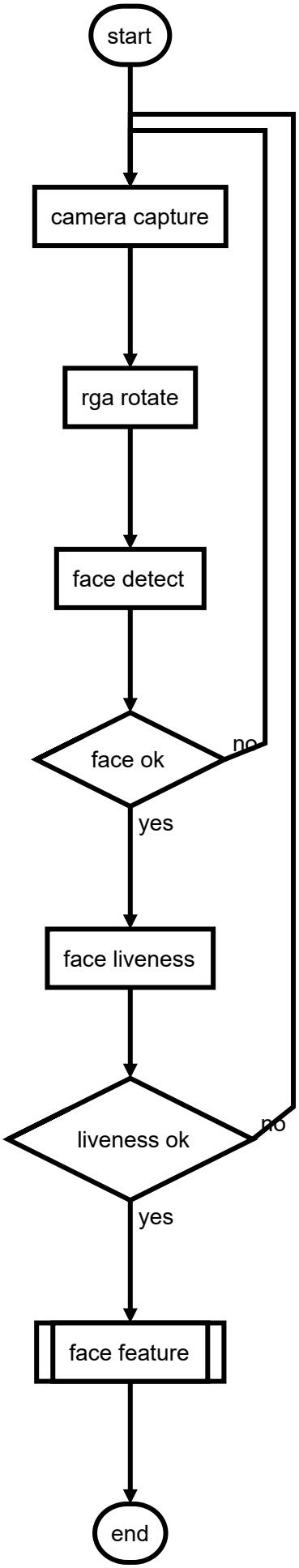
```
#define FACE_SCORE_LANDMARK_IMAGE 0.5 //The score of face feature value of RGB photo, the range 0-1, the larger the value the more stringent
#define FACE_SIMILARITY_CONVERT(f) powf(2.0, -((f))) //RGB face recognition score conversion formula
#define FACE_SIMILARITY_SCORE 1.0 //RGB face recognition score, the recommended range 0.7-1.3, the smaller the value the more stringent
#define FACE_SCORE_REGISTER 0.99 //The face score of face registration, the range 0-1, the larger the value the more stringent
#define FACE_REGISTER_CNT 5 //The number of consecutive face feature value read during face registration is in the database, indicating that it has been registered
#define FACE_REAL_SCORE 0.5 //Minimum requirement for live detection score, range 0-1, the larger the value the more stringent
#define LICENCE_PATH PRE_PATH "/key.lic" //rockface face authorization key storage path
#define BAK_LICENCE_PATH BAK_PATH "/key.lic" //rockface  face authorization backup key storage path
#define FACE_DATA_PATH "/usr/lib" //rockface data storage path
#define MIN_FACE_WIDTH(w) ((w) / 5) //Face detection, feature value extraction, face minimum pixel requirements
#define FACE_TRACK_FRAME 0 //Maximum tracking time of face tracking (frame)
#define FACE_RETRACK_TIME 1 //re-tracking time of face tracking  (seconds)
#define SNAP_TIME 3 //The minimum time between snapshots (seconds)
```

Application Flow Chart

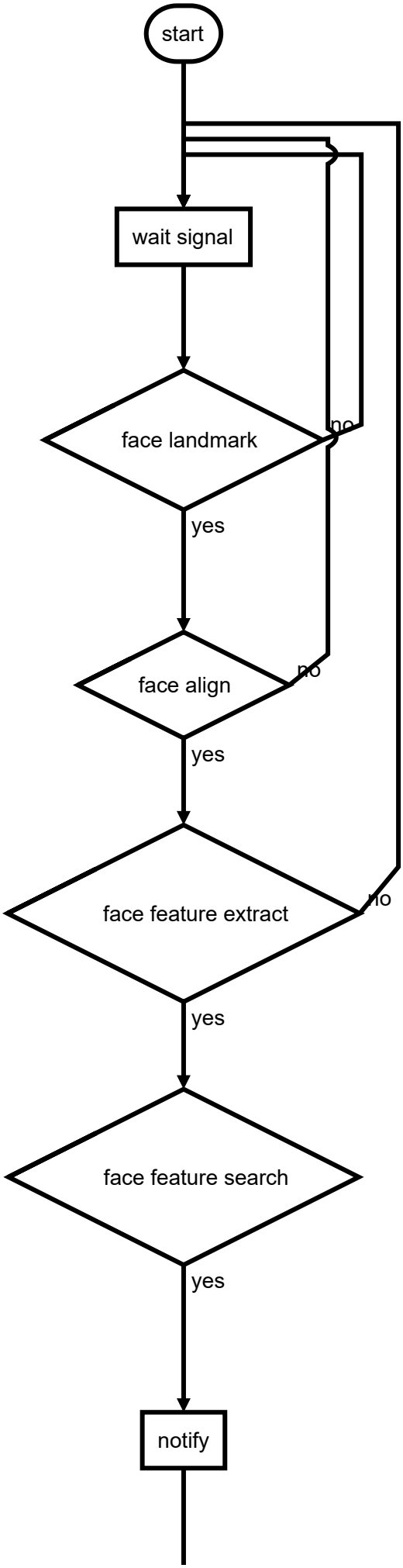
- RGB face detection

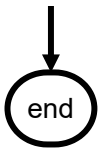


- IR face detection, live detection



- RGB Face feature value recognition


start
wait signal
face landmark
no
yes
face align
no
yes
face feature extract
no
yes
face feature search
yes
notify



1.2 datebase

1.2.1 int database_init(void)

Description

Used to initialize the database

Parameter

void

Return

int 0 succeeds, -1 fails

1.2.2 void database_exit(void)

Description

Used to deinitialization of the database

Parameter

void

Return

void

1.2.3 void database_bak(void)

Description

Complete the database backup

Parameter

void

Return

void

1.2.4 int database_insert(void *data, size_t size, char *name, size_t n_size, bool sync_flag)

Description

Used to insert a piece of data into the database

Parameter

data feature value data address

size the size of feature value

name username

n_size username size

sync_flag When sync_flag is true, the database will be synchronized and saved in real time

Return

int 0 succeeds, -1 fails

1.2.5 int database_record_count(void)

Description

Get the number of recorded face feature values

Parameter

void

Return

int the number of face feature values recorded

1.2.6 int database_get_data(void *dst, const int cnt, size_t d_size, size_t d_off, size_t n_size, size_t n_off)

Description

Extract the feature values with the largest cnt from the database and save them to dst, which will provide rockface for feature value database search and matching

Parameter

dst Pointer to store data

cnt The maximum number of feature values can be extracted

d_size The size of feature value

d_off The offset of the characteristic value in the user data structure

n_size The size of the name

n_off The offset of the name in the user data structure

Return

int The number of feature values obtained

1.2.7 bool database_is_name_exist(char *name)

Description

Determine whether the user name already exists in the database

Parameter

name username

Return

bool true: exists, false: does not exist

1.2.8 bool database_is_id_exist(int id, char *name, size_t size)

Description

Determine whether the user id already exists in the database

Parameter

id User id

name User name storage address

size The storage size of user name

Return

bool true: exists, false: does not exist

1.2.9 int database_get_user_name_id(void)

Description

The id number that the user can use when registering in real time. The id number starts from 0, and the unused one can be used to register for users.

Parameter

void

Return

int The id number that can be used

1.2.10 void database_delete(char *name, bool sync_flag)

Description

Delete a record in the database by user name

Parameter

name Username

sync_flag When sync_flag is true, the database will be synchronized and saved in real time

Return

void

1.3 db_monitor

1.3.1 void db_monitor_init()

Description

db_monitor initialization, used to receive web server related messages

Parameter

void

Return

void

1.3.2 void db_monitor_face_list_add(int id, char *path, char *name, char *type)

Description

Send the added user message to the web server

Parameter

id User id

path User path

name Username

type User type

Return

void

1.3.3 void db_monitor_face_list_delete(int id)

Description

Send deleted user message to web server

Parameter

id user id

Return

void

1.3.4 void db_monitor_snapshot_record_set(char *path)

Description

Send a snapshot message to the web server

Parameter

path The snapshot path

Return

void

1.3.5 void db_monitor_control_record_set(int face_id, char *path, char *status, char *similarity)

Description

Send control message to web server

Parameter

face_id User id

path The snapshot path

status Switch status

similarity

Return

void

1.3.6 void db_monitor_get_user_info(struct user_info *info, int id)

Description

Obtain user information through web server

Parameter

info Storing user information

id User id

Return

void

1.4 display

1.4.1 int display_init(int width, int height)

Description

Initialize the display plane of double plane vop video

Parameter

width The width of the screen

height The height of the screen

Return

void

1.4.2 void display_exit(void)

Description

The display plane of double plane vop video deinitialization

Parameter

void

Return

void

1.4.3 void display_switch(enum display_video_type type)

Description

Switch IR/RGB/USB camera display

Parameter

type The type of camera that needs to be displayed

Return

void

1.5 camrgb_control

1.5.1 int camrgb_control_init(void)

Description

It used to initialize rgb video, and the image data is read, rotated, displayed and sent to the rockface for face recognition, detection, feature value extraction, and trajectory tracking in the process thread.

Parameter

void

Return

int 0: succeeds, -1: fails

1.5.2 void camrgb_control_exit(void)

Description

It used to realize the de-initialization of rgb video.

Parameter

void

Return

void

1.5.3 void camrgb_control_expo_weights_270(int left, int top, int right, int bottom)

Description

It is used to realize partial exposure of rgb image rotated 270 degrees clockwise, which can realize partial exposure of face coordinates in dark environment.

Parameter

left The coordinates of the left side of the face rectangle

top The coordinatesof the top of the face rectangle

right The coordinates of the right of the face rectangle

bottom The coordinates of the bottom of the face rectangle

Return

void

1.5.4 void camrgb_control_expo_weights_90(int left, int top, int right, int bottom)

Description

It is used to realize partial exposure of rgb image rotated 90 degrees clockwise, which can realize partial exposure of face coordinates in dark environment.

Parameter

left The coordinates of the left side of the face rectangle

top The coordinatesof the top of the face rectangle

right The coordinates of the right of the face rectangle

bottom The coordinates of the bottom of the face rectangle

Return

void

1.5.5 void camrgb_control_expo_weights_default(void)

Description

Used to configure and restore the default exposure settings.

Parameter

void

Return

void

1.5.6 void set_rgb_display(display_callback cb)

Description

Set RGB camera display callback

Parameter

cb RGB camera display callback

Return

void

1.5.7 void set_rgb_rotation(int angle)

Description

Set the rotation angle of the RGB camera

Parameter

angle rotation angle, support 90, 270

Return

void

1.6 camir_control

1.6.1 int camir_control_init(void)

Description

Used to realize the initialization of ir video, and the image data is read, rotated, and sent to the rockface for live detection processing in the process thread.

Parameter

void

Return

int 0: succeeds, -1: fails

1.6.2 void camir_control_exit(void)

Description

Used to realize the de-initialization of ir video.

Parameter

void

Return

void

1.6.3 bool camir_control_run(void)

Description

Determine whether the IR camera is running

Parameter

void

Return

bool true: running; false: not running

1.6.4 void set_ir_display(display_callback cb)

Description

Set IR camera display callback

Parameter

cb IR camera display callback

Return

void

1.6.5 void set_ir_rotation(int angle)

Description

Set the rotation angle of the IR camera

Parameter

angle rotation angle, support 90, 270

Return

void

1.7 shadow_display

1.7.1 void shadow_display(void *src_ptr, int src_fd, int src_fmt, int src_w, int src_h)

Description

To realize the function of displaying the camera image on the horizontal screen, a more suitable image will be cropped according to the screen ratio and the camera image ratio and displayed on the horizontal screen.

Parameter

src_ptr image data address

src_fd image data fd

src_fmt image data format

src_w the width of the image

src_h the height of the image

Return

void

1.7.2 void shadow_display_vertical(void *src_ptr, int src_fd, int src_fmt, int src_w, int src_h)

Description

Realize the function of displaying the camera image in the vertical screen, according to the screen ratio and the camera image ratio, a more suitable image will be cropped and displayed on the vertical screen.

Parameter

src_ptr image data address

src_fd image data fd

src_fmt image data format

src_w the width of the image

src_h the height of the image

Return

void

1.7.3 void shadow_paint_box(int left, int top, int right, int bottom)

Description

Send the message of drawing the face rectangle frame to the UI.

Parameter

left The coordinates of the left side of the face rectangle frame

top The coordinates of the top of the face rectangle frame

right The coordinates of the right of the face rectangle frame

bottom The coordinates of the bottom of the face rectangle frame

Return

void

1.7.4 void shadow_paint_info(struct user_info *info, bool real)

Description

Send user information message to UI.

Parameter

info User information

Return

void

1.7.5 void shadow_get_crop_screen(int *width, int *height)

Description

Get the screen size range after cropping.

Parameter

width The width of the screen

height The height of the screen

Return

void

1.8 load_feature

1.8.1 int count_file(const char *path, char *fmt)

Description

Calculate the number of files in a certain directory including all corresponding image formats under the subdirectories.

Parameter

path The path of the directory

fmt The format of the image

Return

int Number of files

1.8.2 int load_feature(const char *path, char *fmt, void *data, unsigned int cnt)

Description

Read the feature value and file name of a certain directory including all subdirectories to the data structure pointer, the maximum number can be read is cnt.

Parameter

path Directory path

fmt Image format

data The pointer used to storage the read feature value and file name

cnt The maximum number can be read

Return

void

1.9 play_wav

1.9.1 int play_wav_thread_init(void)

Description

Used to initialize play_wav and play_wav_thread is completed. The play_wav_thread is waiting to receive the signal to play the specified wav file.

Parameter

void

Return

int 0: succeeds, -1: fails

1.9.2 void play_wav_thread_exit(void)

Description

Used to de-initialize play_wav and the play thread.

Parameter

void

Return

void

1.9.3 void play_wav_signal(char *name)

Description

Play wav audio by specifying the name.

Parameter

name The name of the wav audio file

Return

void

The audio format requires 16000 sampling rate, dual channel, 16bit, and the following 3 macros can be modified to specify other audio formats.

```
#define NUM_CHANNELS 2
#define SAMPLE_RATE 16000
#define BITS_PER_SAMPLE 16
```

Add wav audio:add Chinese to wav/cn, and English to wav/en.

The Install (DIRECTORY wav/cn/DESTINATION ../etc) in CMakeLists.txt is used to specify Chinese or English audio files to install in the specified directory.

1.10 rga_control

1.10.1 int rga_control_buffer_init(bo_t *bo, int *buf_fd, int width, int height, int bpp)

Description

Request drm memory

Parameter

bo Apply for bo parameters of target memory

buf_fd Apply for buf_fd parameters of target memory

width Apply for the width of the target memory

height Apply for the height of the target memory

bpp Apply for the bit corresponding to a pixel in the target memory

Return

void

1.10.2 void rga_control_buffer_deinit(bo_t *bo, int buf_fd)

Description

Release drm memory

Parameter

bo Apply for bo parameters of target memory

buf_fd Apply for buf_fd parameter of target memory

Return

void

1.11 rkfacial

1.11.1 typedef void (*display_callback)(void *ptr, int fd, int fmt, int w, int h, int rotation)

Description

Show callback

Parameter

ptr Memory address of the buffer

fd The memory address of buffer corresponding to fd

fmt The format of buffer

w The width of the buffer

h The height of the buffer

rotation The rotation parameters of the buffer (refer to linux-rga definition)

Return

void

1.11.2 void set_rgb_param(int width, int height, display_callback cb, bool expo)

Description

Set the parameters of the rgb camera

Parameter

width The initialization width of RGB camera

height The initialization height of RGB camera

cb RGB camera display callback, which can be NULL

expo RGB camera partial exposure

Return

void

1.11.3 void set_ir_param(int width, int height, display_callback cb)

Description

Set the parameters of the IR camera

Parameter

width The initialization width of IR camera

height The initialization height of IR camera

cb IR camera display callback, which can be NULL

Return

void

1.11.4 void set_usb_param(int width, int height, display_callback cb)

Description

Set the parameters of USB camera

Parameter

width The initialization width of USB camera

height The initialization height of USB camera

cb USB camera display callback, which can be NULL

Return

void

1.11.5 void set_face_param(int width, int height, int cnt)

Description

Set the initialization parameters of face

Parameter

width The initialization width of face

height The initialization height of face

cnt The Maximum number of faces

Return

void

1.11.6 int rkfacial_init(void)

Description

rkfacial initialization

Parameter

void

Return

int 0: succeeds, -1: fails

1.11.7 void rkfacial_exit(void)

Description

Exit rkfacial

Parameter

void

Return

void

1.11.8 void rkfacial_register(void)

Description

Used rkfacial to registered face

Parameter

void

Return

void

1.11.9 void rkfacial_delete(void)

Description

Use rkfacial to delete face

Parameter

void

Return

void

1.11.10 void register_rkfacial_paint_box(rkfacial_paint_box_callback cb)

Description

Register UI to draw face frame callback

Parameter

cb Use UI to draw face frame callback

Return

void

1.11.11 void register_rkfacial_paint_info(rkfacial_paint_info_callback cb)

Description

Registered UI to draw user information callback

Parameter

cb UI drawing user information callback

Return

void

1.12 snapshot

1.12.1 int snapshot_init(struct snapshot *s, int w, int h)

Description

snapshot initialization

Parameter

s The information of snapshot

w The width of snapshot picture

h The high of snapshot picture

Return

int 0 success

1.12.2 void snapshot_exit(struct snapshot *s)

Description

snapshot deinitialization

Parameter

s The information of snapshot

Return

void

1.12.3 int snapshot_run(struct snapshot *s, rockface_image_t *image, rockface_det_t *face, RgaSURF_FORMAT fmt, long int sec, char mark)

Description

snapshot

Parameter

s The information of snapshot

image Image

face Face

fmt The format of image

sec Minimum interval (in seconds) of snapshot

mark The mark of snapshot

Return

int 0:success

1.13 turbojpeg_decode

1.13.1 void *turbojpeg_decode_get(const char *name, int *w, int *h, int *b)

Description

Using turbojpeg decode MJPEG to get image buffer

Parameter

name The name of the image path

w The width of the stored image resolution

h The height of the stored image resolution

b The number of bytes per pixel of the stored image

Return

void * The buffer after MJPEG decoding

1.13.2 void turbojpeg_decode_put(void *data)

Description

Release turbojpeg decode MJPEG to get the buffer of the image

Parameter

data The buffer after MJPEG decode

Return

void

1.14 usb_camera

1.14.1 int usb_camera_init(void)

Description

usb camera initialization

Parameter

void

Return

int 0: success

1.14.2 void usb_camera_exit(void)

Description

USB camera deinitialization

Parameter

void

Return

void

1.14.3 void set_usb_display(display_callback cb)

Description

Set usb camera display callback

Parameter

cb display callback

Return

void

1.14.4 void set_usb_rotation(int angle)

Description

Set usb camera

Parameter

angle Rotation angle, support 90, 270

Return

void

1.15 vpu decode(MJPEG decode)

1.15.1 int vpu_decode_jpeg_init(struct vpu_decode* decode, int width, int height)

Description

vpu decode initialization

Parameter

decode vpu_decode information

width The width of MJPEG image

height The height of MJPEG image

Return

int 0: success

1.15.2 int vpu_decode_jpeg_doing(struct vpu_decode* decode, void* in_data, RK_S32 in_size, int out_fd, void* out_data)

Description

vpu decode

Parameter

decode vpu_decode information

in_data MJPEG image data

in_size MJPEG image size

out_fd Decoded data fd

out_data Decoded data data

Return

int 0:success

1.15.3 int vpu_decode_jpeg_done(struct vpu_decode* decode)

Description

vpu decode deinitialization

Parameter

decode vpu_decode information

Return

int 0: success

1.16 vpu encode(MJPEG encode)

1.16.1 int vpu_encode_jpeg_init(struct vpu_encode* encode, int width, int height, int quant, MppFrameFormat format)

Description

vpu encode initialization

Parameter

encode vpu_encode information

width The width of MJPEG image

height The height of MJPEG image

quant The quant of MJPEG encoding

format The format of encoding input image

Return

int 0: success

1.16.2 int vpu_encode_jpeg_doing(struct vpu_encode* encode, void* srcbuf, int src_fd, size_t src_size, void *dst_buf, int dst_fd, size_t dst_size)

Description

vpu encode initialization

Parameter

encode vpu_encode information

srcbuf The buffer of input image

src_fd The fd of input image

src_size The size of input image

dst_buf The buf of output image

dst_fd The fd of output image

dst_size The initialization size of output image buf

Return

int 0: success

1.16.3 void vpu_encode_jpeg_done(struct vpu_encode* encode)

Description

vpu encode deinitialization

Parameter

encode vpu_encode information

Return

void