**目 录**

[**目 录** I](#_Toc59027062)

[**第一部分：环境配置** 2](#_Toc59027063)

[1.1奥比中光OpenNISDK安装（Linux） 2](#_Toc59027064)

[1.2 OpenCV在Arm平台上编译 3](#_Toc59027065)

[1.3 OpenCV+OpenNI2配置 3](#_Toc59027066)

[**第二部分：数据获取** 5](#_Toc59027067)

[2.1 获取深度图 8](#_Toc59027068)

[2.2 获取彩色图 10](#_Toc59027069)

[2.3 获取彩色点云 11](#_Toc59027070)

[**第三部分：数据配准** 17](#_Toc59027071)

[3.1配准代码 17](#_Toc59027072)

[**第四部分：姿势归一化** 21](#_Toc59027073)

[4.1 软件处理方法（自主研发） 21](#_Toc59027074)

[**第五部分：三维重建** 23](#_Toc59027075)

[5.1 数据准备 23](#_Toc59027076)

[5.2 初始化 23](#_Toc59027077)

[5.3 非刚配准 23](#_Toc59027078)

[5.4 结果评价 24](#_Toc59027079)

**研究主题：牲畜体表信息的三维重建**

**第一部分：环境配置**

1.1奥比中光OpenNISDK安装（Linux）

奥比中光针对ZaroP1开发板和深度摄像机提供了相关的[OpenNI2的SDK](https://abzg-oss.oss-cn-shenzhen.aliyuncs.com/files/OpenNI-Linux-Arm64-2.3.0.65.rar" \t "_blank),针对Windows/Linux/Android不同平台提供了相关的[安装文档](https://developer.orbbec.com.cn/technical_library.html?id=30" \t "_blank).根据官方文档将OpenNI2配置完成，注意官方文档中说Linux平台内置驱动不需要额外安装，但是需要仔细阅读README文件：

OpenNI

------

Website: http://structure.io/openni

Building Prerequisites(前置依赖)

======================

Linux

-----

- GCC 4.x

From: http://gcc.gnu.org/releases.html

Or via apt:

sudo apt-get install g++

- Python 2.6+/3.x

From: http://www.python.org/download/

Or via apt:

sudo apt-get install python

- LibUSB 1.0.x

From: http://sourceforge.net/projects/libusb/files/libusb-1.0/

Or via apt:

sudo apt-get install libusb-1.0-0-dev

- LibUDEV

sudo apt-get install libudev-dev

- JDK 6.0

From: http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u32-downloads-1594644.html

Or via apt:

sudo apt-get install openjdk-6-jdk

- FreeGLUT3

From: http://freeglut.sourceforge.net/index.php#download

Or via apt:

sudo apt-get install freeglut3-dev

- Doxygen

From: http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc

Or via apt:

sudo apt-get install doxygen

- GraphViz

From: http://www.graphviz.org/Download\_linux\_ubuntu.php

Or via apt:

sudo apt-get install graphviz

将以上的依赖安装完成之后，运行其中示例代码还需要安装对应的USB驱动，驱动安装方法如下此方法将同目录下的USB驱动文件拷贝到系统目录中，并自动生成环境文件指明OpenNI的路径此方法将同目录下的USB驱动文件拷贝到系统目录中，并自动生成环境文件指明OpenNI的路径,进入安装包OpenNI目录中:

**sh**

orbbec@localhost:OpenNI$ ./install.sh

最后，在安装文件中找到NiViewer运行文件，若文件不能运行，查看chmod权限。

**sh**

sudo ./NiViewer



**结果图片**

经过以上的配置过程，OpenNI的环境就配置完成了。

1.2 OpenCV在Arm平台上编译

在运用开发板设备获取数据的时候，通常运用OpenNI2获取数据流，通过OpenCV对数据流进行转换，生成RGB图片和深度图片。

源文件[官方地址](https://opencv.org/releases/)，本文用的是OpenCV3.4.3版本。OpenCV编译的方法自行百度，编译完成后对orbbec中OpenNI2SDK和opencv进行配置。

1.3 OpenCV+OpenNI2配置

根据官方给出的示例代码，进行编写。由于官方的代码是在Makefile文件中进行编写，所以opencv也需要在其中编写。根据Makefile文件编写规则进行编写。

**Makefile**

#Includes

CFLAGS = ${shell pkg-config opencv --cflags}

#lib

LDFLAGS = ${shell pkg-config opencv --libs}

其中pkg-config需要在系统中配置opencv.pc文件，文件内容，在安装的文件中可以找到，若没有则可能默认没有生成（opencv4之后的版本默认不生成），需要在编译的时候设置。

opencv环境配置没有问题后，在orbbec提供的OpenNI2的文件中示例代码中进行整合开发。文件位置在：

**sh**

/OpenNI/Samples/

选择其中SimpleViewer文件夹中的示例代码进行编写。在CommonCppMakefile文件夹中找到CFLAGS和LDFLAGS变量位置，在其后加上：

**Makefile**

#Includes

CFLAGS += ${shell pkg-config opencv --cflags}

#lib

LDFLAGS += ${shell pkg-config opencv --libs}

此外还需要在Makefile文件中加入OpenNI头文件的路径和Redist的路径，这就是在配置OpenNI时运行install.sh文件生成的环境文件中的内容，也在同级目录下。其后编译即可通过。接下来就可以使用opencv对数据进行获取保存了。

**第二部分：数据获取**

根据上文环境配置的结果，对数据获取部分的代码进行调整，由于官方OpenNI2代码运用Makefile进行管理，所以将OpenCV的配置也运用Makefile进行管理。本文以官方示例代码SimpleViewer文件加下的文件作为基础，对其中的Makefile进行配置。

首先，“Makefile”文件内容进行修改,红色部分为添加部分，内容是官方OpenNI2的相关文件路径:

*include* CommonDefs.mak

BIN\_DIR = Bin

OPENNI2\_INCLUDE=/home/orbbec/Desktop/OpenNI-Linux-Arm64-2.3.0.65/OpenNI-Linux-Arm64-2.3.0.65/include/

OPENNI2\_REDIST=/home/orbbec/Desktop/OpenNI-Linux-Arm64-2.3.0.65/OpenNI-Linux-Arm64-2.3.0.65/Redist/

INC\_DIRS = \

    ../../Include \

    ../../ThirdParty/GL/ \

    ../Common

SRC\_FILES = \*.cpp

*ifeq* ("$(OSTYPE)","Darwin")

    CFLAGS += -DMACOS

    LDFLAGS += -framework OpenGL -framework GLUT

*else*

    CFLAGS += -DUNIX -DGLX\_GLXEXT\_LEGACY

    USED\_LIBS += glut GL

*endif*

USED\_LIBS += OpenNI2

EXE\_NAME = SimpleViewer

CFLAGS += -Wall

*ifndef* OPENNI2\_INCLUDE

    $(error OPENNI2\_INCLUDE is not defined. Please define it or 'source' the OpenNIDevEnvironment file from the installation)

else ifndef OPENNI2\_REDIST

    $(error OPENNI2\_REDIST is not defined. Please define it or 'source' the OpenNIDevEnvironment file from the installation)

*endif*

INC\_DIRS += $(OPENNI2\_INCLUDE)

*include* CommonCppMakefile

.PHONY: copy-redist

copy-redist:

    cp -R $(OPENNI2\_REDIST)/\* $(OUT\_DIR)

$(OUTPUT\_FILE): copy-redist

接着对“CommonCppMakefile”文件进行修改，红色部分为修改内容，添加的内容主要为OpenCV的编译文件的路径，由环境配置可知，在pkgconfig中配置OpenCV.pc文件后在系统中调用相关命令即可给出OpenCV安装文件的路径：

*# take this file's dir*

COMMON\_CPP\_MAKE\_FILE\_DIR = $(dir $(lastword $(MAKEFILE\_LIST)))

*include* $(COMMON\_CPP\_MAKE\_FILE\_DIR)CommonDefs.mak

*# define a function to figure .o file for each source file (placed under intermediate directory)*

SRC\_TO\_OBJ = $(addprefix ./$(INT\_DIR)/,$(addsuffix .o,$(notdir $(basename *$1*))))

*# create a list of all object files*

OBJ\_FILES = $(call SRC\_TO\_OBJ,$(SRC\_FILES\_LIST))

*# define a function to translate any source file to its dependency file (note that the way we create*

*# dep files, as a side affect of compilation, always puts the files in the INT\_DIR with suffix .d)*

SRC\_TO\_DEP = $(addprefix ./$(INT\_DIR)/,$(addsuffix .d,$(notdir $(basename *$1*))))

*# create a list of all dependency files*

DEP\_FILES = $(call SRC\_TO\_DEP,$(SRC\_FILES\_LIST))

*# older version of gcc doesn't support the '=' symbol in include dirs, so we replace it ourselves with sysroot*

INC\_DIRS\_FROM\_SYSROOT = $(patsubst =/%,$(TARGET\_SYS\_ROOT)/%,$(INC\_DIRS))

*# append the -I switch to each include directory*

INC\_DIRS\_OPTION = $(foreach dir,$(INC\_DIRS\_FROM\_SYSROOT),-I$(dir))

*# append the -L switch to each library directory*

LIB\_DIRS\_OPTION = $(foreach dir,$(LIB\_DIRS),-L$(dir)) -L$(OUT\_DIR)

*# append the -l switch to each library used*

USED\_LIBS\_OPTION = $(foreach lib,$(USED\_LIBS),-l$(lib))

*# append the -D switch to each define*

DEFINES\_OPTION = $(foreach def,$(DEFINES),-D$(def))

*# tell compiler to use the target system root*

*ifdef* TARGET\_SYS\_ROOT

    CFLAGS += --sysroot=$(TARGET\_SYS\_ROOT)

    LDFLAGS += --sysroot=$(TARGET\_SYS\_ROOT)

*endif*

*# set Debug / Release flags*

*ifeq* "$(CFG)" "Debug"

    CFLAGS += -O0 -g

*endif*

*ifeq* "$(CFG)" "Release"

    CFLAGS += -O2 -DNDEBUG

*endif*

CFLAGS += $(INC\_DIRS\_OPTION) $(DEFINES\_OPTION)

CFLAGS += -fPIC -fvisibility=hidden

*ifneq* "$(ALLOW\_WARNINGS)" "1"

    CFLAGS += -Werror

*ifeq* ("$(OSTYPE)","Darwin")

    CFLAGS += -Wno-deprecated-declarations -Wno-unused-private-field -Wno-unused-const-variable

*endif*

*endif*

LDFLAGS += $(LIB\_DIRS\_OPTION) $(USED\_LIBS\_OPTION)

*##############this is out method######################*

CFLAGS += ${shell pkg-config opencv --cflags}

LDFLAGS += ${shell pkg-config opencv --libs}

*#####################################*

*# some lib / exe specifics*

*ifneq* "$(LIB\_NAME)" ""

    OUTPUT\_NAME = lib$(LIB\_NAME).so

*ifneq* ("$(OSTYPE)","Darwin")

        LDFLAGS += -Wl,--no-undefined

        OUTPUT\_NAME = lib$(LIB\_NAME).so

        OUTPUT\_COMMAND = $(CXX) -o $(OUTPUT\_FILE) $(OBJ\_FILES) $(LDFLAGS) -shared

*else*

        LDFLAGS += -undefined error

        OUTPUT\_NAME = lib$(LIB\_NAME).dylib

        OUTPUT\_COMMAND = $(CXX) -o $(OUTPUT\_FILE) $(OBJ\_FILES) $(LDFLAGS) -dynamiclib -headerpad\_max\_install\_names -install\_name $(OUTPUT\_NAME)

*endif*

*endif*

*ifneq* "$(EXE\_NAME)" ""

    OUTPUT\_NAME = $(EXE\_NAME)

*# We want the executables to look for the .so's locally first:*

    LDFLAGS += -Wl,-rpath ./

    OUTPUT\_COMMAND = $(CXX) -o $(OUTPUT\_FILE) $(OBJ\_FILES) $(LDFLAGS)

*endif*

*ifneq* "$(SLIB\_NAME)" ""

    OUTPUT\_NAME = lib$(SLIB\_NAME).a

*ifneq* ("$(OSTYPE)","Darwin")

        OUTPUT\_COMMAND = $(AR) -cq $(OUTPUT\_FILE) $(OBJ\_FILES)

*else*

        OUTPUT\_COMMAND = libtool -static -o $(OUTPUT\_FILE) $(OBJ\_FILES)

*endif*

*endif*

*define* CREATE\_SRC\_TARGETS

*# create a target for the object file (the CXX command creates both an .o file*

*# and a .d file)*

*ifneq* ("$(OSTYPE)","Darwin")

$(call SRC\_TO\_OBJ,*$1*) : *$1* | $(INT\_DIR)

    $(CXX) -MD -MP -MT "$(call SRC\_TO\_DEP,*$1*) *$$*@" -c $(CFLAGS) -o *$$*@ *$$*<

*else*

$(call SRC\_TO\_OBJ,*$1*) : *$1* | $(INT\_DIR)

    $(CXX) -x c++ -c $(CFLAGS) -o *$$*@ *$$*<

*endif*

*endef*

*#############################################################################*

*# Targets*

*#############################################################################*

.PHONY: clean-objs clean-defs

*include* $(COMMON\_CPP\_MAKE\_FILE\_DIR)CommonTargets.mak

*# create targets for each source file*

$(foreach src,$(SRC\_FILES\_LIST),$(eval $(call CREATE\_SRC\_TARGETS,$(src))))

*# include all dependency files (we don't need them the first time, so we can use -include)*

*-include* $(DEP\_FILES)

$(OUTPUT\_FILE): $(OBJ\_FILES)

    $(OUTPUT\_COMMAND)

clean-objs:

    rm -rf $(OBJ\_FILES)

clean-defs:

    rm -rf $(DEP\_FILES)

clean: clean-objs clean-defs

这样本文以官方SimpleViewer文件夹为基础的Makefile相关文件的编写就完成了，接着就是对文件中主函数文件main.cpp进行代码编写，这样就可以结合openNI和OpenCV进行数据获取。

2.1 获取深度图

获取深度图主要用到了openni的接口获取数据流，然后将获取的数据流通过opencv接口转换成深度图像。

*#include* <OpenNI.h>

*#include* "Viewer.h"

*#include* <stdio.h>

*#include* <iostream>

*#include* <highgui.h>

*#include* <opencv2/opencv.hpp>

*#include* <opencv2/core/core.hpp>

*#include* <opencv2/highgui/highgui.hpp>

*#include* <opencv2/imgproc/imgproc.hpp>

using namespace std;

using namespace openni;

*#define* SAMPLE\_READ\_WAIT\_TIMEOUT 2000*//2000ms*

int wasKeyboardHit();

int main(int argc, char\*\* argv)

{

    openni::Status rc = openni::STATUS\_OK;

    openni::Status rc\_d = openni::STATUS\_OK;

    openni::Device device;

    openni::VideoStream depth, color;

    const char\* deviceURI = openni::ANY\_DEVICE;

    rc = openni::OpenNI::initialize();

    printf("After initialization:\n%s\n", openni::OpenNI::getExtendedError());

    rc = device.open(deviceURI);

*if* (rc != openni::STATUS\_OK)

    {

        printf("SimpleViewer: Device open failed:\n%s\n", openni::OpenNI::getExtendedError());

        openni::OpenNI::shutdown();

*return* 1;

    }

    rc = depth.create(device, openni::SENSOR\_DEPTH);

*if* (rc == openni::STATUS\_OK)

        rc = depth.start();

    openni::VideoFrameRef frameDepth;

    openni::DepthPixel\* pDepth;

    cout<<"stream start, press any key to stop..."<<endl;

    VideoStream\* streams[] = {&depth, &color};

*while* (!wasKeyboardHit())

    {

        int depthstream = -1;

        rc = OpenNI::waitForAnyStream(streams, 2, &depthstream, SAMPLE\_READ\_WAIT\_TIMEOUT);

*if* (rc != STATUS\_OK)

        {

            printf("Wait failed! (timeout is %d ms)\n%s\n", SAMPLE\_READ\_WAIT\_TIMEOUT, OpenNI::getExtendedError());

*continue*;

        }

        cout<<"waitForAnyStream finished, press any key to stop..."<<endl;

*//get color frame*

        rc\_d = depth.readFrame(&frameDepth);

        cout<<"read Frame finished, press any key to stop..."<<endl;

*if* (rc != STATUS\_OK || rc\_d != STATUS\_OK)

        {

            printf("Read failed!\n%s\n", OpenNI::getExtendedError());

*continue*;

        }

        int middleIndex = (frameDepth.getHeight() + 1)\*frameDepth.getWidth() / 2;

        pDepth = (DepthPixel\*)frameDepth.getData();

            cout  << pDepth[middleIndex]  <<endl;

    }

    const cv::Mat mImageDepth( frameDepth.getHeight(), frameDepth.getWidth(), CV\_16UC1, (void\*)frameDepth.getData());

    cv::Mat mScaledDepth;

    int iMaxDepth = depth.getMaxPixelValue();

    mImageDepth.convertTo( mScaledDepth, CV\_8U, 255.0 / iMaxDepth );

    cv::imwrite( "/home/orbbec/Desktop/createDepthImg/Bin/Arm64-Release/DepthImage.png", mScaledDepth );

}

2.2 获取彩色图

获取彩色图不能通过openni接口 readframe()，需要使用opencv的 VideoCapture相关接口。

*#include* <opencv2/opencv.hpp>

using namespace cv;

int main()

{

    VideoCapture videoCapture;

    videoCapture.open(cv::CAP\_ANY);

*if* (!videoCapture.isOpened())

    {

        printf("open UVC color camera failed.");

    }

*//set codec format*

    videoCapture.set(CV\_CAP\_PROP\_FOURCC, CV\_FOURCC('M', 'J', 'P', 'G'));

*//set resolution*

    videoCapture.set(CV\_CAP\_PROP\_FRAME\_WIDTH, 640);

    videoCapture.set(CV\_CAP\_PROP\_FRAME\_HEIGHT, 480);

*for*(int i=0;i<10;i++)

    {

        Mat frame;

        videoCapture >> frame;

        Vec3b &bgr = frame.at<Vec3b>(frame.rows/2, frame.cols/2);

*//print the r g b value of the middle pixel of the frame*

        printf("r = %02d, g = %02d, b = %02d\n",

            bgr[2] & 0xff,

            bgr[1] & 0xff,

            bgr[0] & 0xff);

        waitKey(30);*//delay 30ms*

        imwrite("/home/orbbec/Desktop/saveRGBImg/Bin/Arm64-Release/saved\_img.png",frame);

    }

*return* 0;

}

2.3 获取彩色点云

根据opencv中videoCapture获得的带有彩色Mat数据做为参数输入，最终在文件写入时与坐标一起写入至ply文件中：

*#include* <iostream>

*#include* <sstream>

*#include* "OpenNI.h"

*#include* "OniSampleUtilities.h"

*#include* <opencv2/opencv.hpp>

using namespace openni;

using namespace cv;

*#define* MIN\_DISTANCE 20*//µ¥Î»ºÁÃ×*

*#define* MAX\_DISTANCE 4000*//µ¥Î»ºÁÃ×*

*#define* RESOULTION\_X 640.0*//±ê¶šÊ±µÄ·Ö±æÂÊ*

*#define* RESOULTION\_Y 480.0*//±ê¶šÊ±µÄ·Ö±æÂÊ*

*#define* MAX\_FRAME\_COUNT 50

typedef struct xnIntrinsic\_Params

{

    xnIntrinsic\_Params() :

        c\_x(320.0), c\_y(240.0), f\_x(480.0), f\_y(480.0)

    {}

    xnIntrinsic\_Params(float c\_x\_, float c\_y\_, float f\_x\_, float f\_y\_) :

        c\_x(c\_x\_), c\_y(c\_y\_), f\_x(f\_x\_),f\_y(f\_y\_)

    {}

    float c\_x;*//uÖáÉÏµÄ¹éÒ»»¯œ¹Ÿà*

    float c\_y;*//vÖáÉÏµÄ¹éÒ»»¯œ¹Ÿà*

    float f\_x;*//Ö÷µãx×ø±ê*

    float f\_y;*//Ö÷µãy×ø±ê*

}xIntrinsic\_Params;

xIntrinsic\_Params g\_IntrinsicParam;*//ŽæŽ¢Ïà»úÄÚ²ÎµÄÈ«ŸÖ±äÁ¿*

void getCameraParams(openni::Device& Device, xIntrinsic\_Params& IrParam)

{

    OBCameraParams cameraParam;

    int dataSize = sizeof(cameraParam);

    memset(&cameraParam, 0, sizeof(cameraParam));

    openni::Status rc = Device.getProperty(openni::OBEXTENSION\_ID\_CAM\_PARAMS, (uint8\_t \*)&cameraParam, &dataSize);

*if* (rc != openni::STATUS\_OK)

    {

        std::cout << "Error:" << openni::OpenNI::getExtendedError() << std::endl;

*return*;

    }

    std::cout << "IrParam.f\_x = " << IrParam.f\_x << std::endl;

    std::cout << "IrParam.f\_y = " << IrParam.f\_y << std::endl;

    std::cout << "IrParam.c\_x = " << IrParam.c\_x << std::endl;

    std::cout << "IrParam.c\_y = " << IrParam.c\_y << std::endl;

}

void convertDepthToPointCloud(const uint16\_t \*pDepth,  Mat color, int width, int height,const char \*ply\_filename)

{

*if* (NULL == pDepth)

    {

        printf("depth frame is NULL!");

*return*;

    }

    FILE \*fp;

    fp = fopen(ply\_filename, "w");

    int valid\_count = 0;

    uint16\_t max\_depth = MAX\_DISTANCE;

    uint16\_t min\_depth = MIN\_DISTANCE;

*for* (int v = 0; v < height; ++v)

    {

*for* (int u = 0; u < width; ++u)

        {

            uint16\_t depth = pDepth[v \* width + u];

*if* (depth <= 0 || depth < min\_depth || depth > max\_depth)

*continue*;

            valid\_count += 1;

        }

    }

    fprintf(fp, "ply\n");

    fprintf(fp, "format ascii 1.0\n");

    fprintf(fp, "element vertex %d\n", valid\_count);

    fprintf(fp, "property float x\n");

    fprintf(fp, "property float y\n");

    fprintf(fp, "property float z\n");

    fprintf(fp, "property uchar red\n");

    fprintf(fp, "property uchar green\n");

    fprintf(fp, "property uchar blue\n");

    fprintf(fp, "end\_header\n");

    float world\_x, world\_y, world\_z;

*for* (int v = 0; v < height; ++v)

    {

*for* (int u = 0; u < width; ++u)

        {

            uint16\_t depth = pDepth[v \* width + u];

            Vec3b &bgr = color.at<Vec3b>(v, u);

            int color\_b= bgr[0] & 0xff;

            int color\_g= bgr[1] & 0xff;

            int color\_r= bgr[2] & 0xff;

*if* (depth <= 0 || depth < min\_depth || depth > max\_depth)

*continue*;

            float fdx = g\_IntrinsicParam.f\_x \* ((float)(width) / RESOULTION\_X);

            float fdy = g\_IntrinsicParam.f\_y \* ((float)(height) / RESOULTION\_Y);

            float u0 = g\_IntrinsicParam.c\_x \* ((float)(width)/ RESOULTION\_X);

            float v0 = g\_IntrinsicParam.c\_y \* ((float)(height) / RESOULTION\_Y);

            float tx = (u - u0) / fdx;

            float ty = (v - v0) / fdy;

            world\_x = depth \* tx;

            world\_y = depth \* ty;

            world\_z = depth;

            fprintf(fp, "%f %f %f %d %d %d\n", world\_x, world\_y, world\_z, color\_r,color\_g,color\_b);

        }

    }

    fclose(fp);

}

int g\_imageCount = 0;

void analyzeFrame(const VideoFrameRef& frame,  Mat color)

{

    DepthPixel\* pDepth;

*//¹¹ÔìµãÔÆÎÄŒþÃû*

    char plyFileName[256] = "";

    g\_imageCount++;

*if* (MAX\_FRAME\_COUNT < g\_imageCount)

    {

*return*;

    }

    std::stringstream filename;

    filename << "pointcloud\_";

    filename << g\_imageCount;

    filename << ".ply";

    filename >> plyFileName;

    int middleIndex = (frame.getHeight() + 1)\*frame.getWidth() / 2;

*switch* (frame.getVideoMode().getPixelFormat())

    {

*case* PIXEL\_FORMAT\_DEPTH\_1\_MM:

        pDepth = (DepthPixel\*)frame.getData();

        printf("[%08llu] %8d\n", (long long)frame.getTimestamp(),pDepth[middleIndex]);

        convertDepthToPointCloud(pDepth, color, frame.getWidth(), frame.getHeight(), plyFileName);

*break*;

*default*:

        printf("Unknown format\n");

    }

}

int main(int argc, char\* argv[])

{

*//initialize openNI sdk*

    Status rc = OpenNI::initialize();

*if* (rc != STATUS\_OK)

    {

        printf("Initialize failed\n%s\n", OpenNI::getExtendedError());

*return* 1;

    }

*//open deivce*

    Device device;

    rc = device.open(ANY\_DEVICE);

*if* (rc != STATUS\_OK)

    {

        printf("Couldn't open device\n%s\n", OpenNI::getExtendedError());

*return* 2;

    }

    VideoCapture videoCapture;

    videoCapture.open(cv::CAP\_ANY);

*if* (!videoCapture.isOpened())

    {

        printf("open UVC color camera failed.");

    }

*//set codec format*

    videoCapture.set(CV\_CAP\_PROP\_FOURCC, CV\_FOURCC('M', 'J', 'P', 'G'));

*//set resolution*

    videoCapture.set(CV\_CAP\_PROP\_FRAME\_WIDTH, 640);

    videoCapture.set(CV\_CAP\_PROP\_FRAME\_HEIGHT, 480);

    VideoStream depth;

*//create depth stream*

*if* (device.getSensorInfo(SENSOR\_DEPTH) != NULL)

    {

        rc = depth.create(device, SENSOR\_DEPTH);

*if* (rc != STATUS\_OK)

        {

            printf("Couldn't create depth stream\n%s\n", OpenNI::getExtendedError());

        }

    }

    rc = depth.start();

*if* (rc != STATUS\_OK)

    {

        printf("Couldn't start the depth stream\n%s\n", OpenNI::getExtendedError());

    }

    openni::VideoFrameRef frameDepth;

*for*(int i=0;i<10;i++)

    {

        Mat color\_frame;

        videoCapture >> color\_frame;

        Vec3b &bgr = color\_frame.at<Vec3b>(color\_frame.rows/2, color\_frame.cols/2);

*//print the r g b value of the middle pixel of the frame*

        printf("r = %02d, g = %02d, b = %02d\n",

            bgr[2] & 0xff,

            bgr[1] & 0xff,

            bgr[0] & 0xff);

        rc = depth.readFrame(&frameDepth);

        analyzeFrame(frameDepth, color\_frame);

*if* (rc != openni::STATUS\_OK)

*continue*;

        waitKey(30);

    }

    depth.stop();

    depth.destroy();

*//close device*

    device.close();

*//shutdown OpenNI*

    OpenNI::shutdown();

*return* 0;

}

**第三部分：数据配准**

根据上文数据获取的准备，在Zora P1开发板上进行数据获取工作，作为三维重建的基础，本文只获取了两侧的点云数据进行三维重建，在数据方面要求较低，容易获取。

3.1配准代码

获取得到牲畜的两侧数据后，接下来需要对数据进行配准。本文配准方法主要运用两侧点云数据的同名点，对应点的获取通过点云库手动选择得出，通过对应点计算得出旋转矩阵进行配准。配准函数代码如下：

*#include* <iostream>

*#include* <string>

*#include* <pcl/io/ply\_io.h>

*#include* <pcl/registration/icp.h>

*#include* <pcl/visualization/pcl\_visualizer.h>

*#include* <pcl/console/time.h>   *// TicToc*

*#include* <pcl/common/common.h>

*#include* <pcl/common/angles.h>

*#include* <pcl/common/transforms.h>

*#include* <pcl/point\_cloud.h>

*#include* <pcl/point\_types.h>

*#include* <pcl/io/pcd\_io.h>

*#include* <pcl/registration/transformation\_estimation\_svd.h>

typedef pcl::PointXYZRGB PointT;

typedef pcl::PointCloud<PointT> PointCloudT;

using namespace pcl;

int main(int argc,char\* argv[])

{

*// The point clouds we will be using*

    PointCloudT::Ptr cloud\_in(new PointCloudT);*// Original point cloud*

    PointCloudT::Ptr cloud\_in2(new PointCloudT);*// Transformed point cloud*

    PointCloudT::Ptr cloud\_icp(new PointCloudT);*// ICP output point cloud*

    PointCloudT::Ptr corrspond\_in(new PointCloudT);*// Original point cloud*

    PointCloudT::Ptr corrspond\_out(new PointCloudT);*// Transformed point cloud*

    PointCloudT::Ptr show\_transformed\_clouds(new PointCloudT);

    std::mutex cloud\_mutex\_;

*// Checking program arguments*

*if* (argc < 3)

    {

        printf("Usage :\n");

        printf("\t\t%s file.ply number\_of\_ICP\_iterations\n", argv[0]);

        PCL\_ERROR("Provide one ply file.\n");

*return* (-1);

    }

    pcl::console::TicToc time;

    time.tic();

*if* (pcl::io::loadPLYFile(argv[1], \*cloud\_in) < 0)

    {

        PCL\_ERROR("Error loading cloud %s.\n", argv[1]);

*return* (-1);

    }

*if* (pcl::io::loadPLYFile(argv[2], \*cloud\_in2) < 0)

    {

        PCL\_ERROR("Error loading cloud %s.\n", argv[2]);

*return* (-1);

    }

    std::cout << "\nLoaded file " << argv[1] << " (" << cloud\_in->size() << " points) in " << time.toc() << " ms\n" << std::endl;

    Eigen::Matrix4d transformation\_matrix = Eigen::Matrix4d::Identity();

    PointT point\_in\_1(-0.201973, -0.326478, 1.578000, 0, 0, 0);

    PointT point\_in\_2(-0.328673, -0.325839, 1.616000, 0, 0, 0);

    PointT point\_in\_3(-0.297260, -0.269961, 1.730000, 0, 0, 0);

    PointT point\_in\_4(-0.169498, -0.276550, 1.696000, 0, 0, 0);

    corrspond\_in->push\_back(point\_in\_1);

    corrspond\_in->push\_back(point\_in\_2);

    corrspond\_in->push\_back(point\_in\_3);

    corrspond\_in->push\_back(point\_in\_4);

    PointT point\_in\_5(0.165862, -0.074352, 1.631000, 0, 0, 0);

    PointT point\_in\_6(0.290257, -0.068296, 1.623000, 0, 0, 0);

    PointT point\_in\_7(0.298880, -0.133429, 1.522000, 0, 0, 0);

    PointT point\_in\_8(0.175316, -0.138128, 1.515000, 0, 0, 0);

    corrspond\_out->push\_back(point\_in\_5);

    corrspond\_out->push\_back(point\_in\_6);

    corrspond\_out->push\_back(point\_in\_7);

    corrspond\_out->push\_back(point\_in\_8);

    cout << corrspond\_in->size() << endl;

    cout << corrspond\_out->size() << endl;

*//利用SVD方法求解变换矩阵*

    pcl::registration::TransformationEstimationSVD<pcl::PointXYZRGB, pcl::PointXYZRGB> TESVD;

    pcl::registration::TransformationEstimationSVD<pcl::PointXYZRGB, pcl::PointXYZRGB>::Matrix4 transformation2;

    TESVD.estimateRigidTransformation(\*corrspond\_in, \*corrspond\_out, transformation2);

    cout << transformation2 << endl;

    pcl::transformPointCloud(\*cloud\_in, \*show\_transformed\_clouds, transformation2);

    \*show\_transformed\_clouds += \*cloud\_in2;

    pcl::PCDWriter writer;

    writer.write("registered.pcd", \*show\_transformed\_clouds, false);

*return* (0);

}

CMakeLists.txt文件内容如下

cmake\_minimum\_required(VERSION 2.6 FATAL\_ERROR)

project(transform\_twoside)

find\_package(PCL 1.5 REQUIRED)

include\_directories(${PCL\_INCLUDE\_DIRS})

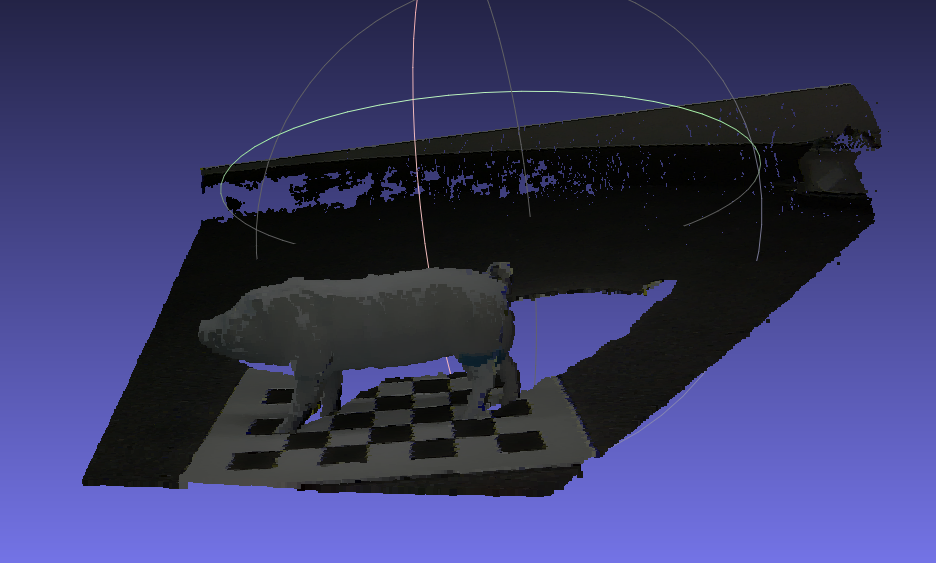
link\_directories(${PCL\_LIBRARY\_DIRS})

add\_definitions(${PCL\_DEFINITIONS})

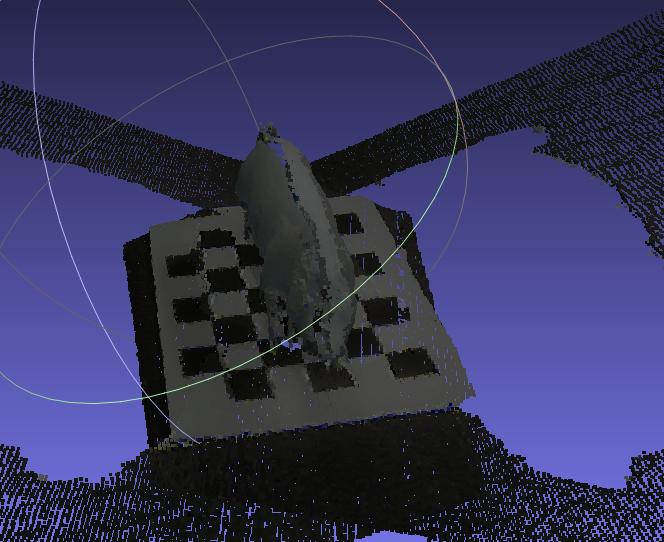
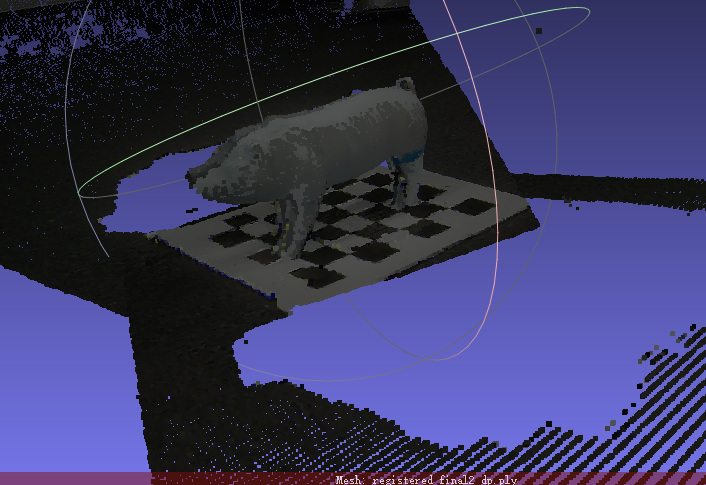
add\_executable (transform\_twoside transform\_twoside.cpp)

target\_link\_libraries (transform\_twoside ${PCL\_LIBRARIES})

获取的原始数据：



配准后数据：



**第四部分：姿势归一化**

本方法中三维重建需要先将双侧配准的点云数据中的从场景中分割出来，并将牲畜按照统一的方向放置于相同的坐标系统中，因此在进行三维重建前，需要将数据进行姿势归一化处理，以下给出姿势归一化方法。

## 4.1 软件处理方法（自主研发）

软件下载地址：<https://github.com/LiveStockShapeAnalysis/LiveStockShapeAnalysis-Software/releases/tag/v1.5>

软件实现方法论文参考：

@article{GUO201760,

title = "LSSA\_CAU: An interactive 3d point clouds analysis software for body measurement of livestock with similar forms of cows or pigs",

journal = "Computers and Electronics in Agriculture",

volume = "138",

pages = "60 - 68",

year = "2017",

issn = "0168-1699",

doi = "https://doi.org/10.1016/j.compag.2017.04.014",

url = "http://www.sciencedirect.com/science/article/pii/S0168169917301746",

}

@article{GUO201959,

title = "A bilateral symmetry based pose normalization framework applied to livestock body measurement in point clouds",

journal = "Computers and Electronics in Agriculture",

volume = "160",

pages = "59 - 70",

year = "2019",

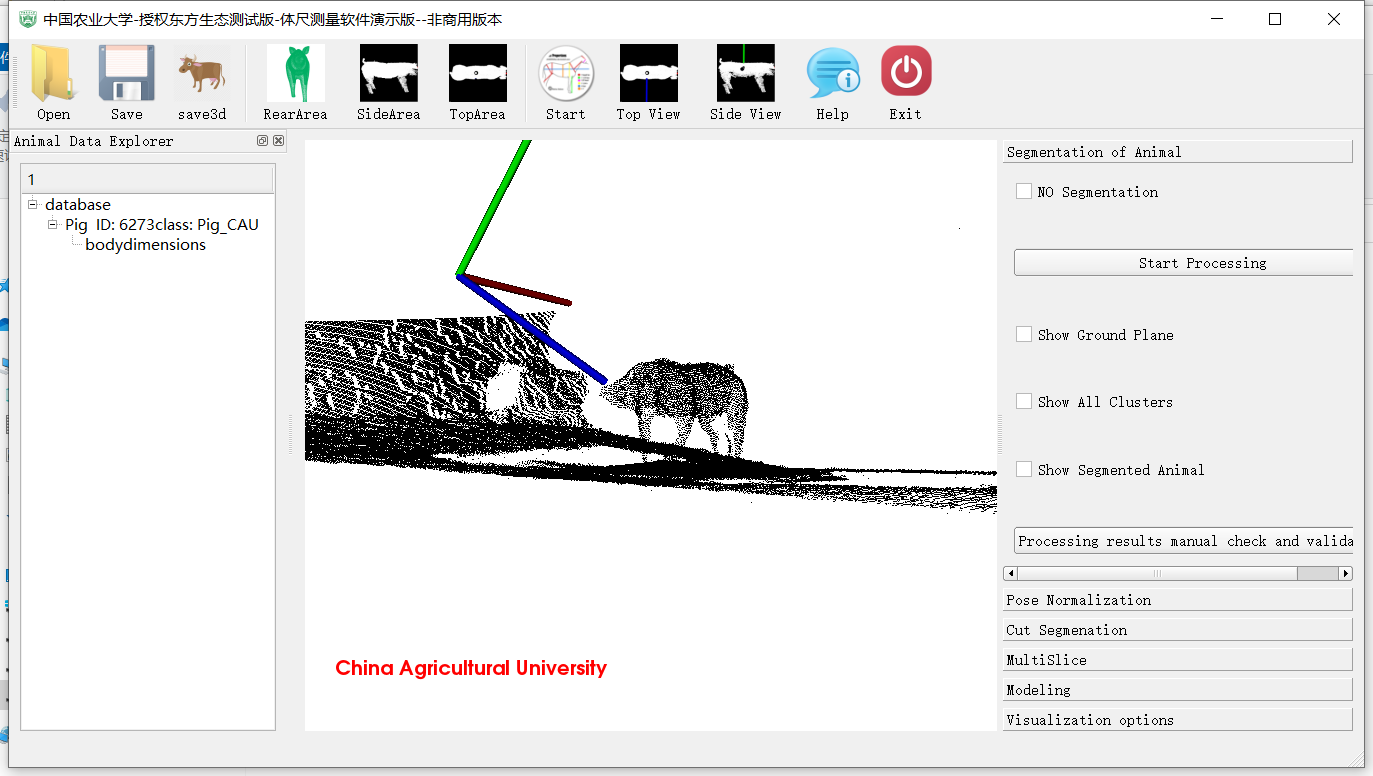
issn = "0168-1699",

doi = "https://doi.org/10.1016/j.compag.2019.03.010",

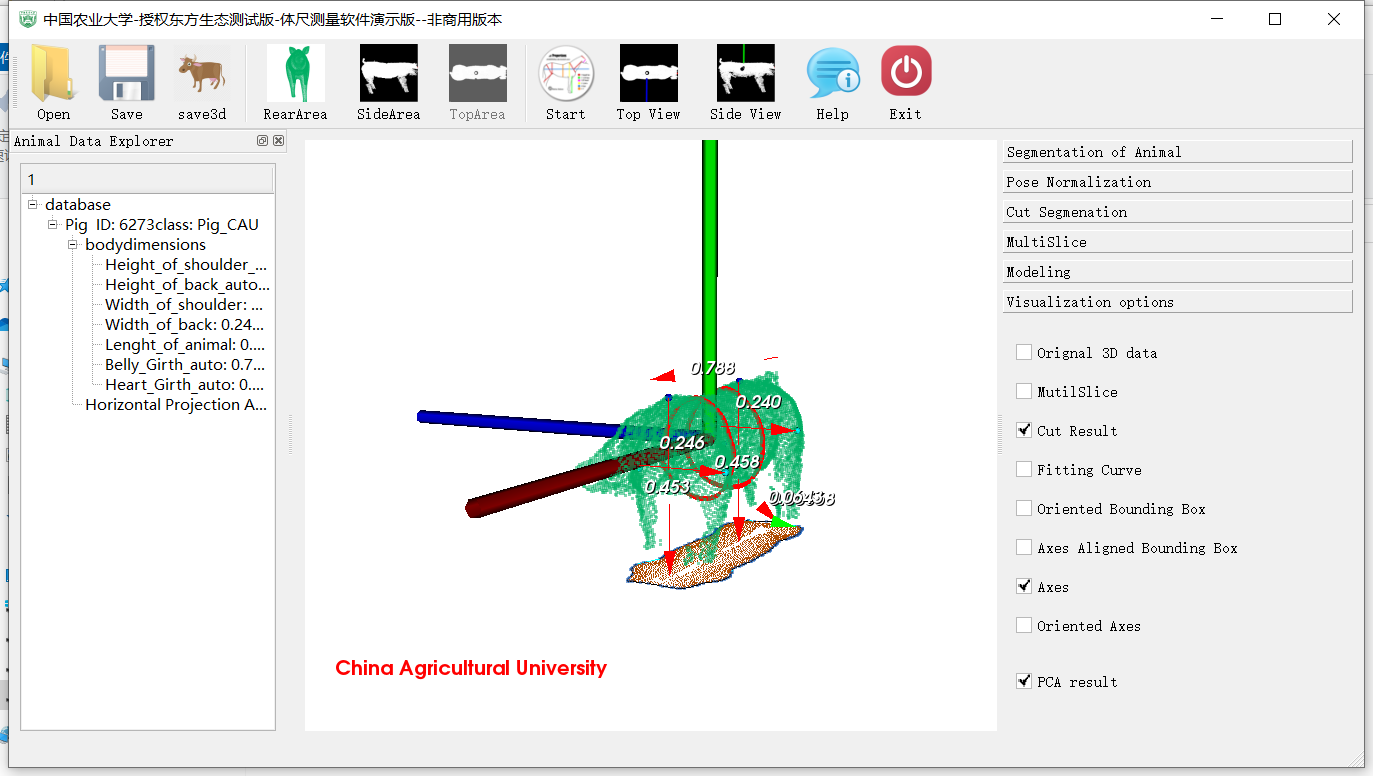
url = "http://www.sciencedirect.com/science/article/pii/S0168169918314893",

}

使用软件处理双侧点云数据，具体操作方法将在视频中给出：



处理结果：



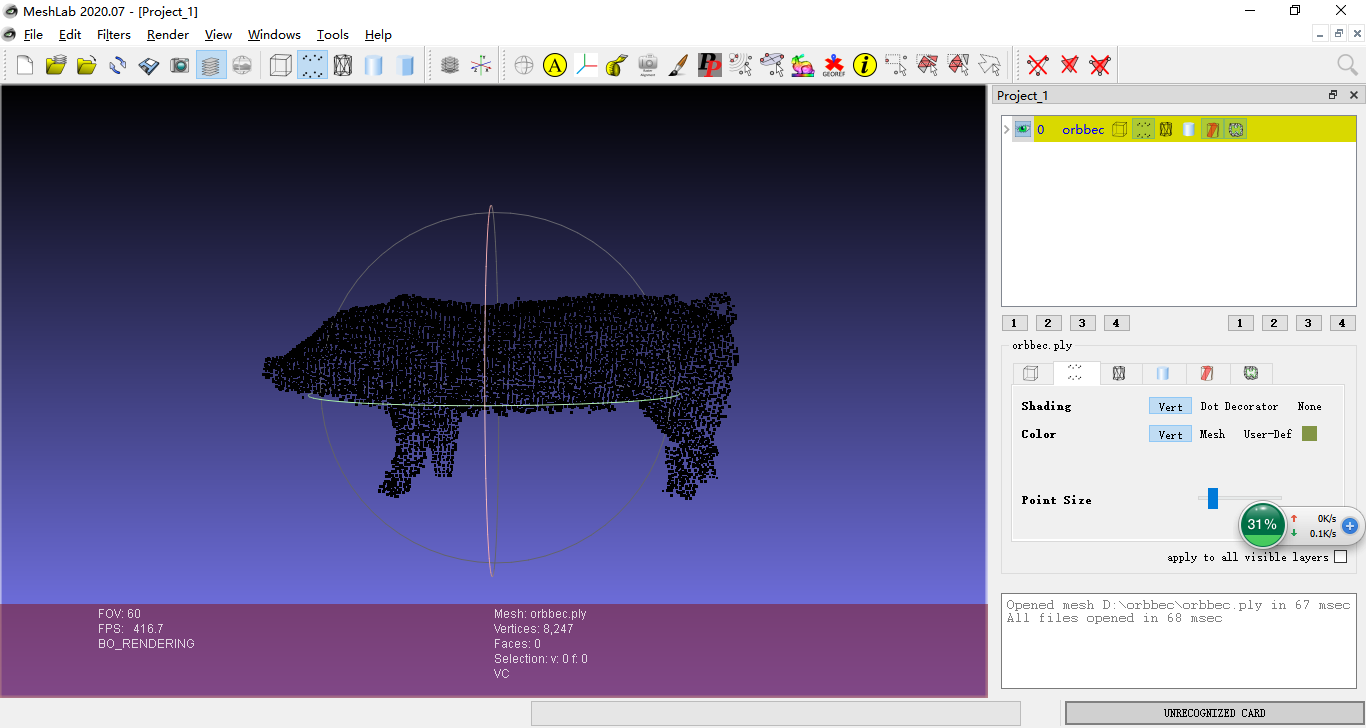
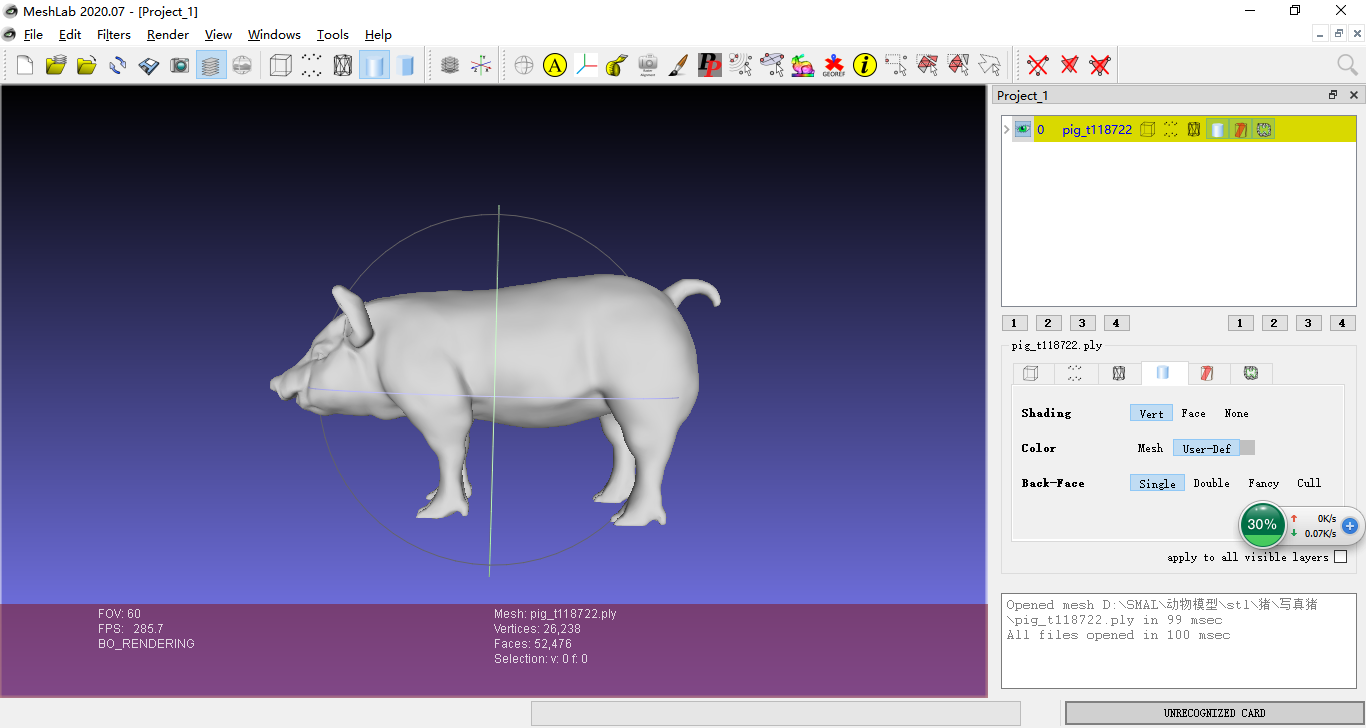
**第五部分：三维重建**

本文参考Leonid Pishchulinp论文[Building Statistical Shape Spaces for 3D Human Modeling]的方法，使用基于模板的非刚配准来实现三维重建。该方法包括初始化和非刚配准两个步骤。本项目在原有算法的基础上，通过修改其中算法参数和模型，将原有算法从人体转移到牲畜上，并达到一定的精度。

5.1 数据准备

模板数据：普通猪的网格数据

扫描数据：第四章获取的归一化后猪的点云数据



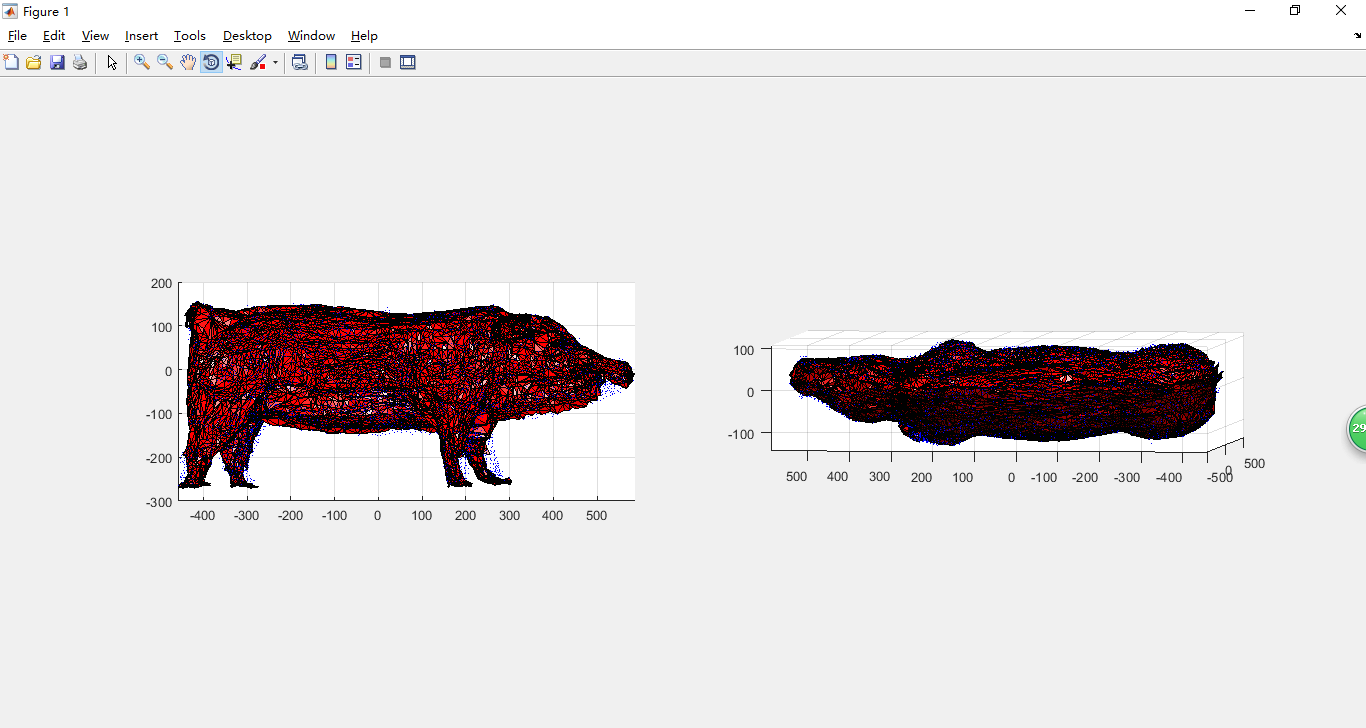
另外还需要选取一些特征点，使配准效果更好。因此，我们根据农业上猪的体尺测量，在模板数据和扫描数据上分别选取了37个对应的特征点。

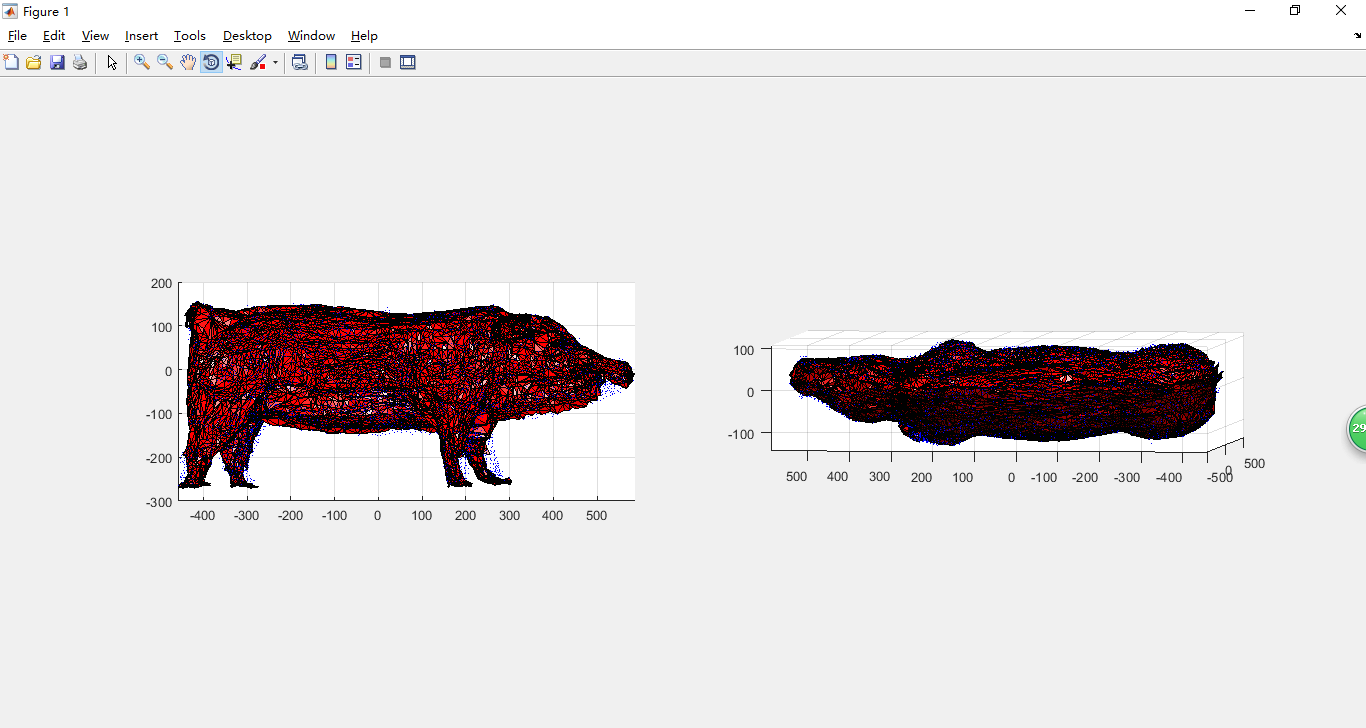
5.2 初始化

为了使非刚配准成功进行，模板数据需要和扫描数据预对齐。这里使用对应的特征点集基于Horn等人提出的单位四元数法，实现模板数据和扫描数据间的刚性配准。

5.3 非刚配准

在变换中，对于模板数据的每个点均有一个4x4的变换矩阵。计算目标即找到这一系列变换矩阵使得模板数据中的顶点尽可能地变换到扫描数据中的对应点上，更好地匹配扫描数据。其中对应点寻找基于K最邻近算法实现，并且考虑两个控制条件：距离<50mm、法线夹角<60°。





非刚配准结果

5.4 结果评价

非刚配准结果与原始点云间的比较。

（1）定量评价

|  |  |  |  |
| --- | --- | --- | --- |
|  | 平均距离误差 | 最大距离误差 | 最小距离误差 |
| 单位（mm） | 1.1497 | 23.1323 | 0.0014 |

1. 定性评价

