KBSW190620

, publishDepthCamFrame

-
-
-
-
-
- 1Intel Realsense D435i Windows SDK
- 2Intel Realsense D435i Linux SDK

SDKpublishDepthCamFrameSlamware Core

RoboStudioSlamwareMCUMakehex

m
m

KBSW180147

KBSW180158 Breakout kit

2.6.7

slamware_core_platform.h

rpos::robot_platforms::SlamwareCorePlatform

publishDepthCamFrame(int sensorId, const rp

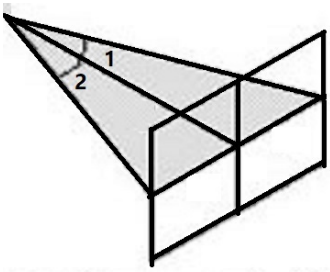
```
void publishDepthCamFrame(int sensorId, const rpos::message::depth_camera::DepthCameraFrame& frame);
```

sensorId: SlamwareId.
DepthCameraFrame: DepthCameraFrame
DepthCameraFrame

```
struct DepthCameraFrame
{
    float minValidDistance;   //@brief Min distance of the sensor (in meters)

    float maxValidDistance;   //@brief Max distance of the sensor (in meters)

    float minFovPitch;        //@brief The pitch of the first line of the frame (in rad)

    float maxFovPitch;        //@brief The pitch of the last line of the frame (in rad)

    float minFovYaw;          //@brief The yaw of the last column of the frame (in rad)

    float maxFovYaw;          //@brief The yaw of the first column of the frame (in rad)

    int cols;                 //@brief how many columns in this frame

    int rows;                 //@brief how many rows in this frame

    // TODO:
    // We now use std::vector to store data, which will make *copying* frames very expensive
    // we should use a data structure support CoW (copy on write) or just use shared pointer to store the internal data
    /**
    * @brief The data (order: row by row, cell by cell in each row)
    */
    std::vector<float> data;
};
```

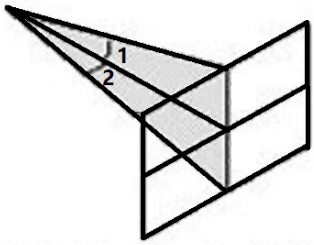
(FOV)DepthCameraFrame

1.
2. maxValidDistance2
3. colsrows320x240,
4. minFovPitchmaxFovPitchVFOV()VFOV=a(rad),minFovPitch=-a/2maxFovPitch=a/2
5. minFovYawmaxFovYawHFOV()HFOV=b(rad),minFovPitch=-b/2maxFovPitch=b/2
6. data



HFOV(水平视角)

1为maxFovYaw,
maxFovYaw=HFOV/2(rad);
2为minFovYaw,
minFovYaw=-HFOV/2(rad)



VFOV(垂直视角)

1为minFovPitch,
minFovPitch=-VFOV/2(rad);
2为maxFovPitch,
maxFovPitch=VFOV/2(rad)

publishDepthCamFrame

```
#include <rpos/robot_platforms/slamware_core_platform.h>

struct CamAttr
{
    float minValidDistance;
    float maxValidDistance;
    float minFovPitch;
    float maxFovPitch;
    float minFovYaw;
    float maxFovYaw;
    int cols;
    int rows;
};

int main(int argc, char* argv[])
{
    try
    {
        rpos::robot_platforms::SlamwareCorePlatform platform = rpos::robot_platforms::SlamwareCorePlatform::
connect("192.168.11.1", 1445);

        CamAttr camera_attr;
        getAttrFromDevice(camera_attr);        // TODO: Get attr from device doc maybe

        rpos::message::depth_camera::DepthCameraFrame frame;
        frame.minValidDistance = camera_attr.minValidDistance;
        frame.maxValidDistance = camera_attr.maxValidDistance;
        frame.minFovPitch = camera_attr.minFovPitch;
        frame.maxFovPitch = camera_attr.maxFovPitch;
        frame.minFovYaw = camera_attr.minFovYaw;
        frame.maxFovYaw = camera_attr.maxFovYaw;
        frame.cols = camera_attr.cols;
        frame.rows = camera_attr.rows;

        std::vector<float> frame_buffer;
        getFrameFromDepthCam(frame_buffer);    // TODO: Get Frame buffer from DepthCam Device


        int sensorId = 1; //TODO: Config your own SensorId
        platform.publishDepthCamFrame(sensorId, frame);
    }
    catch(const std::exception& ex)
    {
        std::string ex_str(ex.what());
    }
    return 0;
}
```

1.publishDepthCamFrame10

2.publishDepthCamFrame100ms

3.

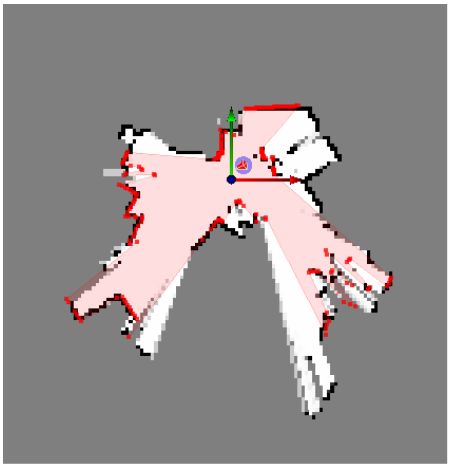
4.SDK

RoboStudioWeb portal

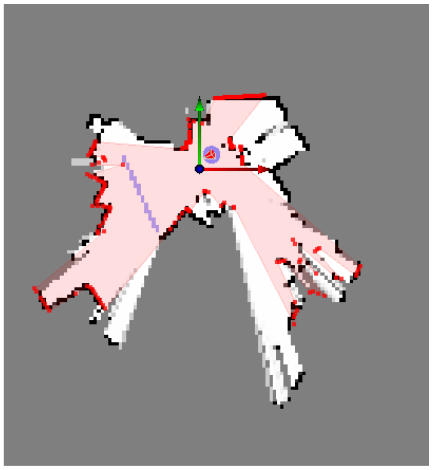
depthcamDepthCameraFramedatavectorVS2010

1. RoboStudio

Slamware Sensor MapRoboStudio



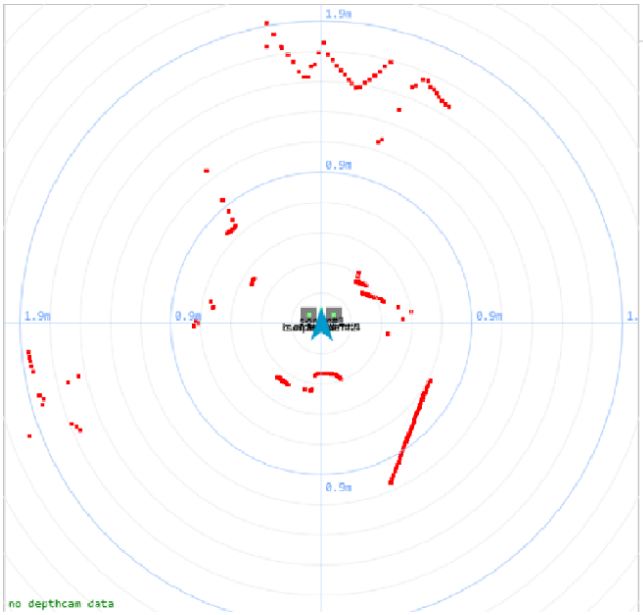
无深度信息



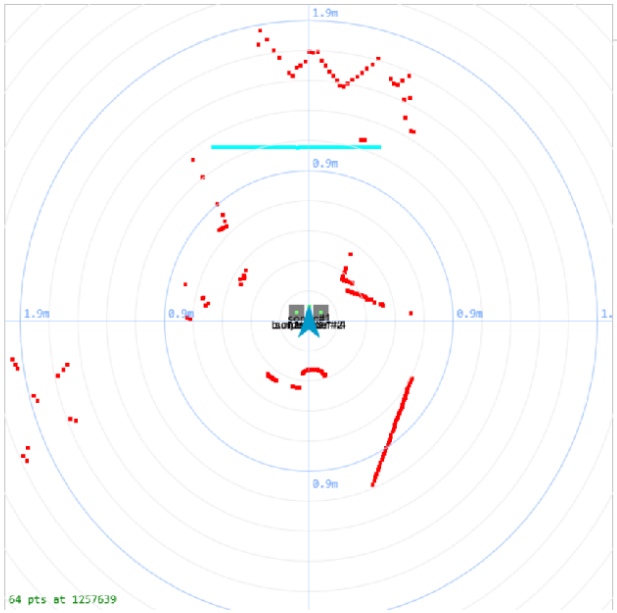
有深度信息

2. Web Portal

Web portalSlamware CoreKBSW180153 SLAMWARE Web Portal Function Overview



无深度信息



有深度信息

1Intel Realsense D435i Windows SDK

1. librealsense2

IntelD435ilibrealsense2

2. slamware sdk

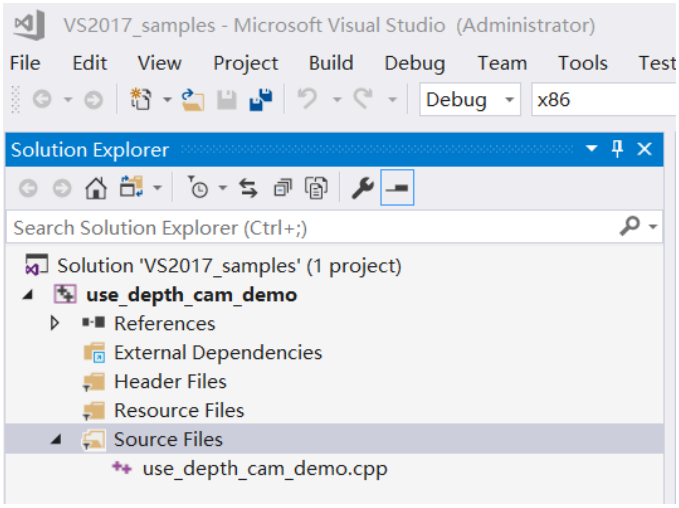
slamtecvs2017sdk

3. VS2017

librealsense2sdkVS2015slamwarewindows sdkVS2010VS2017VS2017

VS2017file new project Visual C++ Empty project

Source Files add new item Visual C++ C++ files(.cpp)cpp

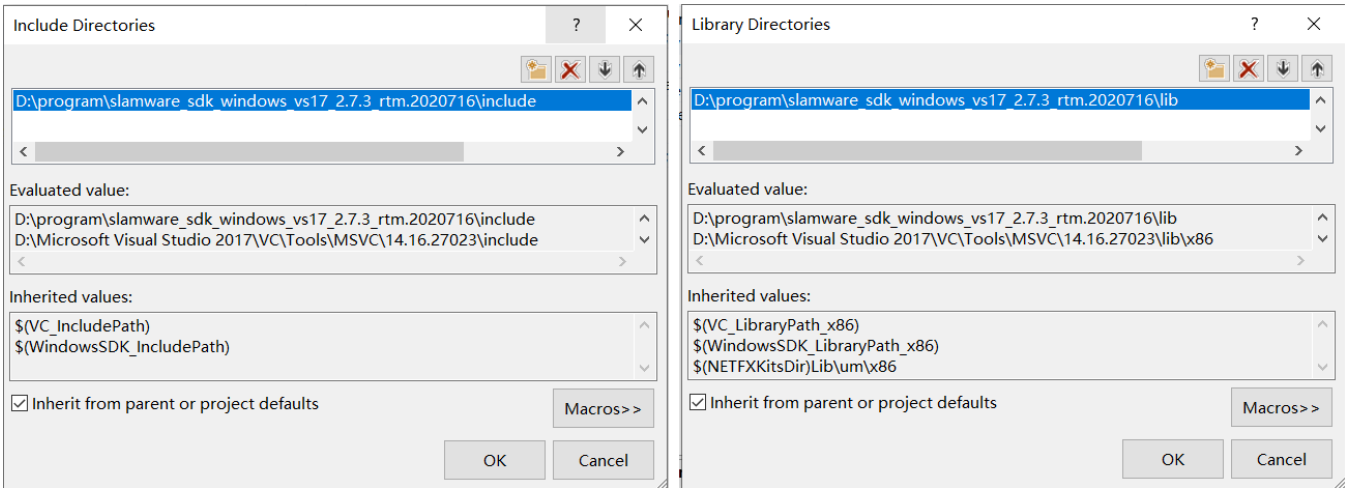


4.

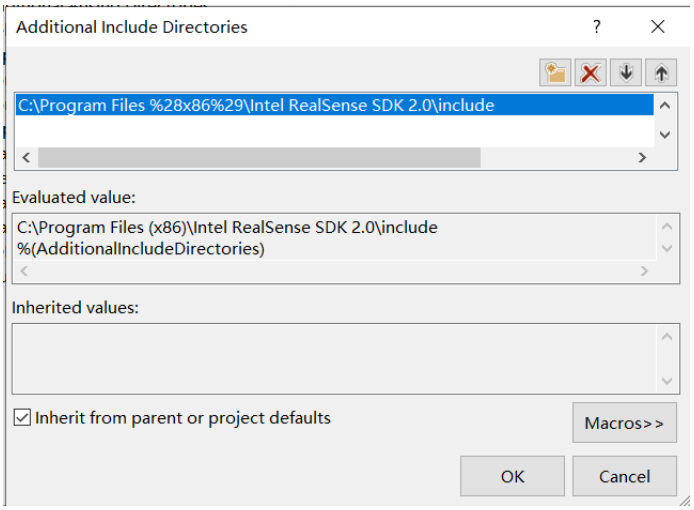
realsense SDKslamware SDK

PropertiesWin32

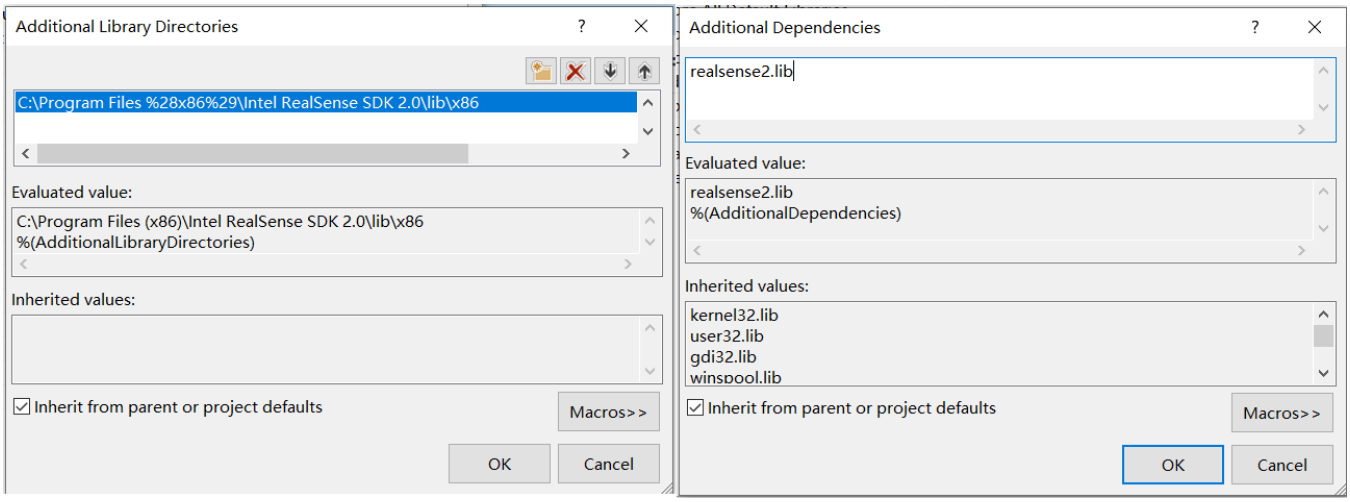
VC++DirectoriesInclude DirectoriesLibrary DirectoriesInclude Directoriesslamware SDKincludeLibrary Directoriesslamware SDKlib



PropertiesC/C++GeneralAdditional Include Directoriesrealsense SDKinclude



PropertiesLinkerGeneralAdditional Library Directoriesrealsense SDKlib/x86LinkerInputAdditional Dependenciesrealsense2.lib



5.

use_depth_cam_demo.cpp

```
#include <regex>
#include <iostream>
#include <boost/thread/thread.hpp>

#include <rpos/robot_platforms/slamware_core_platform.h>  //Include slamware API

#include <librealsense2/rs.hpp>  //Include RealSense Cross Platform API

std::string ip_address = "";
const char *ip_regex = "\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}";
rpos::robot_platforms::SlamwareCorePlatform sdp;

void ShowHelp(std::string app_name) {
        std::cout << "SLAMWARE console demo." << std::endl <<
                "Usage: " << app_name << " <slamware_address>" << std::endl;
}

bool ParseCommandLine(int argc, const char *argv[]) {
        bool opt_show_help = false;
        for (int pos = 1; pos < argc; ++pos) {
                const char *current = argv[pos];
```

```
                if (strcmp(current, "-h") == 0)
                        opt_show_help = true;
                else
                        ip_address = current;
        }
        std::regex reg(ip_regex);
        if (!opt_show_help && !std::regex_match(ip_address, reg))
                opt_show_help = true;
        if (opt_show_help) {
                ShowHelp("use_depth_camera_demo");
                return false;
        }
        return true;
}

int main(int argc, const char *argv[]) {
        if (!ParseCommandLine(argc, argv)) return 1;
        std::cout << "Connecting SDP @ " << ip_address << "..." << std::endl;
        try {
                //connect device
                sdp = rpos::robot_platforms::SlamwareCorePlatform::connect(ip_address, 1445);
                std::cout << "SDK Version: " << sdp.getSDKVersion() << std::endl;
                std::cout << "SDP Version: " << sdp.getSDPVersion() << std::endl;

                //this id should be same as you set the cam info via RS config tool
                int depthCamSensorId = 5;
                //create a depth msg frame
                rpos::message::depth_camera::DepthCameraFrame _frame;
                _frame.minValidDistance = 0.2f;
                _frame.maxValidDistance = 3.0f;
                _frame.minFovPitch = -30 * 3.14 / 180;
                _frame.maxFovPitch = 30 * 3.14 / 180;
                _frame.minFovYaw = -45 * 3.14 / 180;
                _frame.maxFovYaw = 45 * 3.14 / 180;
                _frame.cols = 320;
                _frame.rows = 240;
                //init data vector
                _frame.data.resize(_frame.cols * _frame.rows, 0.0f);

                // Create a Pipeline - this serves as a top-level API for streaming and processing frames
                rs2::pipeline p;
                // Configure and start the pipeline
                p.start();

                //process data
                while (true) {
                        // Block program until frames arrive
                        rs2::frameset rs_frames = p.wait_for_frames();
                        // Try to get a frame of a depth image
                        rs2::depth_frame _depth_frame = rs_frames.get_depth_frame();
                        // Get the depth frame's dimensions
                        int width = _depth_frame.get_width();
                        int height = _depth_frame.get_height();
                        std::cout << "x: " << width << " y: " << height << std::endl;
                        auto it = _frame.data.begin();
                        for (int i = (height - _frame.rows) / 2; i < height - (height - _frame.rows) / 2;
i++) {
                                for (int j = (width - _frame.cols) / 2; j < width - (width - _frame.cols) /
2; j++) {
                                        *it = _depth_frame.get_distance(j, i);
                                        it++;
                                }
                        }
                        sdp.publishDepthCamFrame(depthCamSensorId, _frame);
                        std::cout << "publish a depth frame..." << std::endl;

                        //control the publish rate less than 10HZ
                        boost::this_thread::sleep_for(boost::chrono::milliseconds(100));
                }
                return EXIT_SUCCESS;
        }
```

```
        catch (const rpos::system::detail::ExceptionBase& e) {
                std::cout << e.what() << std::endl;
                return EXIT_FAILURE;
        }
        catch (const rs2::error & e)
        {
                std::cerr << "RealSense error calling " << e.get_failed_function() << "(" << e.
get_failed_args() << "):\n    " << e.what() << std::endl;
                return EXIT_FAILURE;
        }
        catch (const std::exception& e)
        {
                std::cerr << e.what() << std::endl;
                return EXIT_FAILURE;
        }
}
```

6.

realsenseUSB3.0

PropertiesDebuggingCommand ArgumentsIP192.168.11.1

realsense2.dllrealsense SDKbin/x86realsense2.dll

2Intel Realsense D435i Linux SDK

1. librealsense2

IntelD435ilibrealsense2

2. slamware sdk

slamtecLinuxsdk

3.

slamware SDKSDKsamplesuse_depth_cam_demoMakefilesrcsrcmain.cpp

4.

main.cppwindow

5.

use_depth_cam_demo/Makefileslamware SDKlibrealsense2

Makefile

```
HOME_TREE := ../../

MODULE_NAME := $(notdir $(CURDIR))

CXXSRC            := src/main.cpp
CXXFLAGS   := -std=c++0x
LDFLAGS    := -Wl,-Bdynamic -lrealsense2

LD_LIBS    =  -Xlinker "-(" \
              -lstdc++ -ldl -lrt -lm -lpthread \
              -l$(RPOS_SLAMWARE_NAME) -l$(RPOS_CORE_NAME) \
              -lboost_atomic \
              -lboost_chrono \
              -lboost_date_time \
              -lboost_regex \
              -lboost_system \
              -lboost_thread \
              -lboost_filesystem \
              -lboost_random \
              -lbase64 \
              -ljsoncpp \
              -lrlelib \
              -lcrypto \
              -lcurl \
              -lssl \
              -Xlinker "-)" \

all: build_app

clean: clean_app

#includes the common definitions...
#do not remove them unless you know what you are doing
include $(HOME_TREE)/mak_def.inc
include $(HOME_TREE)/mak_common.inc
```

use_depth_cam_demomakegccg++4.9

linux-x86_64-release/output

```
$ ./use_depth_cam_demo 192.168.11.1
```