SmartAR™ Reference

Ver 1.1

Change History

Version	Division of Work	Item	Changed Contents	Date
1.0	Created	-	Document created	2016/01/27
1.0.1	Changed	General	Delete track changes	2016/02/01
1.1	Updated	General	Updated document version	2016/03/28

Table of Contents

Common APIs

Constants

List of Definitions

Definition	Value	Description
SAR_OK	0	Normal end.
SAR_ERROR_ALREADY_INITIALIZED	-2138308607	Initialized. SmartAR™ SDK has already been initialized. (0x808c0001)
SAR_ERROR_ALREADY_REGISTERED	-2138308601	Already registered. Input has already been registered. (0x808c0007)
SAR_ERROR_ALREADY_STARTED	-2138308599	Already started. SmartAR™ SDK has already been started up. (0x808c0009)
SAR_ERROR_BUSY	-2138308594	Locked. SmartAR™ SDK is locked by a call from another thread. (0x808c000e)
SAR_ERROR_INVALID_POINTER	-2138308602	Invalid pointer. The pointer given is invalid. (0x808c0006)
SAR_ERROR_INVALID_VALUE	-2138308603	Invalid value. The value given is invalid. (0x808c0005)
SAR_ERROR_NO_DICTIONARY	-2138308595	Recognition target not registered. The recognition target has not been registered. (0x808c000d)
SAR_ERROR_NOT_EMPTY	-2138308604	Resource not released. Some resources have not been released yet. (0x808c0004)
SAR_ERROR_NOT_REGISTERED	-2138308600	Not registered. Input has not been registered yet. (0x808c0008)
SAR_ERROR_NOT_REQUIRED	-2138308597	Not ready. SmartAR™ SDK is not yet ready to process the request. (0x808c000b)
SAR_ERROR_NOT_STARTED	-2138308598	Not started. SmartAR™ SDK has not been started up yet. (0x808c000a)
SAR_ERROR_NOT_STOPPED	-2138308605	Not stopped. SmartAR™ SDK has not stopped yet. (0x808c0003)
SAR_ERROR_OUT_OF_MEMORY	-2138308606	Insufficient memory. SmartAR™ SDK failed to allocate memory. (0x808c0002)

Definition	Value	Description
SAR_ERROR_UNINITIALIZED	-2138308608	Not initialized. SmartAR™ SDK has not been initialized yet. (0x808c0000)
SAR_ERROR_VERSION_MISSMATCH	-2138308596	Version mismatched. Given value or file is not in its correct version. (0x808c000c)

Enumerators

SarFacing

SarFacing direction of the camera.

Definition

```
#include <SarCommon.h>
enum SarFacing{
        SAR_FACING_BACK,
        SAR_FACING_FRONT,
};
```

Enumeration Values

Value	Description
SAR_FACING_BACK	Back-facing camera
SAR_FACING_FRONT	Front-facing camera

Description

This is the enumerator showing the direction of the camera installed on the device.

SarRotation

SarRotation angle.

Definition

```
#include <SarCommon.h>
enum SarRotation{
        SAR_ROTATION_0,
        SAR_ROTATION_90,
        SAR_ROTATION_180,
        SAR_ROTATION_270,
};
```

Enumeration Values

Value	Description
SAR_ROTATION_0	0 degree
SAR_ROTATION_90	90 degrees
SAR_ROTATION_180	180 degrees
SAR_ROTATION_270	270 degrees

Description

This is the enumerator showing the rotation angle.

SarImageFormat

Image format.

Definition

```
#include <SarCommon.h>
enum SarImageFormat{
        SAR_IMAGE_FORMAT_L8,
        SAR_IMAGE_FORMAT_YCRCB420,
        SAR_IMAGE_FORMAT_YCBCR420,
        SAR_IMAGE_FORMAT_RGBA8888,
        SAR_IMAGE_FORMAT_JPEG
};
```

Enumeration Values

Value	Description
SAR_IMAGE_FORMAT_L8	Monochrome 8bit
SAR_IMAGE_FORMAT_YCRCB420	YCrCb420
SAR_IMAGE_FORMAT_YCBCR420	YCbCr420
SAR_IMAGE_FORMAT_RGBA8888	RGBA, each 8bit
SAR_IMAGE_FORMAT_JPEG	JPEG

Description

This is the enumerator showing the image format.

Classes

SarMemoryAllocator

Memory allocator.

Definition

```
#include <SarCommon.h>
class SarMemoryAllocator {
public:
        virtual~ SarMemoryAllocator();
        virtual void* allocate(size_t size) = 0;
        virtual void deallocate(void* ptr) = 0;
};
```

Description

This is the class defining the memory allocator employed by the user.

SarStreamIn

Input stream.

Definition

```
#include <SarCommon.h>
class SarStreamIn : SarNonCopyable {
public:
        virtual ~SarStreamIn() {}
        virtual size_t sarRead(void* buf, size_t size) = 0;
};
```

Description

This is the class defining the input stream employed by the user.

SarStreamOut

Output stream.

Definition

```
#include <SarCommon.h>
class SarStreamOut : SarNonCopyable {
public:
        virtual ~SarStreamOut() {}
        virtual size_t sarWrite(const void* buf, size_t size) = 0;
};
```

Description

This is the class defining the output stream employed by the user.

SarFileStreamIn

Input stream from file.

Definition

```
#include <SarCommon.h>
class SarFileStreamIn : public SarStreamIn {
public:
        SarFileStreamIn(SarSmart* smart, const char* filepath);
        virtual ~SarFileStreamIn();
        bool sarIsConstructorFailed();
        virtual size_t sarRead(void* buf, size_t size);
};
```

Description

This is the class defining the input stream from file employed by the user.

SarFileStreamOut

Output stream to file.

Definition

```
#include <SarCommon.h>
class SarFileStreamOut : public SarStreamOut {
public:
        SarFileStreamOut(SarSmart* smart, const char* filepath);
        virtual ~SarFileStreamOut();
        bool sarIsConstructorFailed();
        virtual size_t sarWrite(const void* buf, size_t size);
};
```

Description

This is the class defining the output stream to file employed by the user.

SarAssetStreamIn

Input stream from resource.

Definition

```
#include <SarCommon.h>
class SarAssetStreamIn : public SarStreamIn {
public:
        SarAssetStreamIn(SarSmart* smart, const char* filepath);
        virtual ~SarAssetStreamIn();
        bool sarIsConstructorFailed();
        virtual size_t sarRead(void* buf, size_t size);
};
```

Description

This is the class defining the input stream from resource employed by the user.

SarVector2

2D vector.

Definition

```
#include <SarCommon.h>
struct SarVector2 {
        float x_;
        float y_;

        SarVector2();
        SarVector2(float x, float y);

        void set(float x, float y);
};
```

Members

*x_*	X
*y_*	Y

Description

This is the class of vectors defined by 2 float types.

SarVector3

3D vector.

Definition

```
#include <SarCommon.h>
struct SarVector3 {
        float x_;
        float y_;
        float z_;

        SarVector3();
        SarVector3(float x, float y, float z);

        void set(float x, float y, float z);

        SarVector3& operator+=(const SarVector3& rhs);
        SarVector3& operator-=(const SarVector3& rhs);
        SarVector3& operator*=(float rhs);
        SarVector3& operator/=(float rhs);
        SarVector3 operator+(const SarVector3& rhs) const;
        SarVector3 operator-(const SarVector3& rhs) const;
        SarVector3 operator*(float rhs) const;
        SarVector3 operator/(float rhs) const;
};
```

Members

*x_*	X
*y_*	Y
*z_*	Z

Description

This is the class of vectors defined by 3 float types.

SarQuaternion

SarQuaternion.

Definition

```
#include <SarCommon.h>
struct SarQuaternion {
        float w_;
        float x_;
        float y_;
        float z_;

        SarQuaternion();
        SarQuaternion(float w, float x, float y, float z);

        void set(float w, float x, float y, float z);

        SarQuaternion operator-() const;
        SarQuaternion operator*(const SarQuaternion& rhs) const;
        SarQuaternion& operator*=(const SarQuaternion& rhs);
};
```

Members

*w_*	W
*x_*	X
*y_*	Y
*z_*	Z

Description

This is the class of a quaternion defined by 4 float types.

SarMatrix44

4x4 matrix.

Definition

```
#include <SarCommon.h>
struct SarMatrix44 {
        static const int32_t NUM_VALUES = 16;

        float values_[NUM_VALUES];

        SarMatrix44();
        SarMatrix44(float values[16]);

        SarMatrix44 operator*(const SarMatrix44& rhs) const;
        SarVector3 operator*(const SarVector3& rhs) const;

        void sarSetRotation(const SarQuaternion& quat);
        void sarSetTranslation(float x, float y, float z);
        void sarSetScaling(float x, float y, float z);
};
```

Members

*values_*	Elements of the matrix

Definition

This is the class of a 4x4 matrix defined by 16 float types.

SarTriangle2

Array of vertices of a triangle

Definition

```
#include <SarCommon.h>
struct SarTriangle2 {
        static const int32_t NUM_POINTS = 3;

        SarVector2 points_[NUM_POINTS];
};
```

Members

*points_* SarVector2 types containing the 2D coordinate of vertices of the triangle.

Description

This is the class to represent a triangle defined by 3 SarVector2 types containing the 2D coordinate of its vertices.

SarTriangle3

Array of vertices of a triangle.

Definition

```
#include <SarCommon.h>
struct SarTriangle3 {
        static const int32_t NUM_POINTS = 3;

        SarVector3 points_[NUM_POINTS];
};
```

Members

*points_* SarVector3 types containing the 3D coordinate of vertices of the triangle.

Description

This is the class to represent a triangle defined by 3 SarVector3 types containing the 3D coordinate of its vertices.

SarSize

SarSize.

Definition

```
#include <SarCommon.h>
struct SarSize {
        int32_t width_;
        int32_t height_;

        SarSize();
        SarSize(int32_t width, int32_t height);

        bool operator==(const SarSize& rhs);
};
```

Members

*width_*	Width
*Height_*	Height

Definition

This is the class of size defined by 2 int32_t types.

SarRect

SarRectangle.

Definition

```
#include <SarCommon.h>
struct SarRect {
        int32_t left_;
        int32_t top_;
        int32_t right_;
        int32_t bottom_;

        SarRect();
        SarRect(int32_t left, int32_t top, int32_t right, int32_t bottom);

        void set(int32_t left, int32_t top, int32_t right, int32_t bottom);
        int32_t width() const;
        int32_t height() const;

        bool operator==(const SarRect& rhs) const;
        bool operator!=(const SarRect& rhs) const;
        SarRect operator/(int rhs) const;
};
```

Members

*left_*	X coordinate of the leftmost of the rectangle
*top_*	Y coordinate of the upper limit of the rectangle
*right_*	X coordinate of the rightmost of the rectangle
*bottom_*	Y coordinate of the lower limit of the rectangle

Description

This is the class of rectangle defined by 4 int32_t types.

SarImage

Image data.

Definition

```
#include <SarCommon.h>
class SarImage {
public:
        SarImage(SarSmart* smart);
        SarImage(const SarImage &other);
        ~SarImage();
        SarImage & operator = (const SarImage &other);

        void setData(unsigned char* pixels);
        unsigned char* getData();
        void setWidth(int32_t width);
        int32_t getWidth();
        void setHeight(int32_t height);
        int32_t getHeight();
        void setStride(int32_t stride);
        int32_t getStride();
        void setImageFormat(SarImageFormat format);
        SarImageFormat getImageFormat();
};
```

Description

This is the class to store image data.

SarImage::SarImage

Constructor

Definition

```
public SarImage(
        SarSmart* smart
);

public SarImage(
        const SarImage &other
);
```

Members

[in] smart	Pointer to the instance of SarSmart class
[in] other	Other instance of SarImage class

Description

This is the constructor of SarImage class.

SarImage::~SarImage

Destructor

Definition

public ~SarImage();

Description

This is the destructor of SarImage class.

SarImage::setData

Set pixel data

Definition

```
public void setData(
        unsigned char* pixels
);
```

Members

[in] pixels	Pointer to pixel data

Description

Set pixel data.

You can use this function with certified license.

SarImage::getData

Retrieve pixel data

Definition

public unsigned char* getData();

Return Values

Pointer to pixel data

Description

Return pointer to pixel data.
You can use this function with certified license.

SarImage::setWidth

Set width of the image

Definition

```
public void setWidth(
        int32_t width
);
```

Members

[in] width Width of the image

Description

Set width of the image.

SarImage::getWidth

Retrieve width of the image

Definition

public int32_t getWidth();

Return Values

Width of the image

Description

Return width of the image.

SarImage::setHeight

Set height of the image

Definition

```
public void setHeight (
        int32_t height
);
```

Members

[in] height Height of the image

Description

Set height of the image.

SarImage::getHeight

Retrieve height of the image

Definition

public int32_t getHeight();

Return Values

Height of the image

Description

Return height of the image.

SarImage::setStride

Set offset between rows

Definition

```
public void setStride (
        int32_t stride
);
```

Members

[in] stride	Offset between rows. 0 when offset is equal to the width

Description

Set offset from the end of a row to the head of next row.
0 when stride is the same with width of the image.

SarImage::getStride

Retrieve offset between rows

Definition

public int32_t getStride();

Return Values

Offset between rows. 0 when offset is equal to the width

Description

Return offset from the end of a row to the head of next row.
0 when stride is the same with width of the image.

SarImage::setImageFormat

Set format of the image

Definition

```
public void setImageFormat(
        SarImageFormat format
);
```

Members

[in] format Format of the image

Description

Set format of the image.

For more information about image format please refer to the description of SarImageFormat.

SarImage::getImageFormat

Retrieve format of the image

Definition

public SarImageFormat getImageFormat();

Return Values

Format of the image

Description

Return format of the image.
For more information about image format please refer to the description of SarImageFormat.

SarRecognizer

Constants

List of Definitions

Definition	Description
SAR_MAX_NUM_INITIALIZATION_POINTS	Maximum number of initialization points used by SarRecognizer.
SAR_MAX_NUM_LANDMARKS	Maximum number of landmarks used by SarRecognizer.
SAR_MAX_PROPAGATION_DURATION	Maximum duration estimation result can be propagated. [usec]

Enumerators

SarRecognitionMode

Process mode of recognition.

Definition

```
#include <SarRecognizer.h>
enum SarRecognitionMode {
        SAR_RECOGNITION_MODE_TARGET_TRACKING,
        SAR_RECOGNITION_MODE_SCENE_MAPPING
};
```

Values

Value	Description
SAR_RECOGNITION_MODE_TARGET_TRACKING	TargetTracking mode. TargetTracking mode is for recognizing planar object within camera image and estimate the pose of the device running SmartAR™SDK. Various natural images can be registered as recognition target.
SAR_RECOGNITION_MODE_SCENE_MAPPING	SceneMapping mode. Using camera image change and sensor information when a device running SmartAR™SDK is moved in the environment, SceneMapping mode estimates the 3D structure of the environment and the pose (location and proportion) of the device. SLAM (Simultaneous Localization and Mapping) technology is used in this mode.

Description

Enumerator represents the process modes of SarRecognizer.

SarSearchPolicy

Search policy of recognition process.

Definition

```
#include <SarRecognizer.h>
enum SarSearchPolicy {
        SAR_SEARCH_POLICY_FAST,
        SAR_SEARCH_POLICY_PRECISIVE,
};
```

Enumeration Values

Value	Description
SAR_SEARCH_POLICY_FAST	Prioritizes search speed. This search policy focuses on the rapid execution of a single search. The shorter time required for a single search allows executing a larger number of searches within the same period of time; however, the recognition target within the input image needs to be of a certain size. Therefore, use SAR_SEARCH_POLICY_PRECISIVE if you wish to search for smaller recognition targets
SAR_SEARCH_POLICY_PRECISIVE	Prioritizes search sensitivity. This policy focuses on search sensitivity. It requires a longer time for a single search than SAR_SEARCH_POLICY_FAST, but it allows searching for smaller recognition targets in the input image. Search time is about four times that required with SAR_SEARCH_POLICY_FAST, but the size of recognizable targets is about one fourth. If the recognition target in the input image is above a certain size, SAR_SEARCH_POLICY_FAST, which requires shorter time for a single search, may provide better recognition performance

Description

This is the enumeration value representing the recognition target's search policy. In its searching state, the TargetTracking library searches for a recognition target, and then transitions to tracking state after it finds a target. SarSearchPolicy represents the library's search policy when it is in searching state. The calculation time required for a search and the size of searchable target objects within an input image will vary depending on the policy used. This setting is only valid when the recognition target is a natural image.

SarSceneMappingInitMode

SLAM's initialization mode.

Definition

```
#include <SarRecognizer.h>
enum SarSceneMappingInitMode {
        SAR_SCENE_MAPPING_INIT_MODE_TARGET,
        SAR_SCENE_MAPPING_INIT_MODE_HFG,
        SAR_SCENE_MAPPING_INIT_MODE_VFG,
        SAR_SCENE_MAPPING_INIT_MODE_SFM,
        SAR_SCENE_MAPPING_INIT_MODE_DRY_RUN,
};
```

Enumeration Values

Value	Description
SAR_SCENE_MAPPING_INIT_MODE_TARGET	Performs initialization with a learned natural image. Assumes that a learned natural image is at the origin of the scene-fixed coordinate system, and performs initialization by detecting the natural image.
SAR_SCENE_MAPPING_INIT_MODE_HFG	This initialization mode estimates the horizontal plane (Horizontal From Gravity: HFG) using gravity direction. It uses the motion sensors to estimate the horizontal plane and initialize SLAM. Point the camera in the direction of horizontal planes with rich textures (pictures placed on a table, etc.). Note: This mode requires input from the motion sensors to find the gravity direction.
SAR_SCENE_MAPPING_INIT_MODE_VFG	This initialization mode estimates the vertical plane (Vertical From Gravity: VFG) using gravity direction. It uses the motion sensors to estimate the vertical plane and initialize SLAM. Point the camera in the direction of vertical plane with a rich texture (the surface of a building's wall, etc.). The camera and the vertical plane need to be directly confronting each other. Note: This mode requires input from the motion sensors to find the gravity direction.
SAR_SCENE_MAPPING_INIT_MODE_SFM	Tracks feature points within a 2D image, estimates a 3D structure from parallax when tracking begins and ends, and initializes SLAM (Structure From Motion: SFM). Instruct users to move the camera slowly and in a parallel fashion toward a texture-rich scene. Note: If motion sensor data is not passed, the photographed object will be processed by assuming that it is flat.

Value	Description
SAR_SCENE_MAPPING_INIT_MODE_DRY_RUN	This is the dry-run mode for SFM, HFG and VFG. It tracks the feature points used by SFM, HFG and VFG, and checks how many feature points are present within a 2D image. The number and position of feature points can be found with SarRecognizer::sarGetResult() or the member landmarks_ and initPoints_ of SarRecognitionResult structure returned from SarRecognizer::sarGetResults(). Actual initialization cannot be performed in this mode. Use the SFM, HFG or VFG initialization mode to perform initialization.

Description

This enumeration value represents the initialization mode used by the library to initialize SLAM estimation in SceneMapping mode.

SarTargetTrackingState

Recognition state of recognition targets

Definition

```
#include <SarRecognizer.h>
enum SarTargetTrackingState {
        SAR_TARGET_TRACKING_STATE_IDLE,
        SAR_TARGET_TRACKING_STATE_SEARCH,
        SAR_TARGET_TRACKING_STATE_TRACKING,
};
```

Enumeration Values

Value	Description
SAR_TARGET_TRACKING_STATE_IDLE	Idle state. The initial state of TargetTracking mode.
SAR_TARGET_TRACKING_STATE_SEARCH	Search state. Searching for registered recognition target within the camera image.
SAR_TARGET_TRACKING_STATE_TRACKING	Tracking state. Succeeded in searching for registered recognition target and the target are being tracked.

Description

This enumeration value represents the recognition state of SarRecognizer in TargetTracking mode.
State transition diagram of SarRecognizer in TargetTracking mode is shown as follows.

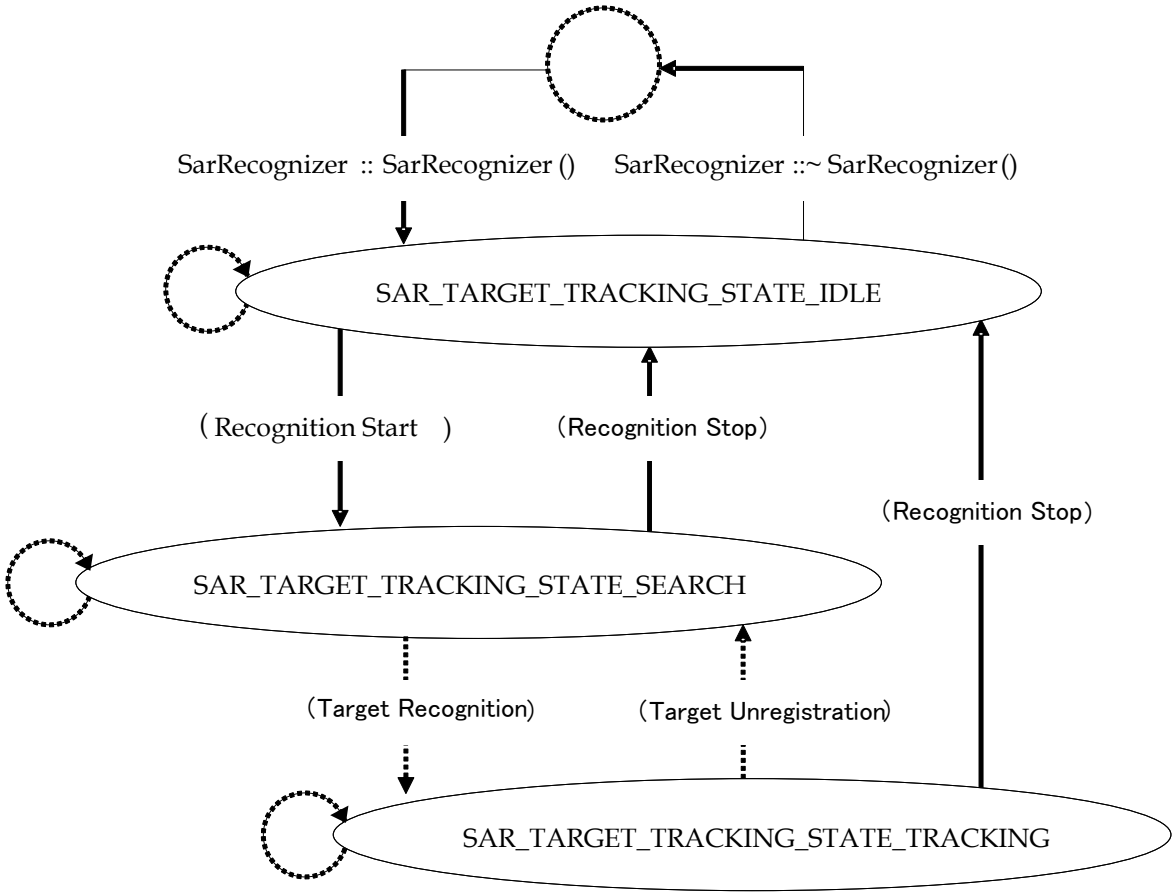

Figure 1 State Transition of SarRecognizer in TargetTracking Mode

SarSceneMappingState

Recognition state of SceneMapping mode.

Definition

```
#include <SarRecognizer.h>
enum SarSceneMappingState {
        SAR_SCENE_MAPPING_STATE_IDLE,
        SAR_SCENE_MAPPING_STATE_SEARCH,
        SAR_SCENE_MAPPING_STATE_TRACKING,
        SAR_SCENE_MAPPING_STATE_LOCALIZE,
        SAR_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE,
};
```

Enumeration Values

Value	Description
SAR_SCENE_MAPPING_STATE_IDLE	Idle state. In this state the library will remain inactive. To execute recognition process the library need to transit to SAR_SCENE_MAPPING_STATE_SEARCH and camera pose/scene map being initialized.
SAR_SCENE_MAPPING_STATE_SEARCH	Search (initialization) state. In this state, the SceneMapping library will attempt to initialize the camera's pose and the scene map in the specified initialization mode. When initialization succeeds, the state will automatically transit to the next one, SAR_SCENE_MAPPING_STATE_TRACKING.
SAR_SCENE_MAPPING_STATE_TRACKING	Tracking state. This state is the library's main state. While in this state, the library can satisfactorily estimate the camera's pose in relation to the scene, as well as the scene's 3D map. The scene's 3D map will be created gradually as the camera moves through the scene. Instruct users to move the camera back and forth, up and down, left and right as opposed to simply rotating it in the same spot. If the camera's pose cannot be estimated due to tracking fails or the camera moving too rapidly, etc. the state will automatically transition to SAR_SCENE_MAPPING_STATE_LOCALIZE.

Value	Description
SAR_SCENE_MAPPING_STATE_LOCALIZE	Localization state. In this state, the SceneMapping library will attempt to restore camera pose estimation by using a 3D map of the scene created while in SAR_SCENE_MAPPING_STATE_TRACKING state. Instruct users to return the camera to the pose it was in during the SAR_SCENE_MAPPING_STATE_TRACKING M state.
SAR_SCENE_MAPPING_STATE_LOCALIZE_IMPOSSIBLE	Localization impossible state. This state indicated that the library is unable to restore the camera's pose because, during SAR_SCENE_MAPPING_STATE_SLAM, it failed to obtain sufficient information on the 3D map required for attempting to restore the camera's pose in the SAR_SCENE_MAPPING_STATE_LOCALIZE state. Instruct users to initialize it again

Description

SarSceneMappingState represents the library's internal state in SceneMapping mode.
State transition diagram of SarRecognizer in SceneMapping mode is shown as follows.

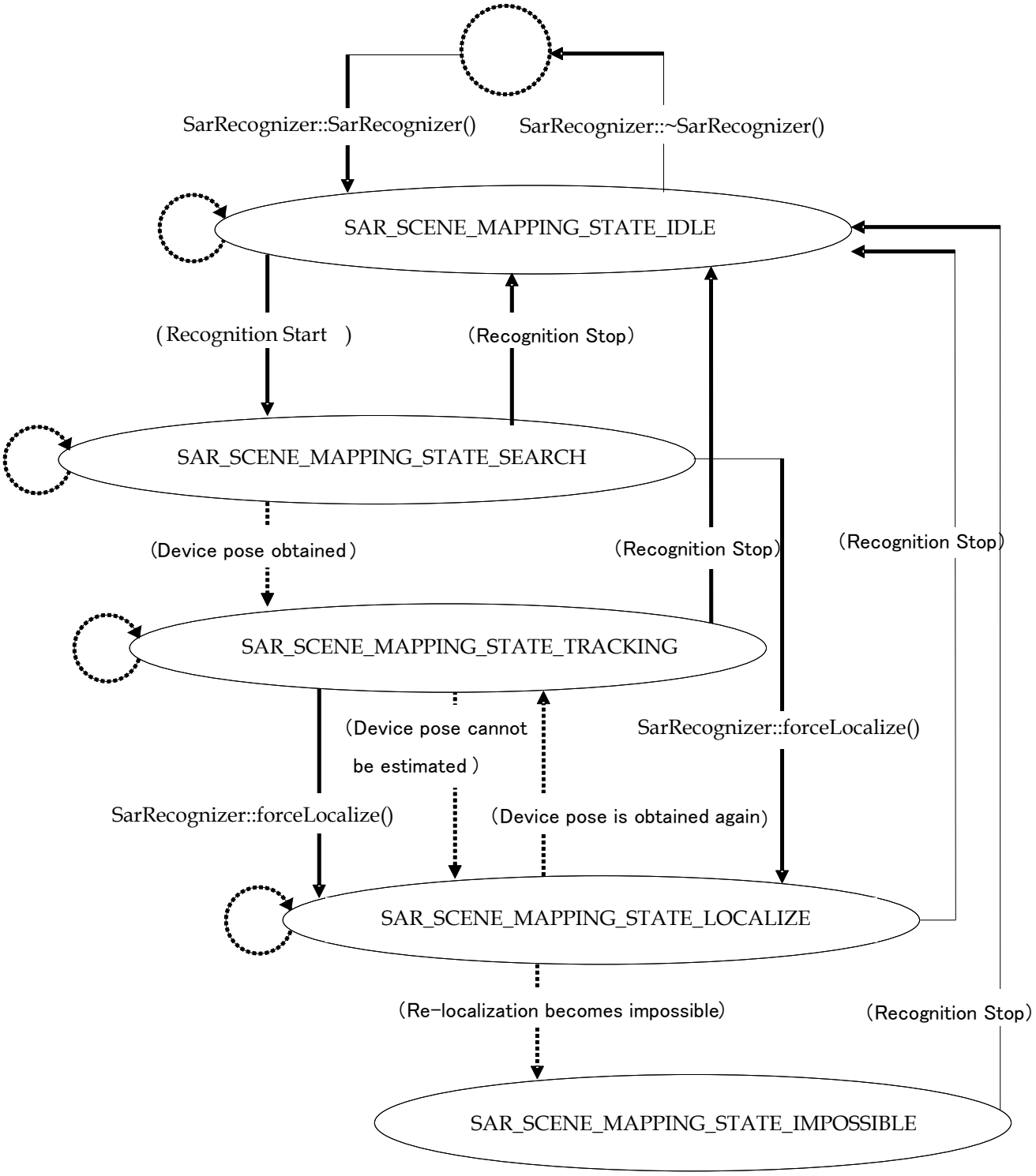

Figure 2 State Transition of the Library in SceneMapping Mode

SarLandmarkState

Tracking state of landmark

Definition

```
#include <SarRecognizer.h>
enum SarLandmarkState {
        SAR_LANDMARK_STATE_TRACKED,
        SAR_LANDMARK_STATE_LOST,
        SAR_LANDMARK_STATE_SUSPENDED,
        SAR_LANDMARK_STATE_MASKED,
};
```

Enumeration Values

Value	Description
SAR_LANDMARK_STATE_TRACKED	Tracking successful
SAR_LANDMARK_STATE_LOST	Tracking failed
SAR_LANDMARK_STATE_MASKED	Masking
SAR_LANDMARK_STATE_SUSPENDED	Other states

Description

This is the enumeration value to represent the tracking state of landmark. SarLandmarks are points in a scene whose 3D position has been estimated. The landmarks appearing in the previous input image and in the current input image are tracked, and their 3D positions are then estimated based on the change in their 2D positions.

SarDenseMapMode

DenseMap mode state.

Definition

```
#include <SarRecognizer.h>
enum SarDenseMapMode {
        SAR_DENSE_MAP_DISABLE,
        SAR_DENSE_MAP_SEMI_DENSE,
};
```

Enumeration Values

Value	Description
SAR_DENSE_MAP_DISABLE	Disable DenseMap mode
SAR_DENSE_MAP_SEMI_DENSE	Enable DenseMap mode

Description

This enumeration value represents the DenseMap mode state.

When DenseMap mode is enabled the density of landmark maintained in the scene map is increased.

Classes

SarRecognizer

Class to process recognition

Definition

```
#include <SarRecognizer.h>
class SarRecognizer : SarNonCopyable {
public:
        SarRecognizer(SarSmart* smart,
            SarRecognitionMode recogMode = SAR_RECOGNITION_MODE_TARGET_TRACKING,
            SarSceneMappingInitMode initMode = SAR_SCENE_MAPPING_INIT_MODE_TARGET);
        ~SarRecognizer();
        bool sarIsConstructorFailed();

        // setting
        int32_t sarSetCameraDeviceInfo(const SarCameraDeviceInfo& info);
        int32_t sarSetSensorDeviceInfo(const SarSensorDeviceInfo& info);
        int32_t sarSetTargets(const SarTarget* const* targets, int32_t numTargets);

        // start and stop
        int32_t sarReset();

        // run
        int32_t sarRun(const SarRecognitionRequest& request);
        int32_t sarDispatch(const SarRecognitionRequest& request);
        int32_t sarRunWorker();
        int32_t sarSetWorkDispatchedListener(SarWorkDispatchedListener* listener);

        // get results
        int32_t sarGetNumResults() const;
        int32_t sarGetResults(SarRecognitionResult* results, int32_t maxResults) const;
        int32_t sarGetResult(const SarTarget& target, SarRecognitionResult* result) const;
        int32_t sarSetRecognizedListener(SarRecognizedListener* listener);

        // at present for target tracking only
        int32_t sarSetMaxTargetsPerFrame(int32_t maxTargets);
        int32_t sarSetSearchPolicy(SarSearchPolicy policy);

        // at present for scene mapping only
        int32_t sarPropagateResult(const SarRecognitionResult& fromResult,
            SarRecognitionResult* toResult, uint64_t timestamp,
            bool useVelocity = true) const;
        int32_t sarSetMaxTriangulateMasks(int32_t maxMasks);

        // for scene mapping
        int32_t sarSaveSceneMap(SarStreamOut* stream) const;
        int32_t sarFixSceneMap(bool isFix);
        int32_t sarForceLocalize();
        int32_t sarRemoveLandmark(const SarLandmark& landmark);
        int32_t sarSetDenseMapMode(SarDenseMapMode mode);
};
```

Description

Recognition is executed in TargetTracking mode or SceneMapping mode.
More information for using SarRecognizer can be found in SmartSDK-Overciew.doc.

SarRecognizer::SarRecognizer

Constructor

Definition

```
public SarRecognizer(
        SarSmart* smart,
        SarRecognitionMode recogMode = SAR_RECOGNITION_MODE_TARGET_TRACKING,
        SarSceneMappingInitMode initMode = SAR_SCENE_MAPPING_INIT_MODE_TARGET
        );
```

Members

[in] smart	Pointer to the instance of SarSmart class
[in] recogMode	Recognition mode
[in] initMode	Initialization mode when using SceneMapping mode. This is only valid when SceneMapping mode is used.

Description

This is the constructor of SarRecognizer class.

SarRecognizer::~SarRecognizer

Destructor

Definition

public ~SarRecognizer();

Description

This is the destructor of SarRecognizer class.

SarRecognizer::sarIsConstructorFailed

Indication of constructor error

Definition

public bool sarIsConstructorFailed();

Return Values

Return true when any error occurs in constructor otherwise return false.

Description

This function is for check of error occurred in constructor.

SarRecognizer::sarSetCameraDeviceInfo

Set camera device information

Definition

```
public int32_t sarSetCameraDeviceInfo(
        const SarCameraDeviceInfo& info
        );
```

Arguments

[in] info	Camera device information

Return Values

Error code.

Description

Set Camera device information.

SarRecognizer::sarSetSensorDeviceInfo

Set sensor device information

Definition

```
public int32_t sarSetSensorDeviceInfo(
        const SarSensorDeviceInfo& info
        );
```

Arguments

[in] info	Sensor device information

Return Values

Error code.

Description

Set sensor device information.

SarRecognizer::sarSetTargets

Set/unset recognition targets

Definition

```
public int32_t sarSetTargets(
        const SarTarget* const* targets,
        int32_t numTargets
        );
```

Arguments

[in] targets	Pointer to array contains recognition targets. When unsetting all recognition targets set NULL to this argument.
[in] numTargets	Number of element in targets When unsetting all recognition targets set 0 to this argument.

Return Values

Error code.

Description

Set/unset recognition targets to SarRecognizer.
When unsetting all recognition targets, call sarSetTargets with NULL to targets and 0 to numTargets.

SarRecognizer::sarReset

Reset recognition state.

Definition

public int32_t sarReset();

Return Values

Error code.

Description

Reset current recognition state.

SarRecognizer::sarRun

Execute recognition process.

Definition

```
public int32_t sarRun(
        const SarRecognitionRequest& request
        );
```

Arguments

[in] request	Input data to recognition process

Return Values

Error code.

Description

Execute recognition process synchronously with input data.

SarRecognizer::sarDispatch

Asynchronous call to recognition process.

Definition

```
public int32_t sarDispatch(
        const SarRecognitionRequest& request
        );
```

Arguments

[in] request	Input data to recognition process

Return Values

Error code.

Description

Call recognition process asynchronously with the input data.
Actual recognition is executed in sarRunWorker().

SarRecognizer::sarRunWorker

Execution of asynchronous recognition process

Definition

public int32_t sarRunWorker();

Return Values

Error code.

Description

Execute recognition process requested by sarDispatch().
This function executes recognition process asynchronously by using worker thread generated by application.

SarRecognizer::sarSetWorkDispatchedListener

Set listener to receive execution request from sarRunWorker()

Definition

```
public int32 sarSetWorkDispatchedListener(
        SarWorkDispatchedListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarWorkDispatchedListener class

Return Values

Error code

Description

Set listener to receive execution request from sarRunWorker().

SarRecognizer::sarGetNumResults

Retrieve number of recognition results.

Definition

public int32_t sarGetNumResults() const;

Return Values

Number of recognition results

Description

Return number of recognition result return from SarRecognizer::sarGetResults().

SarRecognizer::sarGetResults

Retrieve multiple recognition result.

Definition

```
public int32_t sarGetResults(
        SarRecognitionResult* results,
        int32_t maxResults
        ) const;
```

Arguments

[out] results	Pointer to SarRecognitionResult array which contains recognition results
[in] maxResults	Maximum bumber of result to retrieve

Return Values

Number of results is returned if finished normally, otherwise error code.

Description

Retrieve multiple (up to maxResults) recognition result s.

SarRecognizer::sarGetResult

Retrieve specific recognition result

Definition

```
public int32_t sarGetResult(
        const SarTarget* target,
        SarRecognitionResult* result
        ) const;
```

Arguments

[in] target	Recognition target that is associated to the recognition result to retrieve
[out] result	Pointer to SarRecognitionResult structure to contain the recognition result

Return Values

Error code

Description

Retrieve specific recognition result.

SarRecognizer::sarSetRecognizedListener

Set listener to receive recognition result update notice

Definition

```
public int32_t sarSetRecognizedListener(
        SarRecognizedListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarRecognizedListener class

Return Values

Error code

Description

Set listener to receive recognition result update notice.

SarRecognizer::sarSetMaxTargetsPerFrame

Set maximum number of targets to recognize simultaneously

Definition

```
public int32_t sarSetMaxTargetsPerFrame(
        int32_t maxTargets
        );
```

Arguments

[in] maxTargets	Maximum number to recognize

Return Values

Error code

Description

Set number of targets to simultaneously recognize in TargetTracking mode.

SarRecognizer::sarSetSearchPolicy

Set search policy

Definition

```
public int32_t sarSetSearchPolicy(
        SarSearchPolicy policy
        );
```

Arguments

[in] policy	Policy to search recognition target

Return Values

Error code

Description

Set search policy of recognition target.

SarRecognizer::sarPropagateResult

Propagate recognition result along temporal axis

Definition

```
public int32_t sarPropagateResult(
        const SarRecognitionResult& fromResult,
        SarRecognitionResult* toResult,
        uint64_t timestamp,
        bool useVelocity = true
        ) const;
```

Arguments

[in] fromResult	Propagation source result
[out] toResult	Propagated result
[in] timestamp	Time of the propagation destination
[in] useVelocity	Flag deciding to use speed or not

Return Values

Error code

Description

This function propagates the recognition results of a given time to a specified time. Propagation will, to a certain extent, compensate for delays in calculation. However, propagation lasting longer than SAR_MAX_PROPAGATION_DURATION will not be performed, and input will be copied to output as it is. If propagation over a long period of time should be absolutely necessary, distribute it over several calls.

SarRecognizer::sarSetMaxTriangulateMasks

Set maximum number of triangular patch

Definition

```
public int32_t sarSetMaxTriangulateMasks(
        int32_t maxMasks
        );
```

Arguments

[in] maxMasks	Maximum number of triangular patch. Set to 0 will disable masking

Return Values

Error code

Description

Enable masking and allocate data space to store information for masking.
For more information on masking please refer SarRecognitionRequest::triangulateMasks_.

SarRecognizer::sarSaveSceneMap

Save scene map

Definition

```
public int32_t sarSaveSceneMap(
        SarStreamOut* stream
        )const;
```

Arguments

[in] stream	Output stream

Return Values

Error code

Description

Save current scene map to the indicated output stream.
This function is only available when using SceneMapping mode.

SarRecognizer::sarFixSceneMap

Fix/unfix the scene map

Definition

```
public int32_t sarFixSceneMap(
        bool isFix
        );
```

Arguments

[in] isFix	Flag indicates fix or unfix

Return Values

Error code

Description

Stop searching new landmarks and fix the scene map.
To resolve fixed scene map set false to isFix and call sarFixSceneMap().
This function is only available when using SceneMapping mode.

SarRecognizer::sarForceLocalize

Transit state to localization forcefully

Definition

public int32 sarForceLocalize();

Return Values

Error code

Description

Forcefully transit recognition state of SarRecognizer to localization.
This function is only available when using SceneMapping mode.

SarRecognizer::sarRemoveLandmark

Remove the indicated landmark

Definition

```
public int32_t sarRemoveLandmark(
        const SarLandmark& landmark
        );
```

Return Values

[in] landmark SarLandmark ID

Return Values

Error code

Description

This function deletes the landmark specified in id.
Note:
SmartAR™ SDK deletes landmark that has bad effect on estimations automatically.
This function is only available when using SceneMapping mode.

SarRecognizer::sarSetDenseMapMode

Set dense map mode

Definition

```
public int32_t sarSetDenseMapMode(
        SarDenseMapMode mode
        );
```

Arguments

[in] mode	Dense map mode

Return Values

Error code

Description

Set dense map mode.

SarTarget

Recognition target class

Definition

```
#include <SarRecognizer.h>
class SarTarget : SarNonCopyable {
public:
        virtual ~SarTarget() {}
        virtual int32_t sarGetPhysicalSize(SarVector2* size) const=0;
};
```

Description

This is the recognition target interface.

Related Items

SarLearnedImageTarget class
SarCompoundTarget class
SarSceneMapTarget class

SarTarget::~SarTarget

Destructor of target class

Definition

public ~SarTarget();

Description

This is the destructor of target class.

SarTarget::sarGetPhysicalSize

Retrieve physical size of the recognition target

Definition

```
public virtual int32_t sarGetPhysicalSize(
        SarVector2* size
        ) const=0;
```

Arguments

[out] size	Physical size of the recognition target[m]

Return Values

Error code

Description

Retrieve physical size of the recognition target.

SarLearnedImageTarget

Recognition target class contains information of natural image

Definition

```
#include <SarRecognizer.h>
class SarLearnedImageTarget : public SarTarget {
public:
        SarLearnedImageTarget(SarSmart*, SarStreamIn*, unsigned char *, unsigned char *);
        virtual ~SarLearnedImageTarget();
        bool sarIsConstructorFailed();

        virtual int32_t sarGetPhysicalSize(SarVector2* size) const;
};
```

Description

Recognition target contains information of natural image, created by dictionary tool.

Related Items

SarTarget class

SarLearnedImageTarget::SarLearnedImageTarget

Constructor

Definition

```
public SarLearnedImageTarget(
        SarSmart* smart,
        SarStreamIn* stream,
        unsigned char* customerID = NULL,
        unsigned char* customerKey = NULL
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class
[in] stream	Stream to dictionary file
[in] customerID	Customer ID
[in] customerKey	Customer Key

Description

Read dictionary file created by dictionary tool and initialize. If customerID and customerKey have been set when creating dictionary, the same value is required as input to this function.

SarLearnedImageTarget::~SarLearnedImageTarget

Destructor

Definition

public ~SarLearnedImageTarget();

Description

This is the destructor of SarLearnedImageTarget class.

SarLearnedImageTarget::sarIsConstructorFailed

Check error of constructor

Definition

public bool sarIsConstructorFailed();

Return Values

When error occurs in constructor true, otherwise false.

Description

Check if error occurred in constructor.

SarLearnedImageTarget::sarGetPhysicalSize

Get physical size of recognition target

Definition

```
public virtual int32_t sarGetPhysicalSize(
        SarVector2* size
        ) const=0;
```

Arguments

[out] size	Physical size of recognition target[m]

Return Values

Error code

Description

Retrieve physical size of recognition target.

SarCompoundTarget

Recognition target class consists of multiple recognition targets

Definition

```
#include <SarRecognizer.h>
class SarCompoundTarget : public SarTarget {
public:
        SarCompoundTarget(SarSmart*, const SarTarget* const*,
                          const SarChildTargetInfo*, int32_t);
        ~SarCompoundTarget();
        bool sarIsConstructorFailed();

        virtual int32_t sarGetPhysicalSize(SarVector2*) const;
};
```

Description

Recognition target class consists of multiple recognition targets.

Related Items

SarTarget class

SarCompoundTarget::SarCompoundTarget

Constructor

Definition

```
public SarCompoundTarget(
        SarSmart* smart,
        const SarTarget* const* childTargets,
        const SarChildTargetInfo* childTargetInfos,
        int32_t numChildTargets
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class
[in] childTargets	Pointer to array contains recognition targets to be set as components
[in] childTargetInfos	Information of recognition targets
[in] numChildTargets	Element number of childTargets

Description

This is the constructor of SarCompoundTarget class.

SarCompoundTarget::~SarCompoundTarget

Destructor

Definition

public ~SarCompoundTarget();

Description

This is the destructor of SarCompoundTarget class.

SarCompoundTarget::sarIsConstructorFailed

Check error of constructor

Definition

public bool sarIsConstructorFailed();

Return Values

Return true if any error occurred in constructor, otherwise false.

Description

Check if any error occurred in constructor.

SarCompoundTarget::sarGetPhysicalSize

Retrieve physical size of recognition target

Definition

```
public virtual int32_t sarGetPhysicalSize(
        SarVector2* size
        ) const=0;
```

Arguments

[out] size	Physical size of the recognition target[m]

Return Values

Error code

Description

Retrieve the physical size of the recognition target.
Note:
This function returns 0 to both width and height for the current version.

SarSceneMapTarget

Recognition target with scene map information

Definition

```
#include <SarRecognizer.h>
class SarSceneMapTarget : public SarTarget {
public:
        SarSceneMapTarget(SarSmart* smart, SarStreamIn* stream);
        ~SarSceneMapTarget();
         bool sarIsConstructorFailed();

        virtual int32_t sarGetPhysicalSize(SarVector2* size)const;
};
```

Description

Recognition target class with scene map information.
This target can only be used when SarRecognizer is running as SceneMapping mode.

Related Items

SarTarget class

SarSceneMapTarget::SarSceneMapTarget

Constructor

Definition

```
public SarSceneMapTarget(
        SarSmart* smart,
        SarStreamIn* stream
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class
[in] stream	Input stream

Description

Read scene map file from input stream and initialize.

SarSceneMapTarget::~SarSceneMapTarget

Destructor

Definition

public ~SarSceneMapTarget();

Description

This is the destructor of SarSceneMapTarget class.

SarSceneMapTarget::sarIsConstructorFailed

Check constructor error

Definition

public int32_t sarIsConstructorFailed();

Return Values

Return true when any error occurred in constructor, otherwise false.

Description

Check if any error occurred in constructor.

SarSceneMapTarget::sarGetPhysicalSize

Retrieve physical size of recognition target.

Definition

```
public virtual int32_t sarGetPhysicalSize(
        SarVector2* size
        ) const=0;
```

Arguments

[out] size	Physical size of recognition target[m]

Return Values

Error code

Description

Retrieve physical size of the recognition target.
This function returns 0 to both width and height for the current version.

SarWorkDispatchedListener

Listener to receive execute request to sarRunWorker()

Definition

```
#include <SarRecognizer.h>
class SarWorkDispatchedListener {
public:
        virtual ~WorkdispatchedListener();
        virtual void sarOnWorkDispatched() = 0;
};
```

Description

Listener to receive execute request to sarRunWorker().

SarWorkDispatchedListener::~WorkdispatchedListener

Destructor

Definition

public ~WorkdispatchedListener();

Description

This is the destructor of WorkdispatchedListener class.

SarWorkDispatchedListener::sarOnWorkDispatched

Receive execute request to sarRunWorker()

Definition

public virtual void sarOnWorkDispatched() = 0;

Description

Receive execute request to sarRunWorker().

SarRecognitionResultHolder

Class to hold recognition result

Definition

```
#include <SarRecognizer.h>
class SarRecognitionResultHolder : SarNonCopyable {
public:
        virtual int32_t sarGetNumResults() const
        virtual int32_t sarGetResults(SarRecognitionResult*, int32_t) const;
        virtual int32_t sarGetResult(const SarTarget*, SarRecognitionResult*) const;
};
```

Description

This is the class to hold recognition result.

SarRecognitionResultHolder::~SarRecognitionResultHolder

Destructor

Definition

public virtual ~SarRecognitionResultHolder();

Description

This is the destructor of SarRecognitionResultHolder class.

SarRecognitionResultHolder::sarGetNumResults

Retrieve recognition result

Definition

public virtual int32_t sarGetNumResults() const = 0;

Return Values

Number of recognition results.

Description

Retrieve recognition results stored in SarRecognitionResultHolder.

SarRecognitionResultHolder::sarGetResults

Retrieve multiple recognition results

Definition

```
public virtual int32_t sarGetResults(
        SarRecognitionResult* results,
        Int32_t maxResults
        ) const = 0;
```

Arguments

[out] results	Pointer to array of SarRecognitionResult type which holds recognition result
[in] maxResults	Maximum number of recognition result to retrieve

Return Values

Number of results is returned if finished normally, otherwise error code.

Description

Retrieve multiple recognition results.

SarRecognitionResultHolder::sarGetResult

Retrieve specific recognition result

Definition

```
public virtual int32_t sarGetResult(
        const SarTarget* target,
        SarRecognitionResult* result
        ) const = 0;
```

Arguments

[in] target	Recognition target associated with the recognition result to retrieve
[out] result	Pointer to structure of SarRecognitionResult type to contain the retrieved result

Return Values

Error code

Description

Retrieve recognition result of specific recognition target.

SarRecognizedListener

Listener to receive recognition result update

Definition

```
#include <SarRecognizer.h>
class SarRecognizedListener {
public:
        virtual ~SarRecognizedListener();
        virtual void sarOnRecognized(const SarRecognitionResultHolder&) = 0;
};
```

Description

Listener to receive recognition result update.

SarRecognizedListener::~SarRecognizedListener

Destructor

Definition

public virtual ~SarRecognizedListener();

Description

This is the destructor of SarRecognizedListener class.

SarRecognizedListener::sarOnRecognized

Receive recognition result update

Definition

```
public void sarOnRecognized(
        const SarRecognitionResultHolder& resultHolder
        )=0;
```

Arguments

[in] results	SarRecognitionResultHolder holding recognition target

Description

Receive recognition result update.

SarRecognitionRequest

Input date to recognition process

Definition

```
#include <SarRecognizer.h>
struct SarRecognitionRequest {
        SarImage image_;
        int64_t timestamp_;
        int32_t numSensorStates_;
        SarSensorState* sensorStates_;

        int32_t numTriangulateMasks_;
        const SarTriangle2* triangulateMasks_;

        SarRecognitionRequest();
};
```

Members

*image_*	Camera image which search of recognition target is performed. Format can be used for current version are SAR_IMAGE_FORMAT_L8, SAR_IMAGE_FORMAT_YCRCB420 and SAR_IMAGE_FORMAT_YCBCR420. Stride must be the same with the width of the image.
*timestamp_*	Time stamp of the camera image
*numSensorStates_*	Number of elements in sensorStates_
*sensorStates_*	Pointer to array contains sensor information.
*numTriangulateMasks_*	Number of elements in triangulateMasks_
*triangulateMasks_*	Array of triangle patch to be used as masking information. The indicated region is masked and will not be used in recognition process. The coordinate used in triangle patch is based on the assumption that both width and height of the image are 1.0. This member is only valid for SceneMapping mode.

Description

This is the structure to hold information needed for recognition process.

The structure holds camera image, time stamp and sensor information.

When using mask the array of triangle patch need to be presented.

SarLandmark

Information of landmark

Definition

```
#include <SarRecognizer.h>
struct SarLandmark{
        uint32_t id_;
        SarLandmarkState state_;
        SarVector3 position_;
};
```

Members

*id_*	Unique identification ID of landmark
*state_*	State of landmark
*position_*	Coordinate of landmark

Description

This is the structure to hold information of the landmark.

SarLandmark is a point in the scene map which 3D position is estimated.

SarInitPoint

Information of initialization point

Definition

```
#include <SarRecognizer.h>
struct SarInitPoint {
        uint32_t id_;
        SarVector2 position_;
};
```

Members

*id_*	Unique identification number of initialization point
*position_*	Coordinate of initialization point

Description

This is the structure to hold information of initialization point.
Initialization point is point tracked in 2D image that is used in initialization of SAR_SCENE_MAPPING_INIT_MODE_SFM, SAR_SCENE_MAPPING_INIT_MODE_HFG and SAR_SCENE_MAPPING_INIT_MODE_VFG mode.

SarRecognitionResult

Result of recognition process

Definition

```
#include <SarRecognizer.h>
struct SarRecognitionResult {

        const SarTarget* target_;
        bool isRecognized_;
        SarVector3 position_;
        SarQuaternion rotation_;

        uint64_t timestamp_;

        SarVector3 velocity_;
        SarVector3 angularVelocity_;

        SarTargetTrackingState targetTrackingState_;

        SarSceneMappingState sceneMappingState_;

        int32_t numLandmarks_;
        int32_t maxLandmarks_;
        SarLandmark* landmarks_;

        int32_t numInitPoints_;
        int32_t maxInitPoints_;
        SarInitPoint* initPoints_;

        SarRecognitionResult();
};
```

Members

*target_*	Pointer to recognition target associated to the recognition result stored in this structure
*isRecognized_*	Flag showing if the target is in Tracking
*position_*	Position of the device
*rotation_*	SarRotation of the device
*timestamp_*	Timestamp of input image used in recognition
*velocity_*	Speed of the device
*angularVelocity_*	Angular velocity of the device
*targetTrackingState_*	Current state of TargetTracking mode
*sceneMappingState_*	Current state of SceneMapping mode
*numLandmarks_*	Number of landmark stored in landmarks_
*maxLandmarks_*	Maximum number of landmark can be stored in landmarks_
*landmarks_*	Pointer to array to store landmarks
*numInitPoints_*	Number of initialization point stored in initPoint_
*maxInitPoints_*	Maximum number of initialization point can be stored in initPoint_
*initPoints_*	Pointer to array to store initialization points

Description

This is the structure to hold recognition result of specific recognition target.

SarChildTargetInfo

Information of recognition target

Definition

```
#include <SarRecognizer.h>
struct SarChildTargetInfo{
        SarVector3 position_;
        SarQuaternion rotation_;
};
```

Members

*position_*	Position information of recognition target
*rotation_*	SarRotation information of recognition target

Description

This is the structure to hold information of component recognition targets of SarCompoundTarget.
This structure is used to create instance of SarCompoundTarget class.

Related Items

SarCompoundTarget class

SarCameraDevice

Constants

List of Definitions

Definition	Value	Description
SAR_INVALID_CAMERA_ID	-1	Invalid CameraID

Enumerators

SarFocusMode

Focus mode

Definition

```
#include <SarCameraDevice.h>
enum SarFocusMode {
        SAR_FOCUS_MODE_MANUAL,
        SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE,
        SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO,
        SAR_FOCUS_MODE_EDOF,
        SAR_FOCUS_MODE_FIXED,
        SAR_FOCUS_MODE_INFINITY,
        SAR_FOCUS_MODE_MACRO,
};
```

Enumeration Values

Value	Description
SAR_FOCUS_MODE_MANUAL	Manual mode
SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE	Continuous auto focus mode. Focus adjustment is more frequent than SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO.
SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO	Continuous auto focus mode.
SAR_FOCUS_MODE_EDOF	Expanded depth of field mode
SAR_FOCUS_MODE_FIXED	Fixed focus mode
SAR_FOCUS_MODE_INFINITY	Infinite mode
SAR_FOCUS_MODE_MACRO	Macro mode

Description

This is the enumeration value represents the focus mode of camera.
SAR_FOCUS_MODE_MANUAL is the mode focus once when SarCameraDevice::sarRunAutoFocus() is called. SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE and SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO are modes of continuous auto focus.

SarFlashMode

Flash mode

Defintion

```
#include <SarCameraDevice.h>
enum SarFlashMode {
        SAR_FLASH_MODE_AUTO,
        SAR_FLASH_MODE_OFF,
        SAR_FLASH_MODE_ON,
        SAR_FLASH_MODE_RED_EYE,
        SAR_FLASH_MODE_TORCH,
};
```

Enumeration Values

Value	Description
SAR_FLASH_MODE_AUTO	Automaticaly decide to use flash or not
SAR_FLASH_MODE_OFF	Do not use flash
SAR_FLASH_MODE_ON	Use flash
SAR_FLASH_MODE_RED_EYE	Flash to reduce red eye
SAR_FLASH_MODE_TORCH	Always use flash

Description

This is the enumeration value to represent the flash mode of camera.

SarExposureMode

Exposure mode

Definition

```
#include <SarCameraDevice.h>
enum SarExposureMode{
        SAR_EXPOSURE_MODE_MANUAL,
        SAR_EXPOSURE_MODE_CONTINUOUS_AUTO,
};
```

Enumeration Values

名前	解説
SAR_EXPOSURE_MODE_MANUAL	Manual mode
SAR_EXPOSURE_MODE_CONTINUOUS_AUTO	Automatic mode

Description

This is the enumeration value to represent the exposure mode of camera.
When SAR_EXPOSURE_MODE_MANUAL is set, only do exposure adjustment once when SarCameraDevice::sarRunAutoExposure() is called. When SAR_EXPOSURE_MODE_CONTINUOUS_AUTO is set exposure adjustment is done continuously.

SarWhiteBalanceMode

White balance mode

Definition

```
#include <SarCameraDevice.h>
enum SarWhiteBalanceMode{
        SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO,
        SAR_WHITE_BALANCE_MODE_CLOUDY_DAYLIGHT,
        SAR_WHITE_BALANCE_MODE_DAYLIGHT,
        SAR_WHITE_BALANCE_MODE_FLUORESCENT,
        SAR_WHITE_BALANCE_MODE_INCANDESCENT,
        SAR_WHITE_BALANCE_MODE_SHADE,
        SAR_WHITE_BALANCE_MODE_TWILIGHT,
        SAR_WHITE_BALANCE_MODE_WARM_FLUORESCENT,
        SAR_WHITE_BALANCE_MODE_MANUAL,
};
```

Enumeration Values

Value	Description
SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO	Automatic mode
SAR_WHITE_BALANCE_MODE_CLOUDY_DAYLIGHT	Cloudy mode
SAR_WHITE_BALANCE_MODE_DAYLIGHT	Daylight mode
SAR_WHITE_BALANCE_MODE_FLUORESCENT	Fluorescent light mode
SAR_WHITE_BALANCE_MODE_INCANDESCENT	Incandescent light mode
SAR_WHITE_BALANCE_MODE_SHADE	Shade mode
SAR_WHITE_BALANCE_MODE_TWILIGHT	Twilight mode
SAR_WHITE_BALANCE_MODE_WARM_FLUORESCENT	Warm colored fluorescent light mode
SAR_WHITE_BALANCE_MODE_MANUAL	Manual mode

Description

This is the enumeration value to represent white balance mode of camera.
When SAR_WHITE_BALANCE_MODE_MANUAL is set only do white balance adjustment once when SarCameraDevice::sarRunAutoWhiteBalance() is called. When SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO is set white balance adjustment is done continuously.

SarSceneMode

Scene mode

Definition

```
#include <SarCameraDevice.h>
enum SarSceneMode{
        SAR_SCENE_MODE_ACTION,
        SAR_SCENE_MODE_AUTO,
        SAR_SCENE_MODE_BARCODE,
        SAR_SCENE_MODE_BEACH,
        SAR_SCENE_MODE_CANDLELIGHT,
        SAR_SCENE_MODE_FIREWORKS,
        SAR_SCENE_MODE_LANDSCAPE,
        SAR_SCENE_MODE_NIGHT,
        SAR_SCENE_MODE_NIGHT_PORTRAIT,
        SAR_SCENE_MODE_PARTY,
        SAR_SCENE_MODE_PORTRAIT,
        SAR_SCENE_MODE_SNOW,
        SAR_SCENE_MODE_SPORTS,
        SAR_SCENE_MODE_STEADYPHOTO,
        SAR_SCENE_MODE_SUNSET,
        SAR_SCENE_MODE_THEATRE,
};
```

Enumeration Values

Value	Description
SAR_SCENE_MODE_ACTION	Action
SAR_SCENE_MODE_AUTO	Auto
SAR_SCENE_MODE_BARCODE	Barcode
SAR_SCENE_MODE_BEACH	Beach
SAR_SCENE_MODE_CANDLELIGHT	Candle light
SAR_SCENE_MODE_FIREWORKS	Fireworks
SAR_SCENE_MODE_LANDSCAPE	Landscape
SAR_SCENE_MODE_NIGHT	Night
SAR_SCENE_MODE_NIGHT_PORTRAIT	Night portrait
SAR_SCENE_MODE_PARTY	Party
SAR_SCENE_MODE_PORTRAIT	Portrait
SAR_SCENE_MODE_SNOW	Snow
SAR_SCENE_MODE_SPORTS	Sports
SAR_SCENE_MODE_STEADYPHOTO	Hand shake reduced mode
SAR_SCENE_MODE_SUNSET	Sunset
SAR_SCENE_MODE_THEATRE	Theater

Description

This is the enumeration value to represent the scene mode of camera.

Classes

SarCameraDevice

Camera device

Definition

```
#include <SarCameraDevice.h>
class SarCameraDevice : SarNonCopyable {
public:
        static const int SAR_INVALID_CAMERA_ID = -1;

        SarCameraDevice(SarSmart* smart);
        SarCameraDevice(SarSmart* smart, int32_t cameraId, void* nativeDevice = NULL);
        ~SarCameraDevice();
        bool sarIsConstructorFailed();

        // setting
        int32_t sarSetNativeVideoOutput(void* nativeVideoOutput);
        int32_t sarSetVideoImageListener(SarCameraImageListener* listener);
        int32_t sarSetVideoImageSize(int32_t width, int32_t height);
        int32_t sarSetVideoImageFormat(SarImageFormat format);
        int32_t sarSetVideoImageFpsRange(float min, float max);
        int32_t sarSetStillImageListener(SarCameraImageListener* listener);
        int32_t sarSetStillImageSize(int32_t width, int32_t height);
        int32_t sarSetStillImageFormat(SarImageFormat format);
        int32_t sarSetShutterListener(SarCameraShutterListener* listener);
        int32_t sarSetFocusMode(SarFocusMode mode);
        int32_t sarSetFocusAreas(SarCameraArea* areas, int32_t numAreas);
        int32_t sarSetExposureMode(SarExposureMode mode);
        int32_t sarSetExposureAreas(SarCameraArea* areas, int32_t numAreas);
        int32_t sarSetFlashMode(SarFlashMode mode);
        int32_t sarSetWhiteBalanceMode(SarWhiteBalanceMode mode);
        int32_t sarSetSceneMode(SarSceneMode mode);
        int32_t sarSetAutoFocusListener(SarCameraAutoAdjustListener* listener);
        int32_t sarSetAutoExposureListener(SarCameraAutoAdjustListener* listener);
        int32_t sarSetAutoWhiteBalanceListener(SarCameraAutoAdjustListener* listener);
        int32_t sarSetErrorListener(SarCameraErrorListener);
        int32_t sarSetOwningNativeDevice(bool isOwning);

        // get info
        static int32_t sarGetDefaultCameraId(SarSmart* smart, SarFacing facing,
            int32_t* cameraId);
        int32_t sarGetSupportedVideoImageSize(SarSize* sizes, int32_t maxSizes) const;
        int32_t sarGetSupportedVideoImageFormat(SarImageFormat* formats,
            int32_t maxFormats) const;
        int32_t sarGetSupportedVideoImageFpsRange(SarCameraFpsRange* ranges,
            int32_t maxRanges) const;
        int32_t sarGetSupportedStillImageSize(SarSize* sizes, int32_t maxSizes) const;
        int32_t sarGetSupportedStillSarImageFormat(SarImageFormat* formats,
            int32_t maxFormats) const;
        int32_t sarGetSupportedFocusMode(SarFocusMode* modes, int32_t maxModes) const;
        int32_t sarGetMaxNumFocusAreas() const;
```

```
        int32_t sarGetSupportedFlashMode(SarFlashMode* modes, int32_t maxModes) const;
        int32_t sarGetSupportedExposureMode(SarExposureMode* modes,
            int32_t maxModes) const;
        int32_t sarGetMaxNumExposureAreas() const;
        int32_t sarGetSupportedWhiteBalanceMode(SarWhiteBalanceMode* modes,
            int32_t maxModes) const;
        int32_t sarGetSupportedSceneMode(SarSceneMode* modes, int32_t maxModes) const;

        int32_t sarGetDeviceInfo(SarCameraDeviceInfo* info) const;
        int32_t sarGetDeviceInfo(SarCameraDeviceInfo* info, int32_t scaledWidth,
            int32_t scaledHeight, bool isStillImage = false) const;
        int32_t sarGetFovY(float* fovy, float heightRatio = 1.0f,
            bool* calibrated = NULL) const;
        int32_t sarGetFovY(float* fovy, float heightRatio,
            bool* calibrated, bool isStillImage) const;
        int32_t sarGetFovY(float* fovy, int targetWidth, int targetHeight,
            bool* getfromapi = NULL, bool* calibrated = NULL) const;

        int32_t sarGetFacing(SarFacing* facing) const;
        int32_t sarGetRotation(SarRotation* rotation) const;
        int32_t sarGetNativeDevice(void** nativeDevice) const;

        // start and stop
        int32_t sarStart();
        int32_t sarStop();

        // misc
        int32_t sarCaptureStillImage();
        int32_t sarRunAutoFocus();
        int32_t sarRunAutoExposuresarRunAutoExposure();
        int32_t sarRunAutoWhiteBalance();
};
```

Description

SarCameraDevice class is for changing camera settings, capture of image and video image. Captured image can be used as input data to SarRecognizer.

Note:
The following permission setting must be written to AndroidManifest.xml to use SarCameraDevice class in Android environment.

- <uses-permission android:name="android.permission.CAMERA" />

SarCameraDevice::SarCameraDevice

Constructor

Definition

```
public SarCameraDevice(
        SarSmart* smart
        );

pubiic SarCameraDevice(
        SarSmart* smart,
        int32_t cameraId,
        void* nativeDevice=NULL
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class
[in] cameraId	Camera ID
[in] nativeDevice	Pointer to native camera device object

Description

This is the constructor of SarCameraDevice class.
To use specific camera set the camera's ID to cameraID.
When using camera device object native to the platform set pointer to the object to nativeDevice.
This object is different across platforms.
For Android android.hardware.Camera class, for iOS AVCaptureSession class.

SarCameraDevice::sarIsConstructorFailed

Check constructor error

Definition

public bool sarIsConstructorFailed();

Return Values

When error occurred in constructor returns true, otherwise false.

Description

Check if any error occurred in constructor.

SarCameraDevice::sarSetNativeVideoOutput

Set video output

Definition

```
public sarSetNativeVideoOutput(
        void* nativeVideoOutput
        );
```

Arguments

[in] nativeVideoOutput	Pointer to native video output object

Return Values

Error code

Description

When outputting to specific output, set the output object to nativeVideoOutput.
For Android android.vie.SurfaceView class, for iOS AVCaptureOutput class.
On Android device, if video output is not set the library may not work properly.
On iOS device setting is not mandatory except specific need.

SarCameraDevice::sarSetVideoImageListener

Set listener to receive captured video image

Definition

```
public int32_t sarSetVideoImageListener(
        SarCameraImageListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarCameraImageListener class

Return Values

Error code

Description

Set listener to receive captured video image.

SarCameraDevice::sarSetVideoImageSize

Set size of video image

Definition

```
public int32_t sarSetVideoImageSize(
        int32_t width,
        int32_t height
        );
```

Arguments

[in] width	Width of image
[in] height	Height of image

Return Values

Error code

Description

Set size of video image.

SarCameraDevice::sarSetVideoImageFormat

Set format of video image

Definition

```
public int32_t sarSetVideoImageFormat(
        SarImageFormat format
        );
```

Arguments

[in] format	Format of image

Return Values

Error code

Description

Set format of video image.

SarCameraDevice::sarSetVideoImageFpsRange

Set update interval of video image.

Definition

```
public int32_t sarSetVideoImageFpsRange(
        float min,
        float max
        );
```

Arguments

[in] min	Minimum update interval
[in] max	Maximum update interval

Return Values

Error code

Description

Set update interval of video image.

SarCameraDevice::sarSetStillImageListener

Set listener to receive captured image

Definition

```
public int32_t sarSetStillImageListener(
        SarCameraImageListener*    listener
        );
```

Arguments

[in] listener	Pointer to instance of SarCameraImageListener class

Return Values

Error code

Description

Set listener to receive captured still image.

SarCameraDevice::sarSetStillImageSize

Set size of still image

Definition

```
public int32_t sarSetStillImageSize(
        int32_t width,
        int32_t height
        );
```

Arguments

[in] width	Width of image
[in] height	Height of image

Return Values

Error code

Description

Set size of still image.

SarCameraDevice::sarSetStillImageFormat

Set format of still image

Definition

```
public int32_t sarSetStillImageFormat(
        SarImageFormat format
        );
```

Arguments

[in] format	Image format

Return Values

Error code

Description

Set format of still image.

SarCameraDevice::sarSetShutterListener

Set listener to receive notice of shutter completion

Definition

```
public int32_t sarSetShutterListener(
        SarCameraShutterListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarCameraShutterListener class

Return Values

Error code

Description

Set listener to receive notice of shutter completion.

SarCameraDevice::sarSetFocusMode

Set focus mode

Definition

```
public int32_t sarSetFocusMode(
        SarFocusMode mode
        );
```

Arguments

[in] mode	Focus mode

Return Values

Error code

Description

Set focus mode of camera.

SarCameraDevice::sarSetFocusAreas

Set auto focus area

Definition

```
public int32_t sarSetFocusAreas(
        SarCameraArea* areas,
        int32_t numAreas
        );
```

Arguments

[in] areas	Pointer to array contains auto focus area
[in] numAreas	*Element number of areas*

Return Values

Error code

Description

Set auto focus area. Focus area can be canceled by setting 0 to numAreas.

SarCameraDevice::sarSetExposureMode

Set exposure mode

Definition

```
public int32_t sarSetExposureMode(
        SarExposureMode mode
        );
```

Arguments

[in] mode	Exposure mode

Return Values

Error code

Description

Set exposure mode of camera.

SarCameraDevice::sarSetExposureAreas

Set exposure area

Definition

```
public int32_t sarSetExposureAreas(
        SarCameraArea* areas,
        int32_t numAreas
        );
```

Arguments

[in] areas	Pointer to array contains exposure area
[in] numAreas	*Element number of areas*

Return Values

Error code

Description

Set exposure area.

SarCameraDevice::sarSetFlashMode

Set flash mode

Definition

```
public int32_t sarSetFlashMode(
        SarFlashMode mode
        );
```

Arguments

[in] mode	Flash mode

Return Value

Error code

Description

Set flash mode of camera.

SarCameraDevice::sarSetWhiteBalanceMode

Set white balance

Definition

```
public int32_t sarSetWhiteBalanceMode(
        SarWhiteBalanceMode mode
        );
```

Arguments

[in] mode	White balance mode

Return Values

Error code

Description

Set white balance mode of camera.

SarCameraDevice::sarSetSceneMode

Set scene mode

Definition

```
public int32_t sarSetSceneMode(
        SarSceneMode mode
        );
```

Arguments

[in] mode	Scene mode

Return Values

Error code

Description

Set scene mode of camera.

SarCameraDevice::sarSetAutoFocusListener

Set Listener to receive notice of auto focus completion

Definition

```
public int32_t sarSetAutoFocusListener(
        SarCameraAutoAdjustListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarCameraAutoAdjustListener class

Return Values

Error code

Description

Set listener to receive notice of auto focus completion from sarRunAutoFocus().

SarCameraDevice::sarSetAutoExposureListener

Set listener to receive notice of auto exposure completion

Definition

```
public int32_t sarSetAutoExposureListener(
        SarCameraAutoAdjustListener* listener
        );
```

Arguments

[in] listener Pointer to instance of SarCameraAutoAdjustListener class

Return Values

Error code

Description

Set listener to receive notice of auto exposure completion from sarRunAutoExposuresarRunAutoExposure().

SarCameraDevice::sarSetAutoWhiteBalanceListener

Set listener to receive notice of auto white balance adjustment completion

Definition

```
public int32_t sarSetAutoWhiteBalanceListener(
        SarCameraAutoAdjustListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarCameraAutoAdjustListener class

Return Values

Error code

Description

Set listener to receive notice of auto white balance adjustment completion from sarRunAutoWhiteBalance().

SarCameraDevice::sarSetErrorListener

Set listener to receive notice of auto white balance adjustment completion

Definition

```
public int32_t sarSetErrorListener(
        SarCameraErrorListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarCameraErrorListener class

Return Values

Error code

Description

Set listener to receive camera error.

SarCameraDevice::sarSetOwningNativeDevice

Set flag showing ownership of NativeDevice object

Definition

```
public int32_t sarSetOwningNativeDevice(
        bool isOwning
        );
```

Arguments

[in] isOwning Flag value showing ownership

Return Values

Error code

Description

Set flag showing ownership of NativeDevice object.
If the NativeDevice object pointer given to the constructor is null the ownership flag is set to true, otherwise false.
If isOwning is true, when destructor of SarCameraDevice class is called the NativeDevice object is released at the same time.

SarCameraDevice::sarGetDefaultCameraId

Get ID of camera on specified location

Definition

```
public static int32_t sarGetDefaultCameraId(
        SarSmart* smart,
        SarFacing facing,
        int32_t* cameraId
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class
[in] facing	Location of camera
[out] cameraId	Pointer to int32_t type variable to contain camera ID. If the camera does not exist SAR_INVALID_CAMERA_ID is returned.

Return Values

Error code

Description

Get ID of camera on specified location.
If the camera does not exist SAR_INVALID_CAMERA_ID is returned.

SarCameraDevice::sarGetSupportedVideoImageSize

Get supported video image size

Definition

```
public int32_t sarGetSupportedVideoImageSize(
        SarSize* sizes,
        int32_t maxSizes
        ) const;
```

Arguments

[out] sizes	Pointer to array to contain video image size
[in] maxSizes	Element number can be stored to sizes

Return Values

When finished normally returns number of SarSize type structure contained in sizes, otherwise error code.

Description

Get supported video image size of the device.

SarCameraDevice::sarGetSupportedVideoImageFormat

Get supported video image format

Definition

public int32_t sarGetSupportedVideoImageFormat(
SarImageFormat* formats,
int32_t maxFormats
) const;

Arguments

[out] formats	Pointer to array contains video image format
[in] maxFormats	Element number can be stored to formats

Return Values

When finished normally returns the number of SarImageFormat type elements contained in formats, otherwise error code.

Description

Get supported video image format of the device.

SarCameraDevice::sarGetSupportedVideoImageFpsRange

Get supported video image update interval

Definition

public int32_t sarGetSupportedVideoImageFpsRange(
SarCameraFpsRange* ranges,
int32_t maxRanges
) const;

Arguments

[out] ranges	Pointer to array to contain video image update interval
[in] maxRanges	Element number can be stored to ranges

Return Values

When finished normally returns the number of SarCameraFpsRange type elements contained in ranges, otherwise error code.

Description

Get supported video image update interval of the device.

SarCameraDevice::sarGetSupportedStillImageSize

Get supported still image size

Definition

```
public int32_t sarGetSupportedStillImageSize(
        SarSize* sizes,
        int32_t maxSizes
        ) const;
```

Arguments

[out] sizes	Pointer to array to contain still image size
[in] maxSizes	Element number can be stored to sizes

Return Values

When finished normally returns number of SarSize type elements contained in sizes, otherwise error code.

Description

Get supported still image size of the device.

SarCameraDevice::sarGetSupportedStillImageFormat

Get supported still image format

Definition

```
public int32_t sarGetSupportedStillImageFormat(
        SarImageFormat* formats,
        int32_t maxFormats
        ) const;
```

Arguments

[out] formats	Pointer to array to contain still image format
[in] maxFormats	Element number can be stored to formats

Return Values

When finished normally returns number of SarImageFormat type elements stored in formats, otherwise error code.

Description

Get supported still image format of the device.

SarCameraDevice::sarGetSupportedFocusMode

Get supported focus mode

Definition

```
public int32_t sarGetSupportedFocusMode(
        SarFocusMode* modes,
        int32_t maxModes
        ) const;
```

Arguments

[out] modes	Pointer to array to contain focus mode
[in] maxModes	Element number can be stored to modes

Return Values

When finished normally returns number of SarFocusMode type elements stored in modes, otherwise error code.

Description

Get supported focus mode.

SarCameraDevice::sarGetMaxNumFocusAreas

Get supported maximum number of focus area

Definition

public int32_t sarGetMaxNumFocusAreas() const;

Return Values

When finished normally returns maximum number of areas, otherwise error code.

Description

Get supported maximum number of focus area of the device.

SarCameraDevice::sarGetSupportedFlashMode

Get supported flash mode

Definition

```
public int32_t sarGetSupportedFlashMode(
        SarFlashMode* modes,
        int32_t maxModes
        ) const;
```

Arguments

[out] modes	Pointer to array to contain flash modes
[in] maxSizes	Element number can be stored to modes

Return Values

When finished normally returns number of SarFlashMode type elements stored to modes, otherwise error code.

Description

Get supported flash mode of the device.

SarCameraDevice::sarGetSupportedExposureMode

Get supported exposure mode

Definition

```
public int32_t sarGetSupportedExposureMode(
        SarExposureMode* modes,
        int32_t maxModes
        ) const;
```

Arguments

[out] modes	Pointer to array to contain exposure mode
[in] maxSizes	Element number can be stored to modes

Return Values

When finished normally returns the number of SarExposureMode type elements stored to modes, otherwise error code.

Description

Get supported exposure mode of the device.

SarCameraDevice::sarGetMaxNumExposureAreas

Get supported maximum number of exposure adjustment area

Definition

public int32_t sarGetMaxNumExposureAreas() const;

Return Values

When finished normally returns maximum number of exposure adjustment areas, otherwise error code.

Description

Get supported maximum number of exposure adjustment area.

SarCameraDevice::sarGetSupportedWhiteBalanceMode

Get supported white balance mode

Definition

```
public int32_t sarGetSupportedWhiteBalanceMode(
        SarWhiteBalanceMode* modes,
        int32_t maxModes
        ) const;
```

Arguments

[out] modes	Pointer to array to contain white balance mode
[in] maxModes	Number of SarWhiteBalanceMode type elements can be contained to modes

Return Values

When finished normally returns number of SarWhiteBalanceMode type elements contained to modes, otherwise error code.

Description

Get supported white balance mode.

SarCameraDevice::sarGetSupportedSceneMode

Get supported scene mode

Definition

```
public int32_t sarGetSupportedSceneMode(
        SarSceneMode* modes,
        int32_t maxModes
        ) const;
```

Arguments

[out] modes	Pointer to array to contain scene mode
[in] maxModes	Number of SarSceneMode type elements can be contained to modes.

Return Values

When finished normally returns number of SarSceneMode type elements contained to modes, otherwise error code.

Description

Get supported scene mode.

SarCameraDevice::sarGetDeviceInfo

Get camera device information

Definition

```
public int32_t sarGetDeviceInfo(
        SarCameraDeviceInfo* info
        ) const;
```

Arguments

[out] info	Information of camera device

Return Values

Error code

Description

Get camera device information needed by SarRecognizer.

SarCameraDevice::sarGetDeviceInfo

Get camera device information

Definition

```
public int32_t sarGetDeviceInfo(
        SarCameraDeviceInfo* info
        int32_t scaledWidth,
        int32_t scaledHeight,
        bool isStillImage = false
) const;
```

Arguments

[out] info	Information of camera device
[in] scaledWidth	Convert internal parameter of camera in SmartAR by this value If -1 is set, current size is used.
[in] scaledHeight	Convert internal parameter of camera in SmartAR by this value If -1 is set, current size is used.
[in] isStillImage	Choose mode of internal parameter of camera in SmartAR true : for still image, false : for video image

Return Values

Error code

Description

Get camera device information needed by SarRecognizer.

SarCameraDevice::sarGetFovY

Get vertical field of view on screen

Definition

```
public int32_t sarGetFovY(
        float* fovy,
        int heightRatio = 1.0f
        bool* calibrated = NULL
) const;
```

Arguments

[out] fovy	Pointer to float type variable to contain field of view
[in] heightRatio	Set the ratio of cut height and original height here if height-cut camera image is needed, such as 4:3 to 16:9.
[out] calibrated	True if used internal parameter of camera in SmartAR, otherwise false.

Return Values

Error code

Description

Get vertical field of view on screen.

SarCameraDevice::sarGetFovY

Get vertical field of view on screen

Definition

```
public int32_t sarGetFovY(
        float* fovy,
        float heightRatio,
        bool* calibrated,
        bool isStillImage
) const;
```

Arguments

[out] fovy	Pointer to float type variable to contain field of view
[in] heightRatio	Set the ratio of cut height and original height here if height-cut camera image is needed, such as 4:3 to 16:9.
[out] calibrated	True if used internal parameter of camera in SmartAR, otherwise false.
[in] isStillImage	True if use internal parameter of camera for still image. False is for video image.

Return Values

Error code

Description

Get vertical field of view on screen.

SarCameraDevice::sarGetFovY

Get vertical field of view on screen

Definition

```
public int32_t sarGetFovY(
        float* fovy,
        int targetWidth,
        int targetHeight,
        bool* getfromapi = NULL,
        bool* calibrated = NULL
) const;
```

Arguments

[out] fovy	Pointer to float type variable to contain field of view
[in] targetWidth	Screen width
[in] targetHeight	Screen height
[out] getfromapi	True if succeeded to get from Android API, otherwise false.
[out] calibrated	True if used internal parameter of camera in SmartAR, otherwise false.

Return Values

Error code

Description

Get vertical field of view on screen.
Use value from Android API in Xperia™ Z1 and after models made by Sony Mobile Communications Inc.
In other Android devices use sarGetFovY(fovy, 1.0f, calibrated, false) function.
If not Android, return SAR_ERROR_INVALID_VALUE always.

SarCameraDevice::sarGetFacing

Get facing of camera

Definition

```
public int32_t sarGetFacing(
        SarFacing* facing
        ) const;
```

Arguments

[out] facing	Pointer to SarFacing type variable to contain camera facing

Return Values

Error code

Description

Get facing of camera used by SarCameraDevice.

SarCameraDevice::sarGetRotation

Get rotation angle of camera

Definition

public int32_t sarGetRotation(
SarRotation* rotation,
) const;

Arguments

[out] rotation	Pointer to SarRotation type variable to contain rotation angle

Return Values

Error code

Description

Get rotation angle of camera to device (clockwise).

SarCameraDevice::sarGetNativeDevice

Get pointer to NativeDevice type object

Definition

```
public int32_t sarGetNativeDevice(
        void** nativeDevice,
        ) const;
```

Arguments

[out] nativeDevice	Pointer to contain pointer to NativeDevice type object

Return Values

Error code

Description

Get pointer to NativeDevice type object.

SarCameraDevice::sarStart

Start video image capture

Definition

public int32_t sarStart();

Return Values

Error code

Description

Start video image capture.

SarCameraDevice::sarStop

Stop video image capture

Definition

public int32_t sarStop();

Return Values

Error code

Description

Stop video image capture.

SarCameraDevice::sarCaptureStillImage

Take still image

Definition

public int32_t sarCaptureStillImage();

Return Values

Error code

Description

Take still image.

SarCameraDevice::sarRunAutoFocus

Start auto focus

Definition

public int32_t sarRunAutoFocus();

Return Values

Error code

Description

Start auto focus.
SAR_FOCUS_MODE_MANUAL is needed to be set to use this method.
When SAR_FOCUS_MODE_CONTINUOUS_AUTO_PICTURE or SAR_FOCUS_MODE_CONTINUOUS_AUTO_VIDEO is set auto focus is automatically done by the camera and this method is not needed.

SarCameraDevice::sarRunAutoExposuresarRunAutoExposure

Start exposure adjustment

Definition

public int32_t sarRunAutoExposuresarRunAutoExposure();

Return Values

Error code

Description

Start exposure adjustment.
To use this method SAR_EXPOSURE_MODE_MANUAL is needed to be set.
When SAR_EXPOSURE_MODE_CONTINUOUS_AUTO is set exposure is automatically adjusted by camera and this method is not needed.

SarCameraDevice::sarRunAutoWhiteBalance

Start white balance adjustment

Definition

public int32_t sarRunAutoWhiteBalance();

Return Values

Error code

Description

Start white balance adjustment.
To use this method SAR_WHITE_BALANCE_MODE_MANUAL is needed to be set.
When SAR_WHITE_BALANCE_MODE_CONTINUOUS_AUTO is set white balance is automatically adjusted by camera and this method is not needed.

SarCameraDeviceInfo

Camera device information

Definition

```
#include <SarCameraDevice.h>
class SarCameraDeviceInfo : SarNonCopyable {
public:
        SarCameraDeviceInfo();
        ~SarCameraDeviceInfo();

        SarCameraDeviceInfo(const SarCameraDeviceInfo& rhs);
        SarCameraDeviceInfo& operator=(const SarCameraDeviceInfo& rhs);
};
```

Description

This is the class to hold the camera device information.

SarImageHolder

Image data

Definition

```
#include <SarCameraDevice.h>
class SarImageHolder : SarNonCopyable {
public:
        SarImageHolder();
        int32_t sarGetImageSizeInBytes() const = 0;
        int32_t sarGetImage(SarImage*, int32_t)
};
```

Description

This is the class to hold image data.

SarImageHolder::~SarImageHolder

Destructor

Definition

public ~SarImageHolder();

Description

This is the destructor of SarImageHolder class.

SarImageHolder::sarGetImageSizeInBytes

Get image size

Definition

public int32_t sarGetImageSizeInBytes() const;

Return Values

Image size as byte number

Description

Get size of image contained in SarImageHolder as byte number.

SarImageHolder::sarGetImage

Get image data

Definition

```
public int32_t sarGetImage(
        SarImage* image,
        int32_t maxSizeInBytes
        ) const;
```

Return Values

[out] image	Pointer to the instance of SarImage class to contain the image data. Note: pixel data of the instance of SarImage class need to be allocated to a size same with maxSizeInBytes or larger.
[in] maxSizeInBytes	SarSize of pixel data of the instance of SarImage class to return.

Return Values

Error code

Description

Get image data contained in SarImageHolder.

SarCameraImageListener

Listener to receive camera video image and still image

Definition

```
#include <SarCameraDevice.h>
class SarCameraImageListener {
public:
        virtual ~SarCameraImageListener();
        virtual void sarOnImage(const SarImageHolder&, uint64_t) = 0;
};
```

Description

This is the listener to receive camera video image and still image.

SarCameraImageListener::~SarCameraImageListener

Destructor

Definition

public ~SarCameraImageListener();

Description

This is the destructor of SarCameraImageListener class.

SarCameraImageListener::sarOnImage

Get camera video image and still image

Definition

```
public virtual void sarOnImage(
        const SarImageHolder& imageHolder,
        uint64_t timestamp
        ) = 0;
```

Arguments

[in] imageHolder	SarImageHolder holding image data
[in] timestamp	Time stamp of the image

Description

Get video image and still image from camera.

SarCameraShutterListener

Listener to receive notice of shutter completion

Definition

```
#include <SarCameraDevice.h>
class SarCameraShutterListener {
public:
        virtual ~SarCameraShutterListener();
        virtual void sarOnShutter() = 0;
};
```

Description

This is the listener to receive notice of shutter completion.

SarCameraShutterListener::~SarCameraShutterListener

Destructor

Definition

public virtual ~SarCameraShutterListener();

Description

This is the destructor of SarCameraShutterListener class.

SarCameraShutterListener::sarOnShutter

Receive notice of shutter completion

Definition

public virtual void sarOnShutter() = 0;

Description

Receive notice of shutter completion.

SarCameraAutoAdjustListener

Listener to receive notice of auto adjustment completion

Definition

```
#include <SarCameraDevice.h>
class SarCameraAutoAdjustListener {
public:
        virtual ~SarCameraAutoAdjustListener();
        virtual void sarOnAutoAdjust(bool success) = 0;
};
```

Description

This is the listener to receive notice of auto adjustment completion.

SarCameraAutoAdjustListener::~SarCameraAutoAdjustListener

Destructor

Definition

public virtual ~SarCameraAutoAdjustListener();

Description

This is the destructor of CamerAutoAdjustListener class.

SarCameraAutoAdjustListener::sarOnAutoAdjust

Receive notice of auto adjustment completion

Definition

```
public virtual void sarOnAutoAdjust(
        bool success
        ) = 0;
```

Arguments

[in] success	Flag represents if the auto adjustment is finished normally

Description

Receive notice of auto adjustment completion.

SarCameraErrorListener

Listener to receive camera error

Definition

```
#include <SarCameraDevice.h>
class SarCameraErrorListener {
public:
        virtual ~SarCameraErrorListener();
        virtual void sarOnError(int32_t error) = 0;
};
```

Description

This is the listener to receive camera error.

SarCameraErrorListener::~SarCameraErrorListener

Destructor

Definition

public ~SarCameraErrorListener();

Description

This is the destructor of SarCameraErrorListener class.

SarCameraErrorListener::sarOnError

Get camera error

Definition

```
public virtual void sarOnError(
        int32_t error
        ) = 0;
```

Arguments

[in] error	Error code

Description

Get camera error..

SarCameraFpsRange

Update interval of camera

Definition

```
#include <SarCameraDevice.h>
struct SarCameraFpsRange{
        float min_;
        float max_;
};
```

Members

*min_*	Minimum update interval
*max_*	Maximum update interval

Description

This is the structure to hold update interval.

SarCameraArea

Adjustment area of camera

Definition

```
#include <SarCameraDevice.h>
struct SarCameraArea {
        float left_;
        float top_;
        float right_;
        float bottom_;
        float weight_;
};
```

Members

*left_*	X coordinate of leftmost of the rectangle
*top_*	Y coordinate of top limit of the rectangle
*right_*	X coordinate of rightmost of the rectangle
*bottom_*	Y coordinate of the lower limit of the rectangle
*weight_*	Weight of the area

Description

This is the structure to hold information of adjustment area of camera.
SarCameraArea is the area for auto focus and exposure adjustment.
Set the coordinates of the rectangle to between 1.0 and -1.0.
Set weight to weight_ to differentiate importance of adjustment areas. When adjustment areas are overlapped the weights of the overlapped areas are added.
Set weight from 0.0 to 1.0.

SarSensorDevice

Enumerators

SarSensorType

Sensor type

Definition

```
#include <SarPlatformDef.h>
enum SarSensorType {
        SAR_SENSOR_TYPE_ACCELEROMETER,
        SAR_SENSOR_TYPE_GYROSCOPE,
};
```

Enumeration Values

Value	Description
SAR_SENSOR_TYPE_ACCELEROMETER	Acceleration sensor
SAR_SENSOR_TYPE_GYROSCOPE	Gyroscope

Description

This is the enumeration value to represent sensor type.

Classes

SarSensorDevice

Sensor device

Definition

```
#include <SarSensorDevice.h>
class SarSensorDevice : SarNonCopyable {
public:
        SarSensorDevice(SarSmart* smart, void* nativeDevice = NULL);
        ~SarSensorDevice();
        bool sarIsConstructorFailed();

        // setting
        int32_t sarSetSarSensorListener(SarSensorListener* listener);
        int32_t sarSetOwningNativeDevice(bool isOwning);

        // get info
        int32_t sarGetDeviceInfo(SarSensorDeviceInfo* info) const;
        int32_t sarGetNativeDevice(void** nativeDevice) const;

        // start and stop
        int32_t sarStart();
        int32_t sarStop();
};
```

Description

SarSensorDevice is the class to get value of sensors of the device.

The values can be used as input data to SarRecognizer.

SarSensorDevice::SarSensorDevice

Constructor

Definition

```
public SarSensorDevice(
        SarSmart* smart,
        void* nativeDevice = NULL
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class
[in] nativeDevice	Pointer to native sensor device object of the platform

Description

This is the constructor of SarSensorDevice class.
When using sensor device object of application as SarSensorDevice class, set pointer to native sensor device object of the platform to nativeDevice.
For Android use android.hardware.SensorManager class, for iOS use CMMotionManager class.

SarSensorDevice::~SarSensorDevice

Destructor

Definition

public ~SarSensorDevice();

Description

This is the destructor of SarSensorDevice class.

SarSensorDevice::sarIsConstructorFailed

Check constructor error

Definition

public bool sarIsConstructorFailed();

Return Values

When error occurred in constructor returns true, otherwise false.

Description

Check if any error occurred in constructor.

SarSensorDevice::sarSetSarSensorListener

Set listener to receive sensor information

Definition

```
public int32_t sarSetSarSensorListener(
        SarSensorListener* listener
        );
```

Arguments

[in] listener	Pointer to instance of SarSensorListener class

Return Values

Error code

Description

Set listener to receive sensor information.

SarSensorDevice::sarSetOwningNativeDevice

Set flag to represent ownership of NativeDevice object

Definition

```
public int32_t sarSetOwningNativeDevice(
        bool isOwning
        );
```

Arguments

[in] isOwning	Flag represents ownership

Return Values

Error code

Description

Set fag to represent ownership of NativeDevice object.
When NULL pointer of NativeDevice class is given to constructor the ownership flag is set to true, otherwise false.
If the ownership is true, the NativeDevice object is released when destructor of SarSensorDevice is called.

SarSensorDevice::sarGetDeviceInfo

Get sensor device information

Definition

```
public int32_t sarGetDeviceInfo(
        SarSensorDeviceInfo* info
        ) const;
```

Arguments

[out] info	Sensor device information

Return Values

Error code

Description

Get sensor device information for SarRecognizer to use.

SarSensorDevice::sarGetNativeDevice

Get pointer to NativeDevice object

Definition

```
public int32_t sarGetNativeDevice(
        void** nativeDevice,
        ) const;
```

Arguments

[out] nativeDevice	Pointer to contain pointer to NativeDevice object

Return Values

Error code

Description

Get pointer to NativeDevice object.

SarSensorDevice::sarStart

Start to receive value from sensor

Definition

public int32_t sarStart();

Return Values

Error code

Description

Start to receive value from sensor.

SarSensorDevice::sarStop

Stop to receive value from sensor

Definition

public int32_t sarStop();

Return Values

Error code

Description

Stop to receive value from sensor.

SarSensorDeviceInfo

Sensor device information

Definition

```
#include <SarSensorDevice.h>
class SarSensorDeviceInfo : SarNonCopyable {
public:
        SarSensorDeviceInfo();
        ~SarSensorDeviceInfo();

        SarSensorDeviceInfo(const SarSensorDeviceInfo& );
        SarSensorDeviceInfo& operator=(const SarSensorDeviceInfo& );
};
```

Description

This is the class to hold information of sensor device.

SarSensorListener

Listener to receive sensor information

Definition

```
#include <SarSensorDevice.h>
class SarSensorListener {
public:
        virtual ~SarSensorListener();
        virtual void sarOnSensor(const SarSensorState& state) = 0;
};
```

Description

This is the listener to receive sensor information.

SarSensorListener::~SarSensorListener

Destructor

Definition

public virtual ~SarSensorListener();

Description

This is the destructor of SarSensorListener class.

SarSensorListener::sarOnSensor

Receive sensor information

Definition

```
public virtual void sarOnSensor(
        const SarSensorState& state
        ) = 0;
```

Arguments

[in] state	Instance of SarSensorState to contain sensor information.

Description

Receive sensor information.

SarSensorState

Class to hold sensor information

Definition

```
#include <SarSensorDevice.h>
class SarSensorState : SarNonCopyable {
        SarSensorState();
        SarSensorState(const SarSensorState &other);
        ~SarSensorState();
        SarSensorState & operator = (const SarSensorState &other);
};
```

Description

This is the class to hold sensor information.

SarScreenDevice

Classes

SarScreenDevice

Screen device

Definition

```
#include <SarScreenDevice.h>
class SarScreenDevice : SarNonCopyable {
public:
        SarScreenDevice (SarSmart* smart);
        ~ SarScreenDevice ();
        int32_t sarIsConstructorFailed();

        int32_t sarGetRotation(SarRotation* rotation);
};
```

Description

This is the abstract class of screen device of the platform.

SarScreenDevice:: SarScreenDevice

Constructor

Definition

```
public SarScreenDevice (
        SarSmart* smart
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class

Description

This is the constructor of SarScreenDevice class.

SarScreenDevice::~ SarScreenDevice

Destructor

Definition

public ~ SarScreenDevice ();

Description

This is the destructor of SarScreenDevice class.

SarScreenDevice::sarIsConstructorFailed

Check error of constructor

Definition

public bool sarIsConstructorFailed();

Return Values

Return true when error occurred in constructor, otherwise false.

Description

Check if error occurred in constructor.

SarScreenDevice:: sarGetRotation

Retrieve the rotation angle of the screen

Definition

```
public int32_t sarGetRotation (
        SarRotation* rotation
        );
```

Arguments

[out] rotation	Angle of rotation

Return Values

Error code

Description

Retrive the rotation angle of screen.

SarCameraImageDrawer

Classes

SarCameraImageDrawer

Utility for drawing camera image

Definition

```
#include <SarCameraImageDrawer.h>
class SarCameraImageDrawer : SarNonCopyable {
public:
        SarCameraImageDrawer(SarSmart* smart);
        ~SarCameraImageDrawer();
        bool sarIsConstructorFailed();

        int32_t sarSetDrawRange(float x1, float y1, float x2, float y2);
        int32_t sarSetRotation(SarRotation rotation);
        int32_t sarSetFlipX(bool flipX);
        int32_t sarSetFlipY(bool flipY);

        int32_t sarStart();
        int32_t sarStop();
        int32_t sarDraw(const SarImage& image);
        int32_t sarDraw(const SarImage& image, const SarRect& rect);
};
```

Description

This is the class to draw camera image to the screen.

SarCameraImageDrawer works with OpenGLES1.0 and 2.0.

SarCameraImageDrawer::SarCameraImageDrawer

Constructor

Definition

```
public SarCameraImageDrawer(
        SarSmart* smart
        );
```

Arguments

[in] smart	Pointer to instance of SarSmart class

Description

This is the constructor of SarCameraImageDrawer class.

SarCameraImageDrawer::~SarCameraImageDrawer

Destructor

Definition

public ~SarCameraImageDrawer();

Description

This is the destructor of SarCameraImageDrawer class.

SarCameraImageDrawer::sarIsConstructorFailed

Check error of constructor

Definition

public bool sarIsConstructorFailed();

Return Values

Return true if error occurred in constructor, otherwise false.

Description

Check if any error occurred in constructor.

SarCameraImageDrawer::sarSetDrawRange

Set drawing range

Definition

```
public int32_t sarSetDrawRange(
        float x1,
        float y1,
        float x2,
        float y2
        );
```

Arguments

[in] x1	X coordinate of left limit of drawing range
[in] y1	Y coordinate of top limit of drawing range
[in] x2	X coordinate of right limit of drawing range
[in] y2	Y coordinate of lower limit of drawing range

Return Values

Error code

Description

Set drawing range of screen from -1.0 to 1.0.

Note:
This class performs nothing to the region not included in the drawing range.
If necessary call glClear() to clear frame buffer.

SarCameraImageDrawer::sarSetRotation

Set rotation angle when drawing camera image

Definition

```
public int32_t sarSetRotation(
        SarRotation rotation
        );
```

Arguments

[in] rotation	SarRotation angle

Return Values

Error code

Description

Set rotation angle of camera image to draw.

SarCameraImageDrawer::sarSetFlipX

Set if flip horizontally when drawing camera image

Definition

```
public int32_t sarSetFlipX (
        bool flipX
        );
```

Arguments

[in] flipX	If true camera image is flipped horizontally when drawing

Return Values

Error code

Description

Set if flip horizontally when drawing camera image.

SarCameraImageDrawer::sarSetFlipY

Set if flip camera image vertically when drawing

Definition

```
public int32_t sarSetFlipY (
        bool flipY
        );
```

Arguments

[in] flipY	If true flip camera image vertically when drawing

Return Values

Error code

Description

Set if flip camera image vertically when drawing.

SarCameraImageDrawer::sarStart

Initialization of drawing process

Definition

public int32_t sarStart();

Return Values

Error code

Description

Perform initialization needed to draw camera image.
Call this method from OpenGL drawing thread because this method uses OpenGL function.

SarCameraImageDrawer::sarStop

Release resource of drawing process

Definition

public int32_t sarStop();

Return Values

Error code

Description

Release resource used in camera image drawing process.
Call this method from OpenGL drawing thread because this method uses OpenGL function.

SarCameraImageDrawer::sarDraw

Draw camera image to the screen

Definition

```
public int32_t sarDraw(
        const Image& image
        );

public int32_t sarDraw(
        const SarImage& image,
        const SarRect& rect
        );
```

Arguments

[in] image	Instance of SarImage class contains image data to draw. Image format can be used currently are SAR_IMAGE_FORMAT_YCRCB420 and SAR_IMAGE_FORMAT_YCBCR420. Stride must be the same with the width of the image.
[in] rect	SarRect structure contains drawing range information

Return Values

Error code

Description

Draw camera image to the screen.
Camera image is drawn to the region indicated by sarSetDrawRange().
When drawing only a part of the camera image, store range information to SarRect structure and give it to the function.