**图像处理MRLS\_IMGV1.0**

源代码

# 

目 录

[1.AndroidManifest.xml(主配置文件) 4](#_Toc518821288)

[2.相关类 5](#_Toc518821289)

[2.1Adapter类 5](#_Toc518821290)

[2.2Common类 10](#_Toc518821294)

[3.界面相关布局文件代码 37](#_Toc518821300)

[3.1 activity\_guide.xml 37](#_Toc518821301)

[3.2 activity\_leave\_message.xml 38](#_Toc518821302)

[3.3 activity\_login.xml 38](#_Toc518821303)

[3.4 activity\_look.xml 39](#_Toc518821304)

[3.5 activity\_main.xml 42](#_Toc518821305)

[3.6 activity\_mymsg.xml 43](#_Toc518821306)

[3.7 activity\_myreleasw.xml 44](#_Toc518821307)

[3.8 activity\_reach\_us.xml 46](#_Toc518821308)

[3.9 activity\_register.xml 46](#_Toc518821309)

[3.10 activity\_release.xml 50](#_Toc518821310)

[3.11 activity\_title.xml 55](#_Toc518821311)

[3.12 gridview\_item.xml 56](#_Toc518821312)

[3.13 info\_bottom 56](#_Toc518821313)

[3.14 info\_main.xml 58](#_Toc518821314)

[3.15 listview\_footer 61](#_Toc518821315)

[3.16 login\_activity.xml 61](#_Toc518821316)

[3.17 look\_info.xml 62](#_Toc518821317)

[3.18 message\_list\_item.xml 64](#_Toc518821318)

[3.19 my\_dialog.xml 66](#_Toc518821319)

[3.20 personal\_center.xml 68](#_Toc518821320)

[3.21 photo\_item.xml 69](#_Toc518821321)

[3.22 progress\_dialog.xml 70](#_Toc518821322)

[3.23 pulldownview.xml 70](#_Toc518821323)

[3.24 shop\_list\_item.xml 71](#_Toc518821324)

[3.25 spinner\_item.xml 72](#_Toc518821325)

[3.26 main.xml 73](#_Toc518821326)

[3.27 styles.xml 73](#_Toc518821327)

[3.28 btn\_bottom\_bg.xml 76](#_Toc518821328)

[3.29 btn\_center\_bg.xml 76](#_Toc518821329)

[3.30 btn\_first\_selector.xml 77](#_Toc518821330)

[3.31 btn\_last\_selector.xml 77](#_Toc518821331)

[3.32 btn\_login\_selector.xml 77](#_Toc518821332)

[4.JAVA源码 77](#_Toc518821333)

[4.1 BaseActivity.java 77](#_Toc518821334)

[4.2 LeaveMessageActivity.java 78](#_Toc518821335)

[4.3 LoginActivity.java 81](#_Toc518821336)

[4.4 LookActivity.java 85](#_Toc518821337)

[4.5 LookInfoActivity.java 90](#_Toc518821338)

[4.6 LookListActivity.java 93](#_Toc518821339)

[4.7 MainActivity.java 99](#_Toc518821340)

[4.8 MyCollectionActivity.java 105](#_Toc518821341)

[4.9 MyLookActivity.java 112](#_Toc518821342)

[4.10 MyMsgActivity.java 120](#_Toc518821343)

[4.11 MyreleaseActivity.java 126](#_Toc518821344)

[4.12 PersonalActivity.java 133](#_Toc518821345)

[4.13 PicturePreview.java 137](#_Toc518821346)

[4.14 echUsActivity.java 138](#_Toc518821347)

[4.15 ReleaseActivity.java 139](#_Toc518821348)

[4.16 ShopInfoActivity.java 148](#_Toc518821349)

[4.17 ShopListActivity.java 156](#_Toc518821350)

[4.18 UserRegistActivity.java 161](#_Toc518821351)

# AndroidManifest.xml(主配置文件)

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http: schemas.android.com/apk/res/android"*

package=*"com.example.dddddd"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"14"*

android:targetSdkVersion=*"14"* />

<uses-permission android:name=*"android.permission.READ\_EXTERNAL\_STORAGE"* />

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/tubiao"*

android:label=*"@string/mls\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*".MainActivity"*

android:label=*"@string/mls\_name"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

</application>

</manifest>

# 2.相关类

## 2.1JNI类

### 1. Affine\_MLS.cpp

**#include** "Wrap.h"

**void** MLSWrapPoint::**ComputeW\_**()

{

wSum = 0;

**for** (**int** i = 0; i < p.size(); ++i)

{

**if** (p[i].x == source.x && p[i].y == source.y)

wi[i] = 1 << 10;

**else**

wi[i] = 1. / ((p[i].x - source.x) \* (p[i].x - source.x) + (p[i].y - source.y) \* (p[i].y - source.y));

wSum += wi[i];

}

}

**void** MLSWrapPoint::**Compute\_p\_star\_q\_star**()

{

p\_star = Point2d(0, 0);

q\_star = Point2d(0, 0);

**for** (**int** i = 0; i < p.size(); ++i)

{

p\_star += wi[i] \* p[i];

q\_star += wi[i] \* q[i];

}

p\_star.x /= wSum;

p\_star.y /= wSum;

q\_star.x /= wSum;

q\_star.y /= wSum;

}

**void** MLSWrapPoint::**Compute\_p\_q\_Hat**()

{

**for** (**int** i = 0; i < p.size(); ++i)

{

pHat[i] = p[i] - p\_star;

qHat[i] = q[i] - q\_star;

}

}

**void** MLSWrapPoint::**ComputeTransformationParameters**(**const** vector<Point2d> &p\_, **const** vector<Point2d> &q\_, Point2d s)

{

p = p\_;

q = q\_;

source = s;

wi.resize(p.size());

pHat.resize(p.size());

qHat.resize(p.size());

ComputeW\_();

Compute\_p\_star\_q\_star();

Compute\_p\_q\_Hat();

}

Point2d MLSWrapPoint::**transMLS**()

{

Mat result;

Mat pwp = Mat::zeros(2, 2, CV\_64F);

Mat wpq = Mat::zeros(2, 2, CV\_64F);

Mat wpq\_ = Mat::zeros(2, 2, CV\_64F);

**for** (**int** i = 0; i < p.size(); ++i)

{

Mat ps = (Mat\_<**double**>(1, 2) << pHat[i].x, pHat[i].y);

Mat pst = ps.t();

pwp += pst \* (wi[i] \* ps);

}

Mat pwp\_inv = pwp.inv(CV\_LU);

**for** (**int** i = 0; i < p.size(); ++i)

{

Mat wpt = (Mat\_<**double**>(2, 1) << wi[i] \* pHat[i].x, wi[i] \* pHat[i].y);

Mat qs = (Mat\_<**double**>(1, 2) << qHat[i].x, qHat[i].y);

wpq += wpt \* qs;

}

Mat source\_p = (Mat\_<**double**>(1, 2) << source.x - p\_star.x, source.y - q\_star.y);

Mat qs = (Mat\_<**double**>(1, 2) << q\_star.x, q\_star.y);

result = source\_p \* pwp\_inv \* wpq + qs;

**return** Point2d(result.at<**double**>(0, 0), result.at<**double**>(0, 1));

}

### Android.mk

LOCAL\_PATH := $(call my-dir)

include $(CLEAR\_VARS)

OPENCV\_LIB\_TYPE:=STATIC

#logcat native代码打印

LOCAL\_LDLIBS:=-L$(SYSROOT)/usr/lib -llog

LOCAL\_LDLIBS := -ljnigraphics

#try to load OpenCV.mk from default install location

include D:\Opencv\OpenCV-android-sdk\sdk\native\jni\OpenCV.mk

LOCAL\_MODULE := ImgFun

LOCAL\_SRC\_FILES := ImgFun.cpp\

Affine\_MLS.cpp\

Rigid\_MLS.cpp\

WrapMain.cpp\

realizeMRLS.cpp

include $(BUILD\_SHARED\_LIBRARY)

### Application.ml

APP\_STL:=gnustl\_static

APP\_CPPFLAGS:=-frtti -fexceptions

APP\_ABI:= armeabi-v7a

}

### com\_example\_dddddd\_LiblmgFun.h

/\* DO NOT EDIT THIS FILE - it is machine generated \*/

**#include** <jni.h>

/\* Header for class com\_example\_dddddd\_LibImgFun \*/

**#ifndef** \_Included\_com\_example\_dddddd\_LibImgFun

**#define** \_Included\_com\_example\_dddddd\_LibImgFun

**#ifdef** \_\_cplusplus

**extern** "C" {

**#endif**

/\*

\* Class: com\_example\_dddddd\_LibImgFun

\* Method: ImgFun

\* Signature: (Landroid/graphics/Bitmap;II)I

\*/

JNIEXPORT jint JNICALL **Java\_com\_example\_dddddd\_LibImgFun\_ImgFun**

(JNIEnv \*, jclass, jobject, jint, jint);

/\*

\* Class: com\_example\_dddddd\_LibImgFun

\* Method: ImgFirst

\* Signature: ([III)[I

\*/

JNIEXPORT jintArray JNICALL **Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst**

(JNIEnv \*, jclass, jintArray, jint, jint);

/\*

\* Class: com\_example\_dddddd\_LibImgFun

\* Method: ImgMrls

\* Signature: (Landroid/graphics/Bitmap;I)I

\*/

JNIEXPORT jint JNICALL **Java\_com\_example\_dddddd\_LibImgFun\_ImgMrls**

(JNIEnv \*, jclass, jobject, jint);

/\*

\* Class: com\_example\_dddddd\_LibImgFun

\* Method: PasPoint

\* Signature: ([I[I[I[II)V

\*/

JNIEXPORT **void** JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \*, jclass, jintArray, jintArray, jintArray, jintArray, jint);

**#ifdef** \_\_cplusplus

}

**#endif**

**#endif**

### 5.Functionlast.h

**#include** "Wrap.h"

**#include** <iostream>

**int** interpolation\_width;

vector<Point2d> p, q;

**void** **Convert**(Mat& a, Mat& b)

{

**for**(**int** i = 0; i < b.rows; ++i)

**for**( **int** j = 0; j < b.cols; ++j)

{

b.at<Vec3b>(i, j) = a.at<Vec3b>(i, j);

}

}

Mat **getMat**(Mat& source, **int** i){

Mat test = cvLoadImage("rigid.jpg", 1),map\_x, map\_y;

GaussianBlur(source, source, Size( 5, 5 ), 100, 100 );

Mat test(source);

p.push\_back(Point2d(0, 0));

p.push\_back(Point2d(100, 0));

p.push\_back(Point2d(100, 100));

p.push\_back(Point2d(100, 200));

p.push\_back(Point2d(test.rows - 1, test.cols - 1));

q.push\_back(Point2d(0, 0));

q.push\_back(Point2d(100, 0));

q.push\_back(Point2d(80, 100));

q.push\_back(Point2d(100, 200));

q.push\_back(Point2d(test.rows - 20, test.cols - 1));

Mat test\_affine = Mat(test.size(),test.type(), Scalar(100, 100, 100));

Mat test\_rigid = Mat(test.size(), test.type(), Scalar(100, 100, 100));

Mat test\_mrls = Mat(test.size(), test.type(), Scalar(100, 100, 100));

test\_.create(test.size(), test.type());

wrap(test, test\_, p, q);

MLS wp;

wp.InitBeforeComputation(q, p, source, interpolation\_width);

**switch**(i)

{

**case** 1: test\_affine = wp.WarpImage(1); Convert(test\_affine, source); **break**;

**case** 2: test\_rigid = wp.WarpImage(2); Convert(test\_rigid, source); **break**;

**case** 3:

Mat x,y,v,TX,TY,sfv,dxt,dyt,ifxt,ifyt;

Mat dst,tx,ty;

Mat src1=source;

MRLS mrls;

mrls.src=src1;

**for**(**int** i = 0; i < p.size(); ++i){

**double** tp, tq;

tp = p[i].x;

p[i].x = p[i].y;

p[i].y = tp;

tq = q[i].x;

q[i].x = q[i].y;

q[i].y = tq;

}

mrls.p = p;

mrls.q = q;

mrls.q.push\_back(Point2f(130, 182));

mrls.q.push\_back(Point2f(96, 201));

mrls.q.push\_back(Point2f(168, 201));

mrls.p.push\_back(Point2f(132, 168));

mrls.p.push\_back(Point2f(85, 206));

mrls.p.push\_back(Point2f(176, 210));

mrls.mymean(mrls.q,mrls.p);

mrls.std1();

mrls.gener(mrls.src,x,y); 生成步长为十的网格点并进行归一化处理

mrls.Genergrid(x,y,v); 生成m\*n乘2的矩阵V

mrls.Genercompletegrid(mrls.src,TX,TY,tx,ty);

mrls.warp(v,mrls.q1,sfv);

mrls.Computdisplace(sfv,v,x,mrls.src,dxt,dyt);

mrls.Computnewpoints(TX,TY,dxt,dyt,ifxt,ifyt);

dst=mrls.gen\_newimage(src1,tx,ty,ifxt,ifyt);

Convert(dst, source); **break**;

default: break;

}

**return** source;

}

### 6.ImaFun.cpp

#include <jni.h>

#include <stdio.h>

#include <stdlib.h>

#include <opencv2/opencv.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <android/bitmap.h>

#include <android/log.h>

#include <android/log.h>

#include "Functionlast.h"

#define LOG\_TAG "libplasma"

#define TAG "myDemo-jni" 这个是自定义的LOG的标识

#define LOGD(...) \_\_android\_log\_print(ANDROID\_LOG\_DEBUG,TAG ,\_\_VA\_ARGS\_\_) 定义LOGD类型

#define LOGI(...) \_\_android\_log\_print(ANDROID\_LOG\_INFO,TAG ,\_\_VA\_ARGS\_\_) 定义LOGI类型

#define LOGW(...) \_\_android\_log\_print(ANDROID\_LOG\_WARN,TAG ,\_\_VA\_ARGS\_\_) 定义LOGW类型

#define LOGE(...) \_\_android\_log\_print(ANDROID\_LOG\_ERROR,TAG ,\_\_VA\_ARGS\_\_) 定义LOGE类型

#define LOGF(...) \_\_android\_log\_print(ANDROID\_LOG\_FATAL,TAG ,\_\_VA\_ARGS\_\_) 定义LOGF类型

Mat **getMat**(Mat& source, **int** i){

Mat test = cvLoadImage("rigid.jpg", 1),map\_x, map\_y;

GaussianBlur(source, source, Size( 5, 5 ), 100, 100 );

Mat test(source);

p.push\_back(Point2d(0, 0));

p.push\_back(Point2d(100, 0));

p.push\_back(Point2d(100, 100));

p.push\_back(Point2d(100, 200));

p.push\_back(Point2d(test.rows - 1, test.cols - 1));

q.push\_back(Point2d(0, 0));

q.push\_back(Point2d(100, 0));

q.push\_back(Point2d(80, 100));

q.push\_back(Point2d(100, 200));

q.push\_back(Point2d(test.rows - 20, test.cols - 1));

Mat test\_affine = Mat(test.size(),test.type(), Scalar(100, 100, 100));

Mat test\_rigid = Mat(test.size(), test.type(), Scalar(100, 100, 100));

Mat test\_mrls = Mat(test.size(), test.type(), Scalar(100, 100, 100));

test\_.create(test.size(), test.type());

wrap(test, test\_, p, q);

MLS wp;

wp.InitBeforeComputation(q, p, source, interpolation\_width);

**switch**(i)

{

**case** 1: test\_affine = wp.WarpImage(1); Convert(test\_affine, source); **break**;

**case** 2: test\_rigid = wp.WarpImage(2); Convert(test\_rigid, source); **break**;

**case** 3:

Mat x,y,v,TX,TY,sfv,dxt,dyt,ifxt,ifyt;

Mat dst,tx,ty;

Mat src1=source;

MRLS mrls;

mrls.src=src1;

**for**(**int** i = 0; i < p.size(); ++i){

**double** tp, tq;

tp = p[i].x;

p[i].x = p[i].y;

p[i].y = tp;

tq = q[i].x;

q[i].x = q[i].y;

q[i].y = tq;

}

mrls.p = p;

mrls.q = q;

mrls.q.push\_back(Point2f(130, 182));

mrls.q.push\_back(Point2f(96, 201));

mrls.q.push\_back(Point2f(168, 201));

mrls.p.push\_back(Point2f(132, 168));

mrls.p.push\_back(Point2f(85, 206));

mrls.p.push\_back(Point2f(176, 210));

mrls.mymean(mrls.q,mrls.p);

mrls.std1();

mrls.gener(mrls.src,x,y); 生成步长为十的网格点并进行归一化处理

mrls.Genergrid(x,y,v); 生成m\*n乘2的矩阵V

mrls.Genercompletegrid(mrls.src,TX,TY,tx,ty);

mrls.warp(v,mrls.q1,sfv);

mrls.Computdisplace(sfv,v,x,mrls.src,dxt,dyt);

mrls.Computnewpoints(TX,TY,dxt,dyt,ifxt,ifyt);

dst=mrls.gen\_newimage(src1,tx,ty,ifxt,ifyt);

Convert(dst, source); **break**;

default: break;

}

**return** source;

}

using namespace cv;

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

### 7.realizeMRLS.cpp

#include"wrap.h"

void MRLS::mymean(const vector<Point2d> & q, const vector<Point2d> & p)

{

int s = q.size();

double qsumx = 0, qsumy = 0, psumx = 0, psumy = 0;

for (int i = 0; i < s; ++i)

{

qsumx = q[i].x + qsumx;

qsumy = q[i].y + qsumy;

psumx = p[i].x + psumx;

psumy = p[i].y + psumy;

}

qsumx = qsumx / s; psumx = psumx / s;

qsumy = qsumy / s; psumy = psumy / s;

resultq.push\_back(Point2d(qsumx, qsumy));

resultp.push\_back(Point2d(psumx, psumy));

Mat e = Mat::zeros(s, 2, CV\_64F);

Mat f = Mat::zeros(s, 2, CV\_64F);

Mat x = Mat::zeros(s, 2, CV\_64F);

Mat y = Mat::zeros(s, 2, CV\_64F);

for (int i = 1; i <= s; i++)

{

x.row(i - 1).col(0) = q[i - 1].x;

y.row(i - 1).col(0) = p[i - 1].x;

x.row(i - 1).col(1) = q[i - 1].y;

y.row(i - 1).col(1) = p[i - 1].y;

e.row(i - 1).col(0) = resultq[0].x;

e.row(i - 1).col(1) = resultq[0].y;

f.row(i - 1).col(0) = resultp[0].x;

f.row(i - 1).col(1) = resultp[0].y;

}

q1 = x - e;

p1 = y - f;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

void MRLS::std1()

{

float xstd = 0, ystd = 0;

vector<Point2d>q2; vector<Point2d>p2;

int s = q.size();

for (int i = 0; i<s; i++)

{

q2.push\_back(Point2d(q[i].x - resultq[0].x, q[i].y - resultq[0].y));

p2.push\_back(Point2d(p[i].x - resultp[0].x, p[i].y - resultp[0].y));

}

for (int i = 0; i < s; i++)

{

xstd = q2[i].x\*q2[i].x + q2[i].y\*q2[i].y + xstd;

ystd = p2[i].x\*p2[i].x + p2[i].y\*p2[i].y + ystd;

}

xscale = sqrt(xstd / s);

yscale = sqrt(ystd / s);

q1 = q1 / xscale;

p1 = p1 / yscale;

}

void MRLS::gener(Mat X, Mat &x, Mat &y)//生成网格

{

int m = ceil((X.rows) / 10 + 0.5);

int n = ceil((X.cols) / 10 + 0.5);

x = Mat::zeros(m, n, CV\_64F);

y = Mat::zeros(m, n, CV\_64F);

for (int j = 0; j<m; j++)

for (int i = 0; i<n; i++)

{

x.row(j).col(i) = 10 \* i + 1;

y.row(j) = 10 \* j + 1;

}

Mat X1 = Mat::zeros(m, n, CV\_64F);

Mat Y1 = Mat::zeros(m, n, CV\_64F);

for (int j = 0; j<m; j++)

for (int i = 0; i<n; i++)

{

X1.row(j).col(i) = resultq[0].x;

Y1.row(j).col(i) = resultq[0].y;

}

X1 = (x - X1) / xscale;

Y1 = (y - Y1) / xscale;

x = X1;

y = Y1;

}

//Generating the grid:

void MRLS::Genergrid(Mat X, Mat Y, Mat &v)

{

int n = X.rows, m = X.cols;

v = Mat::zeros(n\*m, 2, CV\_64F);

int k = 0;

{for (int i = 0; i<m; i++)

for (int j = 0; j<n; j++)

{

v.at<double>(k, 0) = X.at<double>(j, i);

v.at<double>(k, 1) = Y.at<double>(j, i);

k++;

}

}

}

void MRLS::Genercompletegrid(Mat img, Mat &TX, Mat &TY, Mat &tx, Mat &ty)

{

int w = img.cols;

int h = img.rows;

TX = Mat::zeros(h, w, CV\_64F);

TY = Mat::zeros(h, w, CV\_64F);

tx = Mat::zeros(h, w, CV\_64F);

ty = Mat::zeros(h, w, CV\_64F);

for (double j = 0; j<h; j++)

for (double i = 0; i<w; i++)

{

tx.row(j).col(i) = i + 1;

ty.row(j) = j + 1;

}

Mat xm1 = Mat::zeros(h, w, CV\_64F);

Mat xm2 = Mat::zeros(h, w, CV\_64F);

for (int j = 0; j<h; j++)

for (int i = 0; i<w; i++)

{

xm1.row(j).col(i) = resultq[0].x;/////////////warning!!!!

xm2.col(i).row(j) = resultq[0].y;

}

TX = (tx - xm1) / xscale;

TY = (ty - xm2) / xscale;

}

void MRLS::conk(Mat p, Mat q, Mat &K)

{

int n = p.rows;

int m = q.rows;

K = Mat::zeros(n, m, CV\_64F);

for (int i = 0; i<m; i++)

for (int j = 0; j<n; j++)

{

K.at<double>(j, i) = (p.at<double>(j, 0) - q.at<double>(i, 0))\*(p.at<double>(j, 0) - q.at<double>(i, 0)) + (p.at<double>(j, 1) - q.at<double>(i, 1))\*(p.at<double>(j, 1) - q.at<double>(i, 1));

}

for (int i = 0; i<m; i++)

for (int j = 0; j<n; j++)

{

K.at<double>(j, i) = exp((-1 \* K.at<double>(j, i)) / pow(5, 2));

}

}

void MRLS::computeW(Mat &v, Mat p, Mat& W\_1)

{

int row1 = p.rows;

Mat a = Mat::zeros(1, 2, CV\_64F);

Mat b = Mat::zeros(row1, 2, CV\_64F);

int i = v.rows;

for (int m = 0; m<i; m++)

for (int j = 0; j<row1; j++)

{

b.at<double>(j, 0) = abs(q1.at<double>(j, 0) - v.at<double>(m, 0));

b.at<double>(j, 1) = abs(q1.at<double>(j, 1) - v.at<double>(m, 1));

a.at<double>(0, 0) = pow(b.at<double>(j, 0), 2) + a.at<double>(0, 0);

a.at<double>(0, 1) = pow(b.at<double>(j, 1), 2) + a.at<double>(0, 1);

}

W\_1 = Mat::zeros(2, 2, CV\_64F);

for (int j = 0; j<2; j++)

{

W\_1.at<double>(j, j) = a.at<double>(0, j);

}

}

void MRLS::warp(Mat &v, Mat p, Mat& sfv)

{

int row1 = p.rows;

int i = v.rows;

Mat a = Mat::zeros(row1, 1, CV\_64F);

Mat b = Mat::zeros(row1, 2, CV\_64F);

Mat W\_1 = Mat::zeros(row1, row1, CV\_64F);

sfv = Mat::zeros(i, 2, CV\_64F);

Mat k1, k2;

conk(p1, p1, k1);

conk(v, p1, k2);

for (int m = 0; m<i; m++)

{

for (int j = 0; j<row1; j++)

{

b.at<double>(j, 0) = abs(p.at<double>(j, 0) - v.at<double>(m, 0));

b.at<double>(j, 1) = abs(p.at<double>(j, 1) - v.at<double>(m, 1));

W\_1.at<double>(j, j) = pow(b.at<double>(j, 0), 2) + pow(b.at<double>(j, 1), 2);

}

sfv.row(m) = v.row(m) + k2.row(m)\*(k1 + 10 \* W\_1).inv()\*(q1 - p1);

}

}

void MRLS::Computdisplace(Mat sfv, Mat v, Mat size1, Mat size2, Mat &dxt, Mat&dyt)

{

int k = 0;

Mat dxy = sfv - v;

Mat reshape1 = Mat::zeros(size1.rows, size1.cols, CV\_64F);

Mat reshape2 = Mat::zeros(size1.rows, size1.cols, CV\_64F);

dxt.create(src.size(), src.type());

dyt.create(src.size(), src.type());

for (int i = 0; i<size1.cols; i++)

for (int j = 0; j<size1.rows; j++)

{

reshape1.at<double>(j, i) = dxy.at<double>(k, 0);

reshape2.at<double>(j, i) = dxy.at<double>(k, 1);

k++;

}

resize(reshape1, dxt, Size(size2.cols, size2.rows), 0, 0);

resize(reshape2, dyt, Size(size2.cols, size2.rows), 0, 0);

}

void MRLS::Computnewpoints(Mat &ntx, Mat &nty, Mat &dxt, Mat &dyt, Mat &ifxt, Mat &ifyt)

{

ifxt = (ntx + dxt)\*xscale + resultq[0].x; /////warning

ifyt = (nty + dyt)\*xscale + resultq[0].y;

}

Mat MRLS::gen\_newimage(Mat src, Mat &TX, Mat &TY, Mat &ifxt, Mat &ifyt)

{

Mat dst;

//src.copyTo(dst);

dst.create(src.size(),src.type());

TX.convertTo(TX, CV\_32FC1);

TY.convertTo(TY, CV\_32FC1);

ifxt.convertTo(ifxt, CV\_32FC1);

ifyt.convertTo(ifyt, CV\_32FC1);

Mat map\_x, map\_y;

map\_x.create(src.size(), CV\_32FC1);

map\_y.create(src.size(), CV\_32FC1);;

for (int i = 0; i<map\_x.rows; i++)

for (int j = 0; j<map\_x.cols; j++)

{

map\_x.at<float>(i, j) = (2 \* TX.at<float>(i, j) - ifxt.at<float>(i, j));

map\_y.at<float>(i, j) = (2 \* TY.at<float>(i, j) - ifyt.at<float>(i, j));

//if (map\_x.at<float>(i, j)>map\_x.cols) map\_x.at<float>(i, j) = map\_x.cols;

//if (map\_y.at<float>(i, j)>map\_x.rows) map\_y.at<float>(i, j) = map\_x.rows;

//dst.at<Vec3b>((map\_y.at<float>(i, j) - 0.000001), (map\_x.at<float>(i, j) - 0.000001))[0] = src.at<Vec3b>(i, j)[0];//B

//dst.at<Vec3b>((map\_y.at<float>(i, j) - 0.000001), (map\_x.at<float>(i, j) - 0.000001))[1] = src.at<Vec3b>(i, j)[1];//G

//dst.at<Vec3b>((map\_y.at<float>(i, j) - 0.000001), (map\_x.at<float>(i, j) - 0.000001))[2] = src.at<Vec3b>(i, j)[2];//R

}

remap(src, dst, map\_x, map\_y,INTER\_LINEAR, BORDER\_CONSTANT, Scalar(0,0,0));

return dst;

}

### 8.Rigid\_MLS.cpp

#include "Wrap.h"

void Rigid\_MLS::Compute\_Ai()

{

for (int i = 0; i < p.size(); ++i)

{

Mat a = (Mat\_<double>(2, 2) << MLSWrapPoint::pHat[i].x, MLSWrapPoint::pHat[i].y,

MLSWrapPoint::pHat[i].y, -MLSWrapPoint::pHat[i].x);

Mat b = (Mat\_<double>(2, 2) << MLSWrapPoint::source.x - MLSWrapPoint::p\_star.x, MLSWrapPoint::source.y - MLSWrapPoint::p\_star.y,

MLSWrapPoint::source.y - MLSWrapPoint::p\_star.y, MLSWrapPoint::p\_star.x - MLSWrapPoint::source.x);

// Compute the Ai

Ai.push\_back(wi[i] \* a \* b.t());

}

}

void Rigid\_MLS::ComputeTransformationParameters(const vector<Point2d> &p\_, const vector<Point2d> &q\_, Point2d s)

{

MLSWrapPoint::ComputeTransformationParameters(p\_, q\_, s);

this->Compute\_Ai();

}

Point2d Rigid\_MLS::transMLS()

{

Mat result;

Mat frv = Mat::zeros(1, 2, CV\_64F);

for (int i = 0; i < p.size(); ++i)

{

Mat qHati = (Mat\_<double>(1, 2) << MLSWrapPoint::qHat[i].x, MLSWrapPoint::qHat[i].y);

frv += qHati \* this->Ai[i];

}

//以下两行与c++代码有出入，是语法原因

double v\_ps = sqrt((MLSWrapPoint::source.x - p\_star.x) \* (MLSWrapPoint::source.x - p\_star.x) + (MLSWrapPoint::source.y - MLSWrapPoint::p\_star.y) \* (MLSWrapPoint::source.y - MLSWrapPoint::p\_star.y));

double dfrv = sqrt(frv.at<double>(0, 0) \* frv.at<double>(0, 0) + frv.at<double>(0, 1) \* frv.at<double>(0, 1));

Mat qs = (Mat\_<double>(1, 2) << MLSWrapPoint::q\_star.x, MLSWrapPoint::q\_star.y);

result = v\_ps \* frv / dfrv + qs;

return Point2d(result.at<double>(0, 0), result.at<double>(0, 1));

}

### 9.Wrap.h

#pragma once

#ifndef WRAP

#define WRAP

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <vector>

#include <iostream>

using namespace cv;

using namespace std;

class MLSWrapPoint //for every point compute the transformation point

{

protected:

vector<Point2d> p;

vector<Point2d> q;

Point2d source;

Point2d p\_star, q\_star;

vector<Point2d> pHat;

vector<Point2d> qHat;

vector<double>wi;

double wSum;

void ComputeW\_();

void Compute\_p\_star\_q\_star();

void Compute\_p\_q\_Hat();

public:

void ComputeTransformationParameters(const vector<Point2d> &p\_, const vector<Point2d> &q\_, Point2d s);

Point2d transMLS();

};

class MLS //create grid and interpolation

{

private:

int nPoint;

vector<Point2d> p;

vector<Point2d> q;

int ImgH;

int ImgW;

vector<int> IndexX;

vector<int> IndexY;

vector<bool> vXCompute;

vector<bool> vYCompute;

Mat source;

int choose; // algorithm choose

void SetXYCompute(int StepSize);

vector<int> NewIntIndexArray(int l);

vector<Point2d> ComputeVAndInterpolateXYRow(int iy);

vector<Point2d> InterpolateXYRow(int iy, int iy1, int iy2, const vector<Point2d> & xyRow1, const vector<Point2d> &xyRow2);

Vec3b BilinearInterpolation(Point2d org);

public:

void InitBeforeComputation(const vector<Point2d>& p, const vector<Point2d> &q, const Mat &source, int stepSize = 1);

Mat WarpImage(int i);

};

class Rigid\_MLS : public MLSWrapPoint

{

private:

vector<Mat> Ai;

void Compute\_Ai();

public:

void ComputeTransformationParameters(const vector<Point2d> &p\_, const vector<Point2d> &q\_, Point2d s);

Point2d transMLS();

};

class MRLS

{

public:

vector<Point2d>p;

vector<Point2d>q;

Mat src;

Mat q1;//进行均值化后的最终矩阵q

Mat p1;//进行均值化后的最终矩阵p

vector<Point2d> resultq;//填入q(第一列的平均值，第二列的平均值)

vector<Point2d> resultp;//填入p(第一列的平均值，第二列的平均值)

float xscale, yscale;

void on\_mouse1(int event, int x, int y, int flags, void\* ustc);

void on\_mouse2(int event, int x, int y, int flags, void\* ustc);

void mymean(const vector<Point2d> & q, const vector<Point2d> & p);

void std1();

void gener(Mat X, Mat &x, Mat &y);//生成网格

void warp(Mat &v, Mat p, Mat& sfv);

void conk(Mat p, Mat q, Mat &K);

void computeW(Mat &v, Mat p, Mat& W\_1);

void Genergrid(Mat X, Mat Y, Mat &v);

void Genercompletegrid(Mat img, Mat &TX, Mat &TY, Mat &tx, Mat &ty);

void Computdisplace(Mat sfv, Mat v, Mat size1, Mat size2, Mat &dxt, Mat&dyt);

void Computnewpoints(Mat &ntx, Mat &nty, Mat &dxt, Mat &dyt, Mat &ifxt, Mat &ifyt);

Mat gen\_newimage(Mat src, Mat &TX, Mat &TY, Mat &ifxt, Mat &ifyt);

};

#endif

### 10.WrapMain.cpp

#include "Wrap.h"

void MLS::InitBeforeComputation(const vector<Point2d>& p, const vector<Point2d> &q, const Mat &src, int stepSize)

{

source = src.clone();

ImgH = source.rows;

ImgW = source.cols;

this->p = p;

this->q = q;

IndexX = NewIntIndexArray(ImgW);

IndexY = NewIntIndexArray(ImgH);

SetXYCompute(stepSize);

}

void MLS::SetXYCompute(int stepSize)

{

vXCompute.resize(ImgW);

for(int i = 0; i < ImgW; ++i) vXCompute[i] = false;

for (int i = 0; i < ImgW; i += stepSize) vXCompute[i] = true;

vXCompute[ImgW - 1] = true;

vYCompute.resize(ImgH);

for(int i = 0; i < ImgH; ++i) vYCompute[i] = false;

for (int i = 0; i < ImgH; i += stepSize) vYCompute[i] = true;

vYCompute[ImgH - 1] = true;

}

vector<int> MLS::NewIntIndexArray(int l)

{

vector<int> result(l);

for (int i = 0; i < l; ++i) result[i] = i;

return result;

}

// interpolate a width line

vector<Point2d> MLS::InterpolateXYRow(int iy, int iy1, int iy2, const vector<Point2d> & xyRow1, const vector<Point2d> &xyRow2)

{

vector<Point2d> result(ImgW);

double delta = (double)(iy - iy1) / (double)(iy2 - iy1);

for (int i = 0; i < ImgW; ++i)

{

double RInterpol = xyRow1[i].x \* (1.0 - delta) + xyRow2[i].x \* delta;

double CInterpol = xyRow1[i].y \* (1.0 - delta) + xyRow2[i].y \* delta;

result[i] = Point2d(RInterpol, CInterpol);

}

return result;

}

Vec3b MLS::BilinearInterpolation(Point2d org)

{

Vec3b pixl(100, 100, 100);

int floorX = org.x;//(int)(org.x + 0.5);

if ((floorX < 0) || (floorX >= ImgW)) return pixl;

int floorY = org.y;//(int)(org.y + 0.5);

if ((floorY < 0) || (floorY) >= ImgH) return pixl;

// near pixl

int ceilX = floorX + 1;

int ceilY = floorY + 1;

if (ceilX >= ImgW || ceilY >= ImgH) return pixl;

Vec3b pix1 = source.at<Vec3b>(floorY, floorX);

Vec3b pix2 = source.at<Vec3b>(floorY, ceilX);

Vec3b pix3 = source.at<Vec3b>(ceilY, floorX);

Vec3b pix4 = source.at<Vec3b>(ceilY, ceilX);

double deltaX = org.x - (double)floorX;

double deltaY = org.y - (double)floorY;

double cx1, cx2, c; // chanel blue;

cx1 = (1 - deltaX) \* pix1[0] + deltaX \* pix2[0];

cx2 = (1 - deltaX) \* pix3[0] + deltaX \* pix2[0];

c = (1 - deltaY) \* cx1 + deltaY \* cx2;

pixl[0] = c;

// chanel green

cx1 = (1 - deltaX) \* pix1[1] + deltaX \* pix2[1];

cx2 = (1 - deltaX) \* pix3[1] + deltaX \* pix2[1];

c = (1 - deltaY) \* cx1 + deltaY \* cx2;

pixl[1] = c;

// chanel red

cx1 = (1 - deltaX) \* pix1[2] + deltaX \* pix2[2];

cx2 = (1 - deltaX) \* pix3[2] + deltaX \* pix2[2];

c = (1 - deltaY) \* cx1 + deltaY \* cx2;

pixl[2] = c;

return pixl;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

vector<Point2d> MLS::ComputeVAndInterpolateXYRow(int iy)

{

vector<Point2d> result(ImgW);

int ix1 = 0;

int ix2 = 0;

Point2d vx1;

Point2d vx2;

// compute points V specified by vXcompute

for (int i = 0; i < ImgW; ++i) if (vXCompute[i])

{

Point2d pnt = Point2d(iy, i);

for (int j = 0; j < p.size(); ++j)

{

}

// Compute parameters using class MLSWrapPoint

// Backtransformation by switching parameters p, q

MLSWrapPoint v1;

Rigid\_MLS v2;

switch (choose)

{

case 1:

v1.ComputeTransformationParameters(p, q, pnt);

result[i] = v1.transMLS(); break;

case 2:

v2.ComputeTransformationParameters(p, q, pnt);

result[i] = v2.transMLS(); break;

default:

break;

}

}

// Interplolate points ix between coordinates computed by v i values:ix1 and ix2

vx2 = result[0];

for (int i = 0; i < ImgW; ++i)

{

if (vXCompute[i])

{

//current ix2 reached, ix1 <- ix2

ix1 = ix2;

vx1 = vx2;

bool X2Found = false;

ix2 = i;

while (!X2Found)

{

ix2++;

if ((ix2 >= ImgW) || vXCompute[ix2]) X2Found = true;

}

if (ix2 < ImgW) vx2 = result[ix2];

}

else // interpolate between computed corrdnates vx1 and vx2

{

double deleta = (double)(i - ix1) / (double)(ix2 - ix1);

double CInterpol = vx1.y \* (1 - deleta) + vx2.y \* deleta; // x direction deleta

double RInterpol = vx1.x \* (1 - deleta) + vx2.x \* deleta; // y direction deleta

result[i] = Point2d(RInterpol, CInterpol);

}

}

return result;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

Mat MLS::WarpImage(int i)

{

choose = i;

Mat result(ImgH, ImgW, CV\_8UC3, Scalar(100, 100, 100));

vector<Point2d> xyRow1(ImgW), xyRow2(ImgW), xyRow(ImgW);

xyRow2 = ComputeVAndInterpolateXYRow(0);

int iyRow1 = 0;

int iyRow2 = 0;

for (int iy = 0; iy < ImgH; ++iy)

{

if (vYCompute[iy])

{

//std::cout << iy<< endl;

iyRow1 = iyRow2;

for (int i = 0; i < ImgW; ++i) xyRow1[i] = xyRow2[i];

// find and set next Row2

bool Row2Found = false;

while (!Row2Found)

{

iyRow2++;

if ((iyRow2 >= ImgH) || vYCompute[iyRow2]) Row2Found = true;

}

// compute a new Row

// (y, x)coordinates are computed using MLSWrapPoint where xComputeV[i] true;

// rest of row is interpolated

if (iyRow2 < ImgH) xyRow2 = ComputeVAndInterpolateXYRow(iyRow2);

}

// Interpolate the current xyRow using known xyRow1 and xyRow2

xyRow = InterpolateXYRow(iy, iyRow1, iyRow2, xyRow1, xyRow2);

// Comupte the pixle values at Points of original value

for (int ix = 0; ix < ImgW; ++ix)

{

Point2d coorOrg = xyRow[ix];

result.at<Vec3b>(iy, ix) = BilinearInterpolation(Point2d(coorOrg.y, coorOrg.x));

/\*int r = coorOrg.x;

int c = coorOrg.y;

if (c < ImgW && c >= 0 && r >= 0 && r < ImgH)

result.at<Vec3b>(iy, ix) = source.at<Vec3b>(r, c);\*/

}

}

return result;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C"{

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv \* env, jclass obj, jobject bmpIn, jint cho, jint inter\_width) {

interpolation\_width = 10;

interpolation\_width = inter\_width;

AndroidBitmapInfo inBmpInfo;

void\* inPixelsAddress;

int ret;

if ((ret = AndroidBitmap\_getInfo(env, bmpIn, &inBmpInfo)) < 0) {

LOGD("AndroidBitmap\_getInfo() failed ! error=%d", ret);

return;

}

LOGI("original image :: width is %d; height is %d; stride is %d; format is %d;flags is %d,stride is %u", inBmpInfo.width, inBmpInfo.height, inBmpInfo.stride, inBmpInfo.format, inBmpInfo.flags, inBmpInfo.stride);

if ((ret = AndroidBitmap\_lockPixels(env, bmpIn, &inPixelsAddress)) < 0) {

LOGE("AndroidBitmap\_lockPixels() failed ! error=%d", ret);

}

Mat inMat(inBmpInfo.height, inBmpInfo.width,

CV\_8UC4, inPixelsAddress);

Mat temp = inMat.clone();

cvtColor(inMat,temp,CV\_BGRA2BGR);

GaussianBlur(inMat, inMat, Size( 5, 5 ), 100, 100 );

getMat(temp, cho);

cvtColor( temp,inMat, CV\_BGR2BGRA );

Sobel(inMat, inMat, inMat.depth(), 1, 1);

AndroidBitmap\_unlockPixels(env, bmpIn);

LOGI("Return !! ");

return;

/\*

\* 之后传入的方法

\*/

}

另外一种写法

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray, jint, jint);

JNIEXPORT jintArray JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFirst

(JNIEnv \* env, jclass, jintArray buf, jint w, jint h){

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, false);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage=cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

cvCanny(image3channel,pCannyImage,50,150,3);

用于图像的处理

Mat inMat(image3channel);

getMat(inMat, 1);

转换为IplImage

IplImage result\_img = IplImage(inMat);

int\* outImage=new int[w\*h];

int x = 0;

for(int i=0;i<w\*h;i++)

{

x = inMat.at<Vec3b>(i / w, i % w)[0];

x = x | 255;

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[0]) << 7);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[1]) << 15);

x = x | (((uchar)inMat.at<Vec3b>(i / w, i % w)[2]) << 23);

outImage[i] = x;

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

传递图片中所选取点的坐标信息

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray, jintArray, jintArray, jintArray, jint);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_PasPoint

(JNIEnv \* env, jclass, jintArray px, jintArray py, jintArray qx , jintArray qy, jint len){

LOGE("select\_point size", len);

jint \* arrpx, \* arrpy; 定义一个整形指针

int sum=0;

对于整形数组的处理，主要有GetIntArrayElements与GetIntArrayRegion

第一种方法

arrpx=env->GetIntArrayElements(px, false); 得到一个指向原始数据类型内容的指针

arrpy=env->GetIntArrayElements(py, false);

jint length=env->GetArrayLength(array1); 得到数组的长度

jint \* arrqx, \* arrqy;

arrqx= env->GetIntArrayElements(qx, false);

arrqy= env->GetIntArrayElements(qy, false);

LOGE("select\_point size", len);

首先清楚控制点中的数据4

p.clear();

q.clear();

for(int i = 0; i < len; ++i){

p.push\_back(Point2d(arrpy[i], arrpx[i]));

q.push\_back(Point2d(arrqy[i], arrqx[i]));

}

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

IplImage \* change4channelTo3InIplImage(IplImage \* src);

extern "C" {

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h);

JNIEXPORT void JNICALL Java\_com\_example\_dddddd\_LibImgFun\_ImgFun(

JNIEnv\* env, jobject obj, jclass bitmapcolor, int w, int h) {

jint \*cbuf;

cbuf = env->GetIntArrayElements(buf, NULL);

if (cbuf == NULL) {

return 0;

}

Mat myimg(h, w, CV\_8UC4, (unsigned char\*) cbuf);

cvtColor(myimg,myimg,CV\_RGBA2BGR);

IplImage image=IplImage(myimg);

IplImage\* image3channel = change4channelTo3InIplImage(&image);

IplImage\* pCannyImage = cvCreateImage(cvGetSize(image3channel),IPL\_DEPTH\_8U,1);

GaussianBlur(myimg, myimg, Size( 5, 5 ), 0, 0 );

从BGR转换回RGBA，可再将数据拷贝回bitmap

cvCanny(image3channel,pCannyImage,50,150,3);

cvtColor(myimg, myimg, CV\_BGR2RGBA);

IplImage pCannyImage(myimg);

IplImage \*p = &pCannyImage;

int\* outImage=new int[w\*h];

for(int i=0;i<w\*h;i++)

{

outImage[i]=(int)(&image)->imageData[i];

}

int size = w \* h;

jintArray result = env->NewIntArray(size);

env->SetIntArrayRegion(result, 0, size, outImage);

env->ReleaseIntArrayElements(buf, cbuf, 0);

return result;

}

}

IplImage \* change4channelTo3InIplImage(IplImage \* src) {

if (src->nChannels != 4) {

return NULL;

}

IplImage \* destImg = cvCreateImage(cvGetSize(src), IPL\_DEPTH\_8U, 3);

for (int row = 0; row < src->height; row++) {

for (int col = 0; col < src->width; col++) {

CvScalar s = cvGet2D(src, row, col);

cvSet2D(destImg, row, col, s);

}

}

return destImg;

}

# 3.界面相关布局文件代码

## 3.1 btn\_shap\_other.xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle" >

<solid android:color="#F5DEB3"/>

<corners

android:topLeftRadius="8dip"

android:topRightRadius="8dip"

android:bottomLeftRadius="8dip"

android:bottomRightRadius="8dip"/>

</shape>

## 3.2 btn\_shap\_xml

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle"> >

<solid android:color="#C0C0C0"/>

<corners

android:topLeftRadius="8dip"

android:topRightRadius="8dip"

android:bottomLeftRadius="8dip"

android:bottomRightRadius="8dip"/>

</shape>

## 3.3 activity\_main.xml

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="com.example.dddddd.MainActivity" >

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/hello\_world" />

<EditText

android:id="@+id/test"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/textView1"

android:layout\_marginTop="63dp"

android:layout\_toRightOf="@+id/textView1"

android:ems="10" >

<requestFocus />

</EditText>

</RelativeLayout>

3.4 main\_text.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:background="#4A494F">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:background="#1e222c"

android:layout\_marginTop="10dp"

>

<!--

<Button

android:id="@+id/MRLS"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_margin="6dp"

android:layout\_weight="1"

android:background="@drawable/btn\_shap"

android:text="MRLS"

android:textColor="#28004D" />

-->

<Button

android:id="@+id/btnNDK"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap"

android:textColor="#28004D"

android:drawableTop="@drawable/bianxing1"

android:text="变形" />

<Button

android:id="@+id/btnRestore"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap\_other"

android:drawableTop="@drawable/huanyuan"

android:text="还原" />

<Button

android:id="@+id/choose\_photo"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:onClick="load"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap"

android:drawableTop="@drawable/huoqu2"

android:text="选取图片" />

<Button

android:id="@+id/sava\_photo"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap\_other"

android:onClick="Img\_save"

android:drawableTop="@drawable/baocun"

android:text="保存" />

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp"

android:layout\_marginBottom="10dp"

android:background="#1e222c">

<TextView

android:id="@+id/algoritm"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:gravity="center"

android:layout\_margin="15dp"

android:textColor="#2250DD"

android:background="@drawable/btn\_shap\_other"

android:text="算法选择 :" />

<Spinner

android:id="@+id/spinner1"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:entries="@array/algorithms" >

</Spinner>

</LinearLayout>

<ImageView

android:id="@+id/ImageView01"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"

android:background="#1e222c"

android:orientation="horizontal" >

<TextView

android:id="@+id/inter\_width"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:gravity="center"

android:textColor="#2250DD"

android:text="Selct interpolation width :" />

<NumberPicker

android:id="@+id/valuePicker"

android:layout\_width="231dp"

android:layout\_height="114dp" />

</LinearLayout>

</LinearLayout>

## 3.4 main\_xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:background="#4A494F">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:background="#1e222c"

android:layout\_marginTop="10dp"

>

<!--

<Button

android:id="@+id/MRLS"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_margin="6dp"

android:layout\_weight="1"

android:background="@drawable/btn\_shap"

android:text="MRLS"

android:textColor="#28004D" />

-->

<Button

android:id="@+id/btnNDK"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap"

android:textColor="#28004D"

android:drawableTop="@drawable/bianxing1"

android:text="变形" />

<Button

android:id="@+id/btnRestore"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap\_other"

android:drawableTop="@drawable/huanyuan"

android:text="还原" />

<Button

android:id="@+id/choose\_photo"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:onClick="load"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap"

android:drawableTop="@drawable/huoqu2"

android:text="选取图片" />

<Button

android:id="@+id/sava\_photo"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:layout\_margin="6dp"

android:background="@drawable/btn\_shap\_other"

android:onClick="Img\_save"

android:drawableTop="@drawable/baocun"

android:text="保存" />

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp"

android:layout\_marginBottom="10dp"

android:background="#1e222c">

<TextView

android:id="@+id/algoritm"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:gravity="center"

android:layout\_margin="15dp"

android:textColor="#2250DD"

android:background="@drawable/btn\_shap\_other"

android:text="算法选择 :" />

<Spinner

android:id="@+id/spinner1"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:entries="@array/algorithms" >

</Spinner>

</LinearLayout>

<ImageView

android:id="@+id/ImageView01"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"

android:background="#1e222c"

android:orientation="horizontal" >

<TextView

android:id="@+id/inter\_width"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:gravity="center"

android:textColor="#2250DD"

android:text="Selct interpolation width :" />

<NumberPicker

android:id="@+id/valuePicker"

android:layout\_width="231dp"

android:layout\_height="114dp" />

</LinearLayout>

</LinearLayout>

## 3.5 main\_xml

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

tools:context="com.example.dddddd.MainActivity" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:showAsAction="never"

android:title="@string/action\_settings"/>

</menu>

## 3.6 arrays.xml

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string-array name="algorithms">

<item>Affine</item>

<item>rigid </item>

<item>MRLS</item>

</string-array>

</resources>

## 3.7 dimens.xml

<resources>

<!-- Default screen margins, per the Android Design guidelines. -->

<dimen name="activity\_horizontal\_margin">16dp</dimen>

<dimen name="activity\_vertical\_margin">16dp</dimen>

</resources>

## 3.8 strings.xml

<?xml version="1.0" encoding="utf-8" standalone="no"?>

<resources>

<string name="app\_name">dddddd</string>

<string name="hello\_world">Hello world!</string>

<string name="action\_settings">Settings</string>

<string name="mls\_name">MRLS\_IMG</string>

</resources>

## 3.9 styles.xml

<resources>

<!--

Base application theme, dependent on API level. This theme is replaced

by AppBaseTheme from res/values-vXX/styles.xml on newer devices.

-->

<style name="AppBaseTheme" parent="android:Theme.Light">

<!--

Theme customizations available in newer API levels can go in

res/values-vXX/styles.xml, while customizations related to

backward-compatibility can go here.

-->

</style>

<!-- Application theme. -->

<style name="AppTheme" parent="AppBaseTheme">

<!-- All customizations that are NOT specific to a particular API-level can go here. -->

</style>

</resources>

# 4.JAVA源码

## 4.1 LIbImgFun.java

package com.example.dddddd;

import java.lang.reflect.Array;

import android.R.integer;

import android.graphics.Bitmap;

public class LibImgFun {

static {

System.loadLibrary("ImgFun");

}

/\*\*

\* 加载.so文件

\* @param width the current view width

\* @param height the current view height

\*/

public static native int ImgFun(Bitmap in, int choose, int inter\_width);

// public static native int []ImgFirst(int []buff, int w, int h);

// public static native int ImgMrls(Bitmap in, int choose);

public static native void PasPoint(int []px, int []py, int []qx, int []qy, int len);

}

## MianAvtivity.java

package com.example.dddddd;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.security.PublicKey;

import java.text.ParsePosition;

import java.util.ArrayList;

import android.R.integer;

import android.R.interpolator;

//import java.nio.ByteBuffer;

//

//import javax.crypto.spec.IvParameterSpec;

//

//import org.apache.commons.logging.Log;

//import com.example.flitter\_tesst.LibImgFun;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.graphics.Bitmap;

import android.graphics.Bitmap.Config;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.Point;

import android.graphics.Paint.Style;

import android.graphics.drawable.BitmapDrawable;

import android.net.Uri;

import android.os.Bundle;

import android.os.Environment;

import android.provider.MediaStore;

import android.provider.MediaStore.Images.Thumbnails;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.MotionEvent;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemSelectedListener;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.NumberPicker;

import android.widget.NumberPicker.OnValueChangeListener;

import android.widget.Toast;

import android.widget.SimpleAdapter.ViewBinder;

import android.widget.Spinner;

import android.view.View;

import android.view.WindowManager;

public class MainActivity extends Activity {

ImageView imgView;

Button btnNDK;

Button btnMRLS;

Button btnRestore;

// 变形插值点间距

int interpolation\_width = 10;

// 绘画图，原图，变形图

Bitmap source, tpimg;

Bitmap img\_source, img, test, img\_sourc\_sec, save\_img = null;

boolean Img\_trans = false;

// 用于画变换点的组件

private int startx;

private int starty;

private Point end;

boolean restore = true;

private Bitmap canv;

private Canvas canvas;

private Paint paint, Cirpaint;

private int[] px, py, qx, qy;

private int len;

private ArrayList pxx, pyy, qxx, qyy;

private int Algorithm\_chosse;

private double Scale\_rateX, Scale\_rateY;

private NumberPicker valuepicker;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main\_test);

this.setTitle("NDK测试");

btnRestore = (Button) this.findViewById(R.id.btnRestore);

btnRestore.setOnClickListener(new ClickEvent());

// 插值步长设置

valuepicker = (NumberPicker) findViewById(R.id.valuePicker);

valuepicker.setMinValue(1);

valuepicker.setMaxValue(50);

valuepicker.setValue(10);

valuepicker.setOnValueChangedListener(new OnValueChangeListener() {

@Override

public void onValueChange(NumberPicker picker, int oldVal, int newVal) {

// TODO Auto-generated method stub

interpolation\_width = newVal;

}

});

btnNDK = (Button) this.findViewById(R.id.btnNDK);

btnNDK.setOnClickListener(new ClickEvent());

//btnMRLS=(Button) this.findViewById(R.id.MRLS);

//btnMRLS.setOnClickListener(new ClickEvent());

img = ((BitmapDrawable) getResources().getDrawable(R.drawable.candle)).getBitmap();

imgView = (ImageView) this.findViewById(R.id.ImageView01);

Img\_show\_and\_wrap();

// 点集初始化

pxx = new ArrayList();

pyy = new ArrayList();

qxx = new ArrayList();

qyy = new ArrayList();

source = Bitmap.createBitmap(img);

tpimg = Bitmap.createBitmap(source);

imgView.setImageBitmap(img);

canv = Bitmap.createBitmap(img.getWidth(), img.getHeight(), img.getConfig());

// 必须的用白色的图片来初始化canvas, canv才承载了所有canvas自己所进行的操作

canvas = new Canvas(canv);

// 初始化画笔

paint = new Paint();

Cirpaint = new Paint();

end = new Point();

paint.setColor(Color.RED);

paint.setStrokeWidth(2);

paint.setAntiAlias(true);// 锯齿不显示

Cirpaint.setColor(Color.rgb(20, 230, 20));

Cirpaint.setStyle(Style.FILL);

imgView.setOnTouchListener(new DrawLine());

// algorithm choose

// 初始化控件

Spinner spinner = (Spinner) findViewById(R.id.spinner1);

// 建立数据源

String[] mItems = getResources().getStringArray(R.array.algorithms);

// 建立Adapter并且绑定数据源

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item, mItems);

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

// 绑定 Adapter到控件

spinner.setAdapter(adapter);

spinner.setOnItemSelectedListener(new OnItemSelectedListener() {

@Override

public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {

Algorithm\_chosse = pos + 1;

// String[] languages =

// getResources().getStringArray(R.array.algorithms);

// Toast.makeText(MainActivity.this, "你点击的是:"+languages[pos],

// 2000).show();

}

@Override

public void onNothingSelected(AdapterView<?> arg0) {

// TODO Auto-generated method stub

}

});

}

// 显示图像变形图像复制

void Img\_show\_and\_wrap() {

img\_source = img.copy(img.getConfig(), true);

// img\_sourc\_sec = Bitmap.createBitmap(img\_source);

img = adapter\_screen\_img(img);

// --------image inialize end

}

// --------------------图片屏幕适配(问题)

Bitmap adapter\_screen\_img(Bitmap temp) {

// 获取图像来源

/\*

\* 用适配屏幕大小的图片来初试化View的图片

\*/

WindowManager wm1 = this.getWindowManager();

int width = wm1.getDefaultDisplay().getWidth();

int height = wm1.getDefaultDisplay().getHeight();

Scale\_rateX = (float) width / temp.getWidth();

Scale\_rateY = (float) height / temp.getWidth();

Matrix matrix = new Matrix();

matrix.postScale((float) width / temp.getWidth(), 0.5f \* (float) height / temp.getHeight());

temp = Bitmap.createBitmap(temp, 0, 0, temp.getWidth(), temp.getHeight(), matrix, true);

return temp;

}

// --------------------点击事件

class ClickEvent implements View.OnClickListener {

public void onClick(View v) {

if (v == btnNDK) {

// ------- add point

if (pxx.size() <= 0) {

Toast.makeText(MainActivity.this, "没有选取点", 2000).show();

return;

}

len = pxx.size();

Log.i("选取点的大小", "here");

px = new int[len];

py = new int[len];

qx = new int[len];

qy = new int[len];

Log.e("here", "转换前");

for (int i = 0; i < len; ++i) {

px[i] = (Integer) pxx.get(i);

py[i] = (Integer) pyy.get(i);

qx[i] = (Integer) qxx.get(i);

qy[i] = (Integer) qyy.get(i);

}

Log.e("here", "转换后");

// ----------------------------------------------

LibImgFun.PasPoint(px, py, qx, qy, len);

// ------- add point

// 去除图中绘制的路径点

long current = System.currentTimeMillis();

// 深复制

Bitmap img1 = Bitmap.createBitmap(img\_source.getWidth(), img\_source.getHeight(),

img\_source.getConfig());

img1.eraseColor(Color.RED);

img1 = img\_source.copy(img\_source.getConfig(), true);

Bitmap img2 = Bitmap.createBitmap(img1.getWidth(), img1.getHeight(), img1.getConfig());

img2.eraseColor(Color.parseColor("#FF00FF"));

LibImgFun.ImgFun(img1, Algorithm\_chosse, interpolation\_width);

Img\_trans = true;

long performance = System.currentTimeMillis() - current;

save\_img = img1.copy(img1.getConfig(), true);

// 变换后的图片适配屏幕

img1 = adapter\_screen\_img(img1);

imgView.setImageBitmap(img1);

MainActivity.this.setTitle("w:" + String.valueOf(img1.getWidth()) + ",h:"

+ String.valueOf(img1.getHeight()) + "NDK耗时" + String.valueOf(performance) + " 毫秒");

} else if (v == btnRestore) {

if (pxx.size() > 0) {

pxx.clear();

pyy.clear();

qxx.clear();

qyy.clear();

}

imgView.setImageBitmap(img);

restore = true;

MainActivity.this.setTitle("NDK\_TEST");

}

}

}

public byte[] getPixelsBGR(Bitmap image) {

// calculate how many bytes our image consists of

int bytes = image.getByteCount();

ByteBuffer buffer = ByteBuffer.allocate(bytes); // Create a new buffer

image.copyPixelsToBuffer(buffer); // Move the byte data to the buffer

byte[] temp = buffer.array(); // Get the underlying array containing the

// data.

byte[] pixels = new byte[(temp.length / 4) \* 3]; // Allocate for BGR

// Copy pixels into place

for (int i = 0; i < temp.length / 4; i++) {

pixels[i \* 3] = temp[i \* 4 + 2]; // B

pixels[i \* 3 + 1] = temp[i \* 4 + 1]; // G

pixels[i \* 3 + 2] = temp[i \* 4]; // R

}

return pixels;

}

class DrawLine implements View.OnTouchListener {

@Override

public boolean onTouch(View v, MotionEvent event) {

int action = event.getAction();

switch (action) {

case MotionEvent.ACTION\_DOWN:

startx = (int) event.getX();

starty = (int) event.getY();

break;

case MotionEvent.ACTION\_MOVE:

// 赋值上一拷贝的图像

if (restore) {

tpimg = Bitmap.createBitmap(img);

source = Bitmap.createBitmap(img);

restore = false;

// 选取点重设

} else {

tpimg = Bitmap.createBitmap(source);

}

// 将上一次画完的图片画到canvas上去，再来在其上面进行绘制

canvas.drawBitmap(tpimg, new Matrix(), paint);

int x = (int) event.getX();

int y = (int) event.getY();

end = new Point(x, y);

canvas.drawLine(startx, starty, x, y, paint);

// Bitmap canv才是canvas的画板

imgView.setImageBitmap(canv);

break;

case MotionEvent.ACTION\_UP:

pxx.add((int) (startx / Scale\_rateX));

pyy.add((int) (2 \* starty / Scale\_rateY));

// 添加点

qxx.add((int) (end.x / Scale\_rateX));

qyy.add((int) (2 \* end.y / Scale\_rateY));

// 末点处画上一个圆

canvas.drawCircle(end.x, end.y, 8, Cirpaint);

source = Bitmap.createBitmap(canv);

imgView.setImageBitmap(source);

Toast.makeText(MainActivity.this,

"Point\_num: " + pxx.size() + " p:(" + startx + "," + starty + ")" + " " + "q:" + end, 1000)

.show();

break;

}

return true;

}

}

// --------------------图片选取

public void load(View view) {

// 激活系统图库，选择一张图片

Intent intent = new Intent();

intent.setAction(Intent.ACTION\_PICK);

intent.setType("image/\*");

startActivityForResult(intent, 0);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (data != null) {

// 得到图片的全路径

Uri uri = data.getData();

// 通过路径加载图片

// 这里省去了图片缩放操作，如果图片过大，可能会导致内存泄漏

// 图片缩放的实现，请看：http://blog.csdn.net/reality\_jie\_blog/article/details/16891095

String selectImagePath = getPath(uri);

try {

img = MediaStore.Images.Media.getBitmap(this.getContentResolver(), uri);

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

// 源图像设置

// img = lessenUriImage(selectImagePath);

// 初始化变形图像，并且使img适应屏幕大小

Img\_show\_and\_wrap();

// imgView更新

imgView.setImageBitmap(img);

// imgView.setImageURI(uri);

// 获取图片的缩略图，可能为空！

// Bitmap bitmap = data.getParcelableExtra("data");

// this.iv\_image.setImageBitmap(bitmap);

}

super.onActivityResult(requestCode, resultCode, data);

}

// -----uri转换成Bitmap的方法

public String getPath(Uri uri) {

String[] projection = { MediaStore.Images.Media.DATA };

Cursor cursor = managedQuery(uri, projection, null, null, null);

int column\_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);

cursor.moveToFirst();

return cursor.getString(column\_index);

}

public final static Bitmap lessenUriImage(String path) {

BitmapFactory.Options options = new BitmapFactory.Options();

options.inJustDecodeBounds = true;

Bitmap bitmap = BitmapFactory.decodeFile(path, options); // 此时返回 bm 为空

options.inJustDecodeBounds = false; // 缩放比。由于是固定比例缩放，只用高或者宽其中一个数据进行计算即可

int be = (int) (options.outHeight / (float) 320);

if (be <= 0)

be = 1;

options.inSampleSize = be; // 重新读入图片，注意此时已经把 options.inJustDecodeBounds

// 设回 false 了

bitmap = BitmapFactory.decodeFile(path, options);

// int w = bitmap.getWidth();

// int h = bitmap.getHeight();

// System.out.println(w + " " + h); // after zoom

return bitmap;

}

// 保存图片到SD卡中

public void Img\_save(View view) {

if (!Img\_trans)

Toast.makeText(MainActivity.this, "图像没有变形", 2000).show();

else

saveImageToGallery(getApplicationContext(), save\_img);

}

// 保存文件到指定路径

public boolean saveImageToGallery(Context context, Bitmap bmp) {

// 首先保存图片

String storePath = Environment.getExternalStorageDirectory().getAbsolutePath() + File.separator

+ "MLS\_TRANSFORMATION";

File appDir = new File(storePath);

if (!appDir.exists()) {

appDir.mkdir();

}

String fileName = System.currentTimeMillis() + ".jpg";

File file = new File(appDir, fileName);

try {

FileOutputStream fos = new FileOutputStream(file);

// 通过io流的方式来压缩保存图片

boolean isSuccess = bmp.compress(Bitmap.CompressFormat.JPEG, 60, fos);

fos.flush();

fos.close();

// 把文件插入到系统图库

// MediaStore.Images.Media.insertImage(context.getContentResolver(),

// file.getAbsolutePath(), fileName, null);

// 保存图片后发送广播通知更新数据库

Uri uri = Uri.fromFile(file);

context.sendBroadcast(new Intent(Intent.ACTION\_MEDIA\_SCANNER\_SCAN\_FILE, uri));

Toast.makeText(MainActivity.this, "图片保存成功", 2000).show();

if (isSuccess) {

return true;

} else {

return false;

}

} catch (IOException e) {

e.printStackTrace();

}

return false;

}

}

## R.java

/\* AUTO-GENERATED FILE. DO NOT MODIFY.

\*

\* This class was automatically generated by the

\* aapt tool from the resource data it found. It

\* should not be modified by hand.

\*/

package com.example.dddddd;

public final class R {

public static final class array {

public static final int algorithms=0x7f040000;

}

public static final class attr {

}

public static final class dimen {

/\*\* Default screen margins, per the Android Design guidelines.

Example customization of dimensions originally defined in res/values/dimens.xml

(such as screen margins) for screens with more than 820dp of available width. This

would include 7" and 10" devices in landscape (~960dp and ~1280dp respectively).

\*/

public static final int activity\_horizontal\_margin=0x7f050000;

public static final int activity\_vertical\_margin=0x7f050001;

}

public static final class drawable {

public static final int baocun=0x7f020000;

public static final int bianxing1=0x7f020001;

public static final int btn\_shap=0x7f020002;

public static final int btn\_shap\_other=0x7f020003;

public static final int candle=0x7f020004;

public static final int huanyuan=0x7f020005;

public static final int huoqu2=0x7f020006;

public static final int ic\_launcher=0x7f020007;

public static final int search=0x7f020008;

public static final int smile=0x7f020009;

public static final int tubiao=0x7f02000a;

}

public static final class id {

public static final int ImageView01=0x7f090008;

public static final int action\_settings=0x7f09000b;

public static final int algoritm=0x7f090006;

public static final int btnNDK=0x7f090002;

public static final int btnRestore=0x7f090003;

public static final int choose\_photo=0x7f090004;

public static final int inter\_width=0x7f090009;

public static final int sava\_photo=0x7f090005;

public static final int spinner1=0x7f090007;

public static final int test=0x7f090001;

public static final int textView1=0x7f090000;

public static final int valuePicker=0x7f09000a;

}

public static final class layout {

public static final int activity\_main=0x7f030000;

public static final int main\_test=0x7f030001;

}

public static final class menu {

public static final int main=0x7f080000;

}

public static final class string {

public static final int action\_settings=0x7f060002;

public static final int app\_name=0x7f060000;

public static final int hello\_world=0x7f060001;

public static final int mls\_name=0x7f060003;

}

public static final class style {

/\*\*

Base application theme, dependent on API level. This theme is replaced

by AppBaseTheme from res/values-vXX/styles.xml on newer devices.

Theme customizations available in newer API levels can go in

res/values-vXX/styles.xml, while customizations related to

backward-compatibility can go here.

Base application theme for API 11+. This theme completely replaces

AppBaseTheme from res/values/styles.xml on API 11+ devices.

API 11 theme customizations can go here.

Base application theme for API 14+. This theme completely replaces

AppBaseTheme from BOTH res/values/styles.xml and

res/values-v11/styles.xml on API 14+ devices.

API 14 theme customizations can go here.

\*/

public static final int AppBaseTheme=0x7f070000;

/\*\* Application theme.

All customizations that are NOT specific to a particular API-level can go here.

\*/

public static final int AppTheme=0x7f070001;

}

}