Lab Report

——for android app 'Kanvas'

2017.05.20

1 Prototype System Introduction

1.1 Name

We name our first app 'Kanvas', which signifies unbounded possibilities and a broad spectrum of operations that can be implemented on the photo. The Kanvas is where users' imagination and artistic taste can be interpreted and visualized with most simple motions such as clicks and slidings.

1.2 Functions

(1) Allow photos from the phone or those taken instantly from camera as sources for processing
(2) Allow filters onto a photo
(3) Allow basic enhancement on the photo (cropping, contrast, brightness, saturation etc…)
(4) Allow watermarks, doodling, texts and other relatively trivial functions

1.3 Running Environment

Windows 10 (64-bits)

1.4 Developing Environment

Android Studio 2.3.2

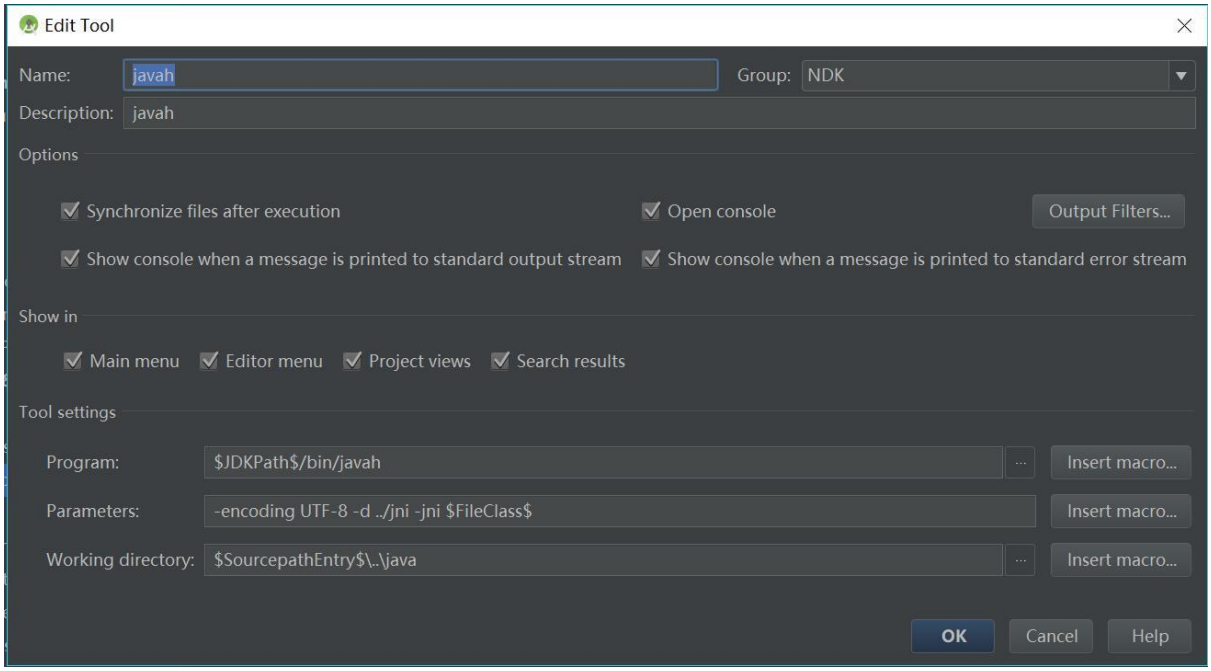
2 Task Allocation

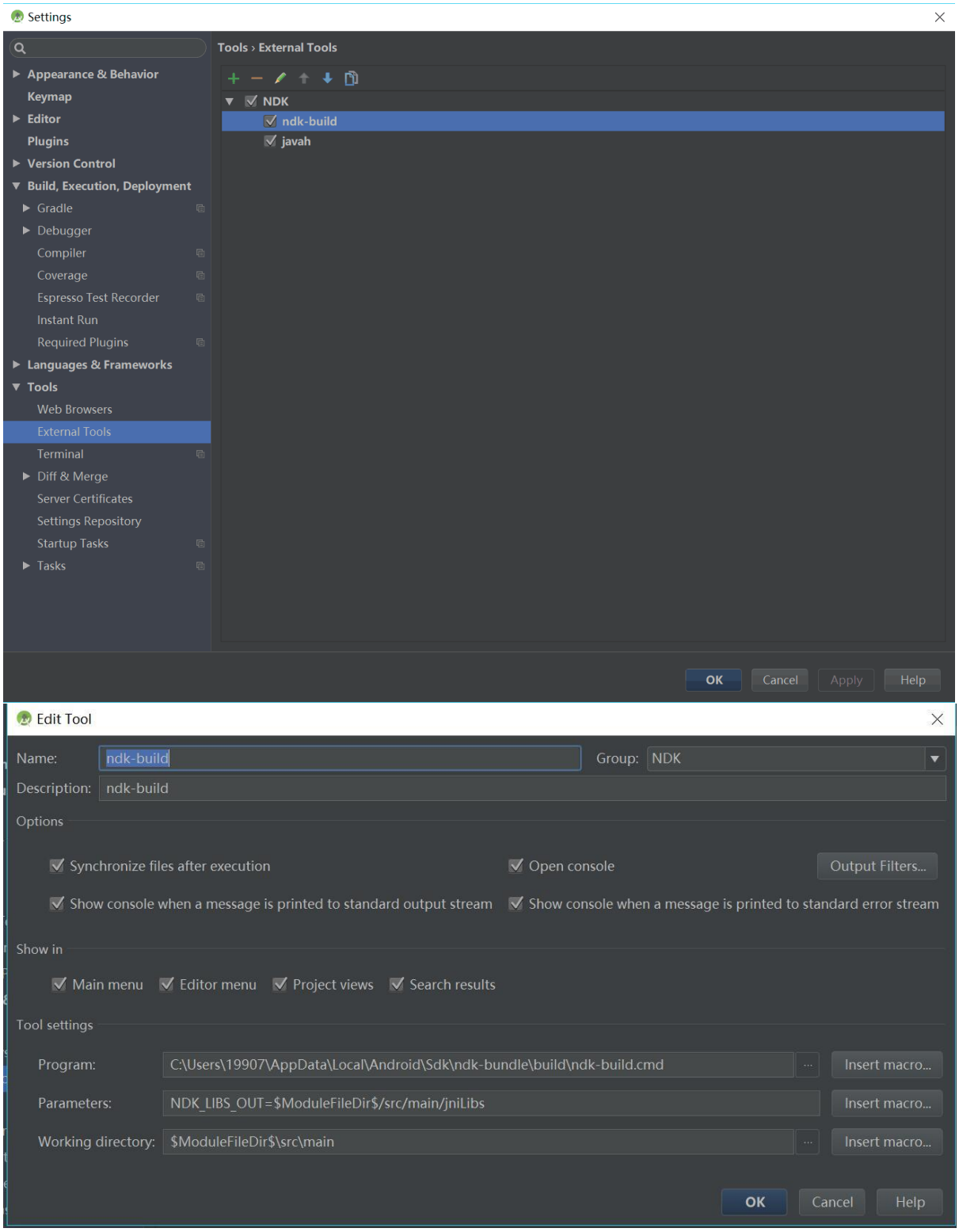
User Interface: Lu Xinyu & Xu Sihui
Algorithm: Chen Xinyue & Xie Fan
Environment Building and Configuration (developing via jni): Chen Xinyue & Xie Fan
PPT: Lu Xinyu, Xie Fan, Xu Sihui
Lab Report: Chen Xinyue
Debugging and System Testing: All the members

2 Building up Environment for NDK Development

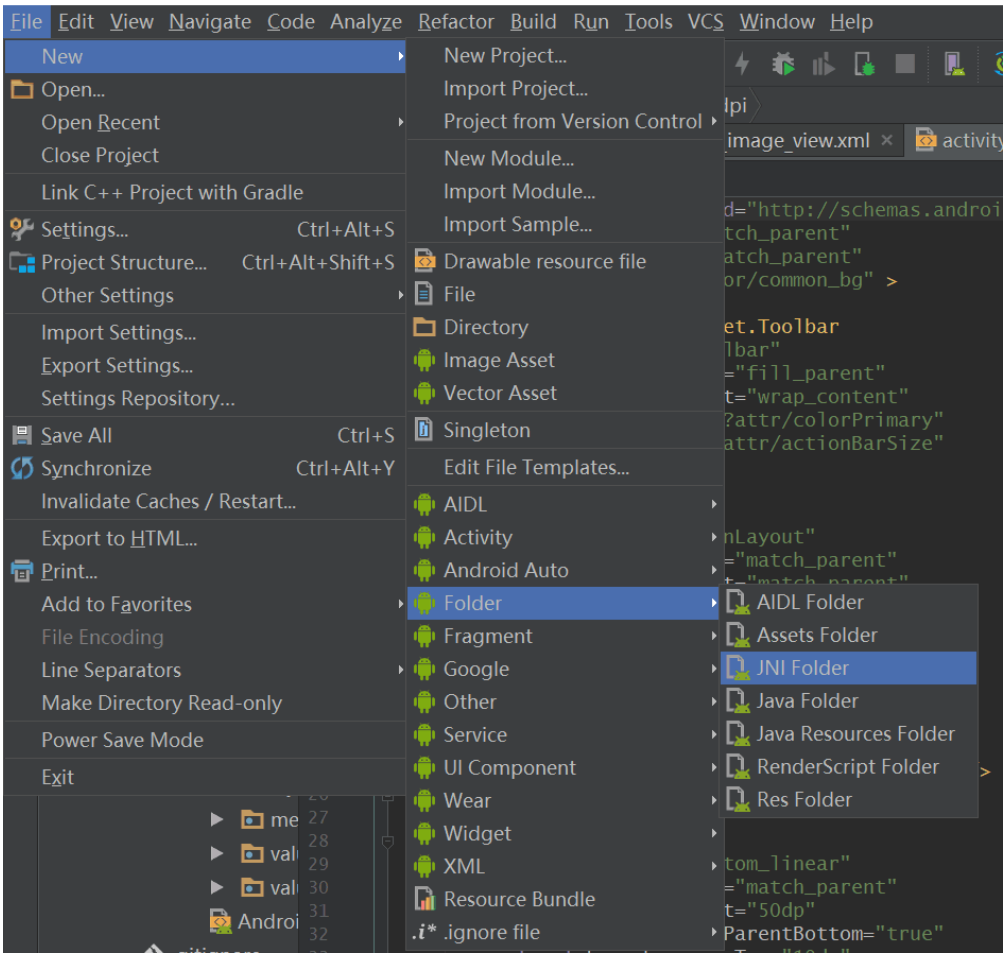
We choose to use c++ to develop our core algorithms which enable the users to warp the photos or add filters onto the photos. Therefore, as a method to bridge java and c++ on the platform of Android Studio, NDK development is a prime choice. This part requires a little configuration beforehand. The steps are listed as follows:

a. Add shortcuts





b. New a JNI folder



c. Revise gradle.properties

add "`android.useDeprecatedNdk=true`"

d. Add Android.mk、Application.mk and cpp under jni folder

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := nativefilter
LOCAL_SRC_FILES =: NativeFilter.cpp
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)

LOCAL_MODULE := picwarp
LOCAL_SRC_FILES =: Picwarp.cpp
include $(BUILD_SHARED_LIBRARY)
```

Android.mk

```
APP_STL:=gnustl_static
APP_CPPFLAGS:=-frtti -fexceptions
APP_ABI := all

#armeabi armeabi-v7a x86 x86_64 arm64-v8a
```

Application.mk

Then we can directly add cpp files under the folder, with a free head files included, the c++ codes will then integrate into the program.

e. Compile the files using ndk-build, which has been initialized just now, from external tools.

3 System Architecture

3.1 User Interface Component

Event Control

1. selection of sources: taken from camera or from album

```
    /* 从相册中获取照片 */
    private void getPictureFromPhoto() {
        Intent openphotoIntent = new Intent(Intent.ACTION_PICK,
                android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(openphotoIntent, PHOTO_PICKED_WITH_DATA);
    }

    /* 从相机中获取照片 */
    private void getPictureFormCamera() {
        Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");

        tempPhotoPath = FileUtils.DCIMCamera_PATH + FileUtils.getNewFileName()
                + ".jpg";

        mCurrentPhotoFile = new File(tempPhotoPath);

        if (!mCurrentPhotoFile.exists()) {
            try {
                mCurrentPhotoFile.createNewFile();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        intent.putExtra(MediaStore.EXTRA_OUTPUT,
                Uri.fromFile(mCurrentPhotoFile));
        startActivityForResult(intent, CAMERA_WITH_DATA);
    }
```

2. add text

```
/**
 * 添加文字图片工具类
 */
public class OperateUtils
{
   private Activity activity;
   private int screenWidth;// 手机屏幕的宽（像素）
   private int screenHeight;// 手机屏幕的高（像素）

   public static final int LEFTTOP = 1;
   public static final int RIGHTTOP = 2;
   public static final int LEFTBOTTOM = 3;
   public static final int RIGHTBOTTOM = 4;
   public static final int CENTER = 5;

   public OperateUtils(Activity activity)
   {
      this.activity = activity;
      if (screenWidth == 0)
      {
         DisplayMetrics metric = new DisplayMetrics();
         activity.getWindowManager().getDefaultDisplay().getMetrics(metric);
         screenWidth = metric.widthPixels; // 屏幕宽度（像素）
        screenHeight = metric.widthPixels; // 屏幕宽度（像素）
      }
   }

   /**
    * 根据路径获取图片并且压缩，适应view
```

```
     *
     * @param filePath
     *            图片路径
     * @param contentView
     *            适应的view
     * @return Bitmap 压缩后的图片
     */
    public Bitmap compressionFiller(String filePath, View contentView)
    {
        BitmapFactory.Options opt = new BitmapFactory.Options();
        opt.inPreferredConfig = Bitmap.Config.RGB_565;
        opt.inPurgeable = true;
        opt.inInputShareable = true;
        Bitmap bitmap = BitmapFactory.decodeFile(filePath, opt);
        int layoutHeight = contentView.getHeight();
        float scale = 0f;
        int bitmapHeight = bitmap.getHeight();
        int bitmapWidth = bitmap.getWidth();
        scale = bitmapHeight > bitmapWidth
              ? layoutHeight / (bitmapHeight * 1f)
              : screenWidth / (bitmapWidth * 1f);
        Bitmap resizeBmp;
        if (scale != 0)
        {
            int bitmapheight = bitmap.getHeight();
            int bitmapwidth = bitmap.getWidth();
            Matrix matrix = new Matrix();
            matrix.postScale(scale, scale); // 长和宽放大缩小的比例
          resizeBmp = Bitmap.createBitmap(bitmap, 0, 0, bitmapwidth,
                  bitmapheight, matrix, true);
        } else
        {
            resizeBmp = bitmap;
        }
        return resizeBmp;
    }

    /**
     * 根据压缩图片并且适应view
     *
     * @param bitmap
     *            压缩图片
     * @param contentView
     *            适应的view
     * @return 压缩后的图片
     */
    public Bitmap compressionFiller(Bitmap bitmap, View contentView)
    {
        int layoutHeight = contentView.getHeight();
        float scale = 0f;
        int bitmapHeight = bitmap.getHeight();
        int bitmapWidth = bitmap.getWidth();
        scale = bitmapHeight > bitmapWidth
              ? layoutHeight / (bitmapHeight * 1f)
              : screenWidth / (bitmapWidth * 1f);
        Bitmap resizeBmp;
        if (scale != 0)
        {
            int bitmapheight = bitmap.getHeight();
            int bitmapwidth = bitmap.getWidth();
            Matrix matrix = new Matrix();
```

```
            matrix.postScale(scale, scale); // 长和宽放大缩小的比例
            resizeBmp = Bitmap.createBitmap(bitmap, 0, 0, bitmapwidth,
                    bitmapheight, matrix, true);
        } else
        {
            resizeBmp = bitmap;
        }
        return resizeBmp;
    }

    /**
     * 添加文字方法
     *
     * @param text
     *              所添加的文字
     * @return TextObject 返回的字体图片对象
     */
    public TextObject getTextObject(String text)
    {
        TextObject textObj = null;
        if (TextUtils.isEmpty(text))
        {
            Toast.makeText(activity, "请添加文字", Toast.LENGTH_SHORT).show();
            return null;
        }

        Bitmap rotateBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.rotate);
        Bitmap deleteBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.delete);

        textObj = new TextObject(activity, text, 150, 150, rotateBm, deleteBm);
        textObj.setTextObject(true);
        return textObj;
    }
    /**
     * 添加图片的方法
     *
     * @param text
     *              文本内容
     * @param operateView
     *              容器 View 对象
     * @param quadrant
     *              需要图片显示的区域 （1、左上方，2、右上方，3、左下方，4、右下方，5、中
心）
     * @param x
     *              离边界 x 坐标
     * @param y
     *              离边界 y 坐标
     * @return
     */
    public TextObject getTextObject(String text, OperateView operateView,
            int quadrant, int x, int y)
    {
        TextObject textObj = null;
        if (TextUtils.isEmpty(text))
        {
            Toast.makeText(activity, "请添加文字", Toast.LENGTH_SHORT).show();
            return null;
        }
        int width = operateView.getWidth();
```

```
        int height = operateView.getHeight();
        switch (quadrant)
        {
            case LEFTTOP :
                break;
            case RIGHTTOP :
                x = width - x;
                break;
            case LEFTBOTTOM :
                y = height - y;
                break;
            case RIGHTBOTTOM :
                x = width - x;
                y = height - y;
                break;
            case CENTER :
                x = width / 2;
                y = height / 2;
                break;
            default :
                break;
        }
        Bitmap rotateBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.rotate);
        Bitmap deleteBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.delete);
        textObj = new TextObject(activity, text, x, y, rotateBm, deleteBm);
        textObj.setTextObject(true);
        return textObj;
    }

    /**
     * 添加图片方法
     *
     * @param srcBmp
     *            被操作的图片
     * @return
     */

    public ImageObject getImageObject(Bitmap srcBmp)
    {
        Bitmap rotateBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.rotate);
        Bitmap deleteBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.delete);
        ImageObject imgObject = new ImageObject(srcBmp, rotateBm, deleteBm);
        Point point = new Point(20, 20);
        imgObject.setPoint(point);
        return imgObject;
    }

    /**
     * 添加图片方法
     *
     * @param srcBmp
     *            被操作的图片
     * @param operateView
     *            容器View对象
     * @param quadrant
     *            需要图片显示的区域 （1、左上方，2、右上方，3、左下方，4、右下方，5、中
心）
```

```
     * @param x
     *            离边界 x 坐标
     * @param y
     *            离边界 y 坐标
     * @return
     */

    public ImageObject getImageObject(Bitmap srcBmp, OperateView operateView,
            int quadrant, int x, int y)
    {
        Bitmap rotateBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.rotate);
        Bitmap deleteBm = BitmapFactory.decodeResource(activity.getResources(),
                R.drawable.delete);
        int width = operateView.getWidth();
        int height = operateView.getHeight();
//      int srcWidth = srcBmp.getWidth();
//      int srcHeight = srcBmp.getHeight();
//      if (height > width)
//      {
//          if (srcHeight > srcWidth)
//          {
//              srcBmp = ImageUtils.ResizeBitmap(srcBmp, height / 3 * srcWidth
//                      / srcHeight, height / 3);
//          } else
//          {
//              srcBmp = ImageUtils.ResizeBitmap(srcBmp, width / 3, width / 3
//                      * srcHeight / srcWidth);
//          }
//      } else
//      {
//          if (srcHeight > srcWidth)
//          {
//              srcBmp = ImageUtils.ResizeBitmap(srcBmp, height / 2 * srcWidth
//                      / srcHeight, height / 2);
//          } else
//          {
//              srcBmp = ImageUtils.ResizeBitmap(srcBmp, width / 3, width / 3
//                      * srcHeight / srcWidth);
//          }
//      }
        switch (quadrant)
        {
            case LEFTTOP :
                break;
            case RIGHTTOP :
                x = width - x;
                break;
            case LEFTBOTTOM :
                y = height - y;
                break;
            case RIGHTBOTTOM :
                x = width - x;
                y = height - y;
                break;
            case CENTER :
                x = width / 2;
                y = height / 2;
                break;
            default :
                break;
        }
```

```
        ImageObject imgObject = new ImageObject(srcBmp, x, y, rotateBm,
                deleteBm);
        Point point = new Point(20, 20);
        imgObject.setPoint(point);
        return imgObject;
    }

}
```

3. add watermarks

```
public class OperateView extends View
{
    private List<ImageObject> imgLists = new ArrayList<ImageObject>();
    private Rect mCanvasLimits;
    private Bitmap bgBmp;
    private Paint paint = new Paint();
// private Context mContext;
    private boolean isMultiAdd;// true 代表可以添加多个水印图片（或文字），false 代表只可添加单个水印图片（或文字）
   private float picScale = 0.4f;
    /**
     * 设置水印图片初始化大小
     * @param picScale
     */
    public void setPicScale(float picScale)
    {
        this.picScale = picScale;
    }
    /**
     * 设置是否可以添加多个图片或者文字对象
     *
     * @param isMultiAdd
     *            true 代表可以添加多个水印图片（或文字），false 代表只可添加单个水印图片（或文字）
     */
    public void setMultiAdd(boolean isMultiAdd)
    {
        this.isMultiAdd = isMultiAdd;
    }
    public OperateView(Context context, Bitmap resizeBmp)
    {
        super(context);
//      this.mContext = context;
        bgBmp = resizeBmp;
        int width = bgBmp.getWidth();
        int height = bgBmp.getHeight();
        mCanvasLimits = new Rect(0, 0, width, height);
    }

    /**
     * 将图片对象添加到View 中
     *
     * @param imgObj
     *            图片对象
     */
    public void addItem(ImageObject imgObj)
    {
        if (imgObj == null)
        {
```

```
            return;
        }
        if (!isMultiAdd && imgLists != null)
        {
            imgLists.clear();
        }
        imgObj.setSelected(true);
        if (!imgObj.isTextObject)
        {
            imgObj.setScale(picScale);
        }
        ImageObject tempImgObj = null;
        for (int i = 0; i < imgLists.size(); i++)
        {
            tempImgObj = imgLists.get(i);
            tempImgObj.setSelected(false);
        }
        imgLists.add(imgObj);
        invalidate();
    }
    /**
     * 画出容器内所有的图像
     */
    @Override
    protected void onDraw(Canvas canvas)
    {
        super.onDraw(canvas);
        int sc = canvas.save();
        canvas.clipRect(mCanvasLimits);
        canvas.drawBitmap(bgBmp, 0, 0, paint);
        drawImages(canvas);
        canvas.restoreToCount(sc);
        for (ImageObject ad : imgLists)
        {
            if (ad != null && ad.isSelected())
            {
                ad.drawIcon(canvas);
            }
        }
    }

    public void save()
    {
        ImageObject io = getSelected();
        if (io != null)
        {
            io.setSelected(false);
        }
        invalidate();
    }

    /**
     * 根据触控点重绘View
     */
    @Override
    public boolean onTouchEvent(MotionEvent event)
    {
        if (event.getPointerCount() == 1)
        {
            handleSingleTouchManipulateEvent(event);
        } else
        {
```

```
            handleMultiTouchManipulateEvent(event);
        }
        invalidate();

        super.onTouchEvent(event);
        return true;
    }

    private boolean mMovedSinceDown = false;
    private boolean mResizeAndRotateSinceDown = false;
    private float mStartDistance = 0.0f;
    private float mStartScale = 0.0f;
    private float mStartRot = 0.0f;
    private float mPrevRot = 0.0f;
    static public final double ROTATION_STEP = 2.0;
    static public final double ZOOM_STEP = 0.01;
    static public final float CANVAS_SCALE_MIN = 0.25f;
    static public final float CANVAS_SCALE_MAX = 3.0f;
    private Point mPreviousPos = new Point(0, 0); // single touch events
    float diff;
    float rot;

    /**
     * 多点触控操作
     *
     * @param event
     */
    private void handleMultiTouchManipulateEvent(MotionEvent event)
    {
        switch (event.getAction() & MotionEvent.ACTION_MASK)
        {
            case MotionEvent.ACTION_POINTER_UP :
                break;
            case MotionEvent.ACTION_POINTER_DOWN :
                float x1 = event.getX(0);
                float x2 = event.getX(1);
                float y1 = event.getY(0);
                float y2 = event.getY(1);
                float delX = (x2 - x1);
                float delY = (y2 - y1);
                diff = (float) Math.sqrt((delX * delX + delY * delY));
                mStartDistance = diff;
                // float q = (delX / delY);
                mPrevRot = (float) Math.toDegrees(Math.atan2(delX, delY));
                for (ImageObject io : imgLists)
                {
                    if (io.isSelected())
                    {
                        mStartScale = io.getScale();
                        mStartRot = io.getRotation();
                        break;
                    }
                }
                break;

            case MotionEvent.ACTION_MOVE :
                x1 = event.getX(0);
                x2 = event.getX(1);
                y1 = event.getY(0);
                y2 = event.getY(1);
                delX = (x2 - x1);
                delY = (y2 - y1);
```

```
            diff = (float) Math.sqrt((delX * delX + delY * delY));
            float scale = diff / mStartDistance;
            float newscale = mStartScale * scale;
            rot = (float) Math.toDegrees(Math.atan2(delX, delY));
            float rotdiff = mPrevRot - rot;
            for (ImageObject io : imgLists)
            {
               if (io.isSelected() && newscale < 10.0f && newscale > 0.1f)
               {
                  float newrot = Math.round((mStartRot + rotdiff) / 1.0f);
                  if (Math.abs((newscale - io.getScale()) * ROTATION_STEP) >
Math
                        .abs(newrot - io.getRotation()))
                  {
                     io.setScale(newscale);
                  } else
                  {
                     io.setRotation(newrot % 360);
                  }
                  break;
               }
            }

            break;
      }
   }
   /**
    * 获取选中的对象 ImageObject
    *
    * @return
    */
   private ImageObject getSelected()
   {
      for (ImageObject ibj : imgLists)
      {
         if (ibj.isSelected())
         {
            return ibj;
         }
      }
      return null;
   }

   private long selectTime = 0;
   /**
    * 单点触控操作
    *
    * @param event
    */
   private void handleSingleTouchManipulateEvent(MotionEvent event)
   {

      long currentTime = 0;
      switch (event.getAction())
      {
         case MotionEvent.ACTION_DOWN :

            mMovedSinceDown = false;
            mResizeAndRotateSinceDown = false;
            int selectedId = -1;

            for (int i = imgLists.size() - 1; i >= 0; --i)
```

```
            {
                ImageObject io = imgLists.get(i);
                if (io.contains(event.getX(), event.getY())
                        || io.pointOnCorner(event.getX(), event.getY(),
                                OperateConstants.RIGHTBOTTOM)
                        || io.pointOnCorner(event.getX(), event.getY(),
                                OperateConstants.LEFTTOP))
                {
                    io.setSelected(true);
                    imgLists.remove(i);
                    imgLists.add(io);
                    selectedId = imgLists.size() - 1;
                    currentTime = System.currentTimeMillis();
                    if (currentTime - selectTime < 300)
                    {
                        if (myListener != null)
                        {
                            if (getSelected().isTextObject())
                            {
                                myListener
                                        .onClick((TextObject) getSelected());
                            }
                        }
                    }
                    selectTime = currentTime;
                    break;
                }
            }
            if (selectedId < 0)
            {
                for (int i = imgLists.size() - 1; i >= 0; --i)
                {
                    ImageObject io = imgLists.get(i);
                    if (io.contains(event.getX(), event.getY())
                            || io.pointOnCorner(event.getX(), event.getY(),
                                    OperateConstants.RIGHTBOTTOM)
                            || io.pointOnCorner(event.getX(), event.getY(),
                                    OperateConstants.LEFTTOP))
                    {
                        io.setSelected(true);
                        imgLists.remove(i);
                        imgLists.add(io);
                        selectedId = imgLists.size() - 1;
                        break;
                    }
                }
            }
            for (int i = 0; i < imgLists.size(); ++i)
            {
                ImageObject io = imgLists.get(i);
                if (i != selectedId)
                {
                    io.setSelected(false);
                }
            }

            ImageObject io = getSelected();
            if (io != null)
            {
                if (io.pointOnCorner(event.getX(), event.getY(),
                        OperateConstants.LEFTTOP))
                {
```

```
                imgLists.remove(io);
            } else if (io.pointOnCorner(event.getX(), event.getY(),
                  OperateConstants.RIGHTBOTTOM))
            {
               mResizeAndRotateSinceDown = true;
               float x = event.getX();
               float y = event.getY();
               float delX = x - io.getPoint().x;
               float delY = y - io.getPoint().y;
               diff = (float) Math.sqrt((delX * delX + delY * delY));
               mStartDistance = diff;
               mPrevRot = (float) Math.toDegrees(Math
                     .atan2(delX, delY));
               mStartScale = io.getScale();
               mStartRot = io.getRotation();
            } else if (io.contains(event.getX(), event.getY()))
            {
               mMovedSinceDown = true;
               mPreviousPos.x = (int) event.getX();
               mPreviousPos.y = (int) event.getY();
            }
         }
         break;

      case MotionEvent.ACTION_UP :

         mMovedSinceDown = false;
         mResizeAndRotateSinceDown = false;

         break;

      case MotionEvent.ACTION_MOVE :
         // Log.i("jarlen"," 移动了");
         // 移动
       if (mMovedSinceDown)
         {
            int curX = (int) event.getX();
            int curY = (int) event.getY();
            int diffX = curX - mPreviousPos.x;
            int diffY = curY - mPreviousPos.y;
            mPreviousPos.x = curX;
            mPreviousPos.y = curY;
            io = getSelected();
            Point p = io.getPosition();
            int x = p.x + diffX;
            int y = p.y + diffY;
            if (p.x + diffX >= mCanvasLimits.left
                  && p.x + diffX <= mCanvasLimits.right
                  && p.y + diffY >= mCanvasLimits.top
                  && p.y + diffY <= mCanvasLimits.bottom)
               io.moveBy((int) (diffX), (int) (diffY));
         }
         // 旋转和缩放
       if (mResizeAndRotateSinceDown)
         {
            io = getSelected();
            float x = event.getX();
            float y = event.getY();
            float delX = x - io.getPoint().x;
            float delY = y - io.getPoint().y;
            diff = (float) Math.sqrt((delX * delX + delY * delY));
            float scale = diff / mStartDistance;
```

```
                float newscale = mStartScale * scale;
                rot = (float) Math.toDegrees(Math.atan2(delX, delY));
                float rotdiff = mPrevRot - rot;
                if (newscale < 10.0f && newscale > 0.1f)
                {
                   float newrot = Math.round((mStartRot + rotdiff) / 1.0f);
                   if (Math.abs((newscale - io.getScale()) * ROTATION_STEP) >
Math
                         .abs(newrot - io.getRotation()))
                   {
                      io.setScale(newscale);
                   } else
                   {
                      io.setRotation(newrot % 360);
                   }
                }
             }
             break;
       }

       cancelLongPress();

    }
    /**
     * 循环画图像
     *
     * @param canvas
     */
    private void drawImages(Canvas canvas)
    {
       for (ImageObject ad : imgLists)
       {
          if (ad != null)
          {
             ad.draw(canvas);
          }
       }
    }

    /**
     * 向外部提供双击监听事件（双击弹出自定义对话框编辑文字）
     */
    MyListener myListener;

    public void setOnListener(MyListener myListener)
    {
       this.myListener = myListener;
    }

    public interface MyListener
    {
       public void onClick(TextObject tObject);
    }
}
```

3.2 Graph Processing Component

The app allows the following adjustments:

a.Photo enhancement

-Contrast

-Saturation

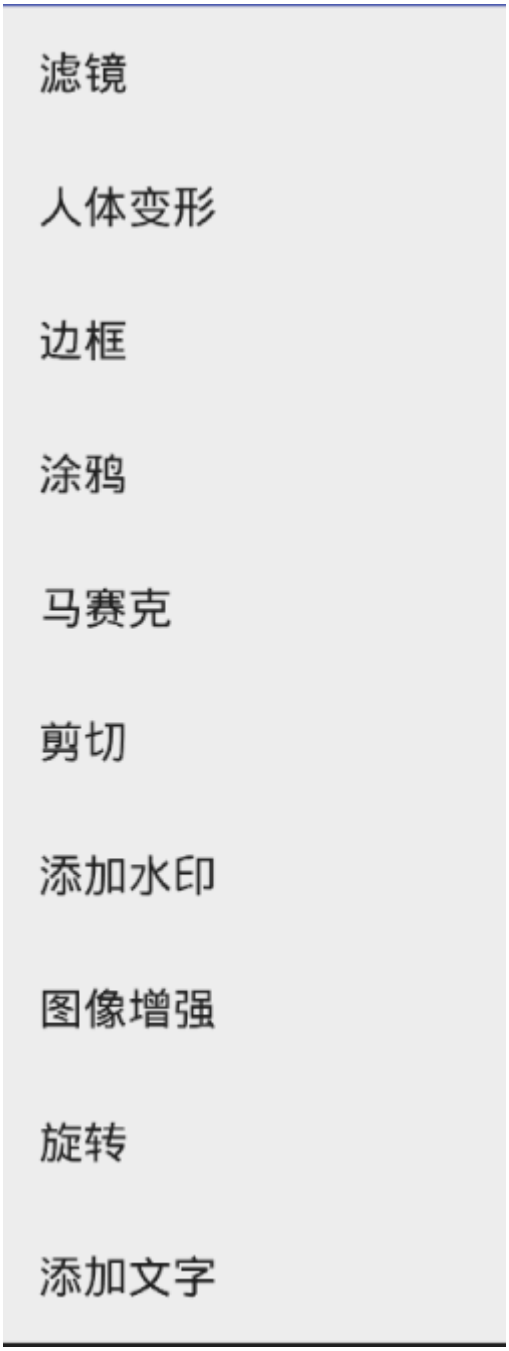
-Brightness

b.Cropping and Rotation

c.Filters

d.Warp

e.Add frames, mosaics, doodling, texts



[funtions]

4 Algorithm Description

a. About filters

In this section, we studied the effects and parameters of a couple of popular filters. After looking through data and files and comparing the slight differences between different methods, we adopted the following algorithms and methods as the base of our app.

```
jintArray brown(JNIEnv *env, jobject obj, jintArray srcPixels, jint width, jint height,
                jfloat factor) {
    jint * cbuf;
    cbuf = env->GetIntArrayElements(srcPixels, JNI_FALSE);

    if (cbuf == NULL) {
        return 0;
    }

    int size = width * height;
    int buffTemp[size];

    float brownMatrix[4][4] = {
            {0.3588, 0.299, 0.2392, 0},
            {0.7044, 0.587, 0.4696, 0},
            {0.1368, 0.114, 0.0912, 0},
            {0,      0,     0,      1}
    };

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            int currentColor = cbuf[j * width + i];

            int a = alpha(currentColor);
            int r = red(currentColor);
            int g = green(currentColor);
            int b = blue(currentColor);

            int colorTemp[4];
            int resultColor[4] = {0};


            colorTemp[0] = r;
            colorTemp[1] = g;
            colorTemp[2] = b;
            colorTemp[3] = a;

            matrix1X4(colorTemp, brownMatrix, resultColor);
```

```
            matrix1X4(colorTemp, brownMatrix, resultColor);

            int temp = ARGB(resultColor[3], resultColor[0], resultColor[1], resultColor[2]);

            buffTemp[j * width + i] = temp;

        }
    }

    jintArray result = env->NewIntArray(size);
    env->SetIntArrayRegion(result, 0, size, buffTemp);
    env->ReleaseIntArrayElements(srcPixels, cbuf, 0);
    return result;
}
```

filter- brown

```
jintArray nostalgic(JNIEnv *env, jobject obj, jintArray srcPixels, jint width, jint height
                    jfloat factor) {
    jint * pixels = NULL;

    pixels = env->GetIntArrayElements(srcPixels, JNI_FALSE);
    if (pixels == NULL) {
        return srcPixels;
    }

    int size = width * height;
    int result[size];

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            int index = i * height + j;

            int a0 = alpha(pixels[index]);
            int r0 = red(pixels[index]);
            int g0 = green(pixels[index]);
            int b0 = blue(pixels[index]);

            int r = 0.393 * r0 + 0.769 * g0 + 0.189 * b0;
            int g = 0.349 * r0 + 0.686 * g0 + 0.168 * b0;
            int b = 0.272 * r0 + 0.534 * g0 + 0.131 * b0;

            r = Min(255, Max(r, 0));
            g = Min(255, Max(g, 0));
            b = Min(255, Max(b, 0));

            result[index] = ARGB(a0, r, g, b);
        }
    }

    jintArray resultArray = env->NewIntArray(size);
    env->SetIntArrayRegion(resultArray, 0, size, result);
    env->ReleaseIntArrayElements(srcPixels, pixels, 0);
    return resultArray;
}
```

[nostalgia]

```
jintArray sketchPencil(JNIEnv *env, jobject obj, jintArray srcPixels, jint width, jint height) {

    jint * cbuf;
    cbuf = env->GetIntArrayElements(srcPixels, JNI_FALSE);
    if (cbuf == NULL) {
        return 0;
    }

    int newSize = width * height;
    jint rbuf[newSize]; // 新图像像素值

    // 先对图象的像素处理成灰度颜色后再取反

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {

            int curr_color = cbuf[j * width + i];
            int color_row = cbuf[j * width + i + 1];
            int color_col = cbuf[(j + 1) * width + i];

            //原图像素
            int r0 = red(curr_color);
            int g0 = green(curr_color);
            int b0 = blue(curr_color);

            // 同行像素
            int r1 = red(color_row);
            int g1 = green(color_row);
            int b1 = blue(color_row);

            //同列像素
            int r2 = red(color_col);
            int g2 = green(color_col);
            int b2 = blue(color_col);
```

```
            //梯度处理，产生霓虹效果
            int r = 2 * sqrt((r0 - r1) * (r0 - r1) + (r0 - r2) * (r0 - r2));
            int g = 2 * sqrt((g0 - g1) * (g0 - g1) + (g0 - g2) * (g0 - g2));
            int b = 2 * sqrt((b0 - b1) * (b0 - b1) + (b0 - b2) * (b0 - b2));

            //反色处理
            r = 255 - r;
            g = 255 - g;
            b = 255 - b;

            // 灰度化
            r = 0.299 * r + 0.587 * g + 0.114 * b;
            g = r;
            b = r;

            int a = alpha(curr_color);
            int modif_color = ARGB(a, r, g, b);
            rbuf[j * width + i] = modif_color;
        }
    }

    jintArray result = env->NewIntArray(newSize);
    env->SetIntArrayRegion(result, 0, newSize, rbuf);
    env->ReleaseIntArrayElements(srcPixels, cbuf, 0);
    return result;
}
```

filter- sketchpencil

There’re other filters alike in our app. Due to the length limit of this report, other realization of filters are omitted.

b. About enhancement

```
public class PhotoEnhance
{

   public final int Enhance_Saturation = 0;
   public final int Enhance_Brightness = 1;
   public final int Enhance_Contrast = 2;

   private Bitmap mBitmap;

   private float saturationNum = 1.0f;
   private float brightNum = 0.0f;
   private float contrastNum = 1.0f;

   public PhotoEnhance()
   {

   }

   public PhotoEnhance(Bitmap bitmap)
   {
      this.mBitmap = bitmap;
   }

   public float getSaturation()
   {
      return saturationNum;
   }

   /**
    * 设置饱和度 ( 0 ~ 2)
    *
    * @param saturationNum
    *             (范围 :0 ~ 255)
    *
    */
   public void setSaturation(int saturationNum)
   {
      this.saturationNum = (float) (saturationNum * 1.0f / 128);
   }

   public float getBrightness()
   {
      return brightNum;
   }

   /**
    * 设置亮度 (-128 ~ 128 )
    *
    * @param brightNum
    *             (范围: 0 ~ 255)
    *
    */
   public void setBrightness(int brightNum)
   {
      this.brightNum = brightNum - 128;
   }

   public float getContrast()
   {
      return contrastNum;
```

```
    }

    /**
     * 设置对比度 (0.5 ~ 1.5)
     *
     * @param contrastNum
     *            (范围 : 0 ~ 255)
     *
     */
    public void setContrast(int contrastNum)
    {
        this.contrastNum = (float) ((contrastNum / 2 + 64) / 128.0);
    }

    private ColorMatrix mAllMatrix = null;
    private ColorMatrix saturationMatrix = null;
    private ColorMatrix contrastMatrix = null;
    private ColorMatrix brightnessMatrix = null;

    public Bitmap handleImage(int type)
    {

        Bitmap bmp = Bitmap.createBitmap(mBitmap.getWidth(),
              mBitmap.getHeight(), Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(bmp);
        Paint paint = new Paint();
        paint.setAntiAlias(true);

        if (mAllMatrix == null)
        {
            mAllMatrix = new ColorMatrix();
        }

        /* 饱和度矩阵 */
        if (saturationMatrix == null)
        {
            saturationMatrix = new ColorMatrix();
        }

        /* 对比度矩阵 */
        if (contrastMatrix == null)
        {
            contrastMatrix = new ColorMatrix();
        }

        /* 亮度矩阵 */
        if (brightnessMatrix == null)
        {
            brightnessMatrix = new ColorMatrix();
        }

        switch (type)
        {
            case Enhance_Saturation :
                saturationMatrix.reset();
                saturationMatrix.setSaturation(saturationNum);
                break;

            case Enhance_Brightness :
                brightnessMatrix.reset();
                brightnessMatrix.set(new float[]{1, 0, 0, 0, brightNum, 0, 1,
                      0, 0, brightNum, 0, 0, 1, 0, brightNum, 0, 0, 0, 1, 0});
```

```
            break;
        case Enhance_Contrast :

            /* 在亮度不变的情况下，提高对比度必定要降低亮度 */

            float regulateBright = 0;
            regulateBright = (1 - contrastNum) * 128;

            contrastMatrix.reset();
            contrastMatrix.set(new float[]{contrastNum, 0, 0, 0,
                  regulateBright, 0, contrastNum, 0, 0, regulateBright,
                  0, 0, contrastNum, 0, regulateBright, 0, 0, 0, 1, 0});
            break;

        default :
            break;
      }

      mAllMatrix.reset();
      mAllMatrix.postConcat(saturationMatrix);
      mAllMatrix.postConcat(brightnessMatrix);
      mAllMatrix.postConcat(contrastMatrix);

      paint.setColorFilter(new ColorMatrixColorFilter(mAllMatrix));
      canvas.drawBitmap(mBitmap, 0, 0, paint);
      return bmp;

   }

}
```

c. About rotation

```
public class PhotoUtils
{

    /**
     * 图片旋转
     * @param bit
     * 旋转原图像
     *
     * @param degrees
     * 旋转度数
     *
     * @return
     * 旋转之后的图像
     *
     */
    public static Bitmap rotateImage(Bitmap bit, int degrees)
    {
        Matrix matrix = new Matrix();
        matrix.postRotate(degrees);
        Bitmap tempBitmap = Bitmap.createBitmap(bit, 0, 0, bit.getWidth(),
                bit.getHeight(), matrix, true);
        return tempBitmap;
    }
```

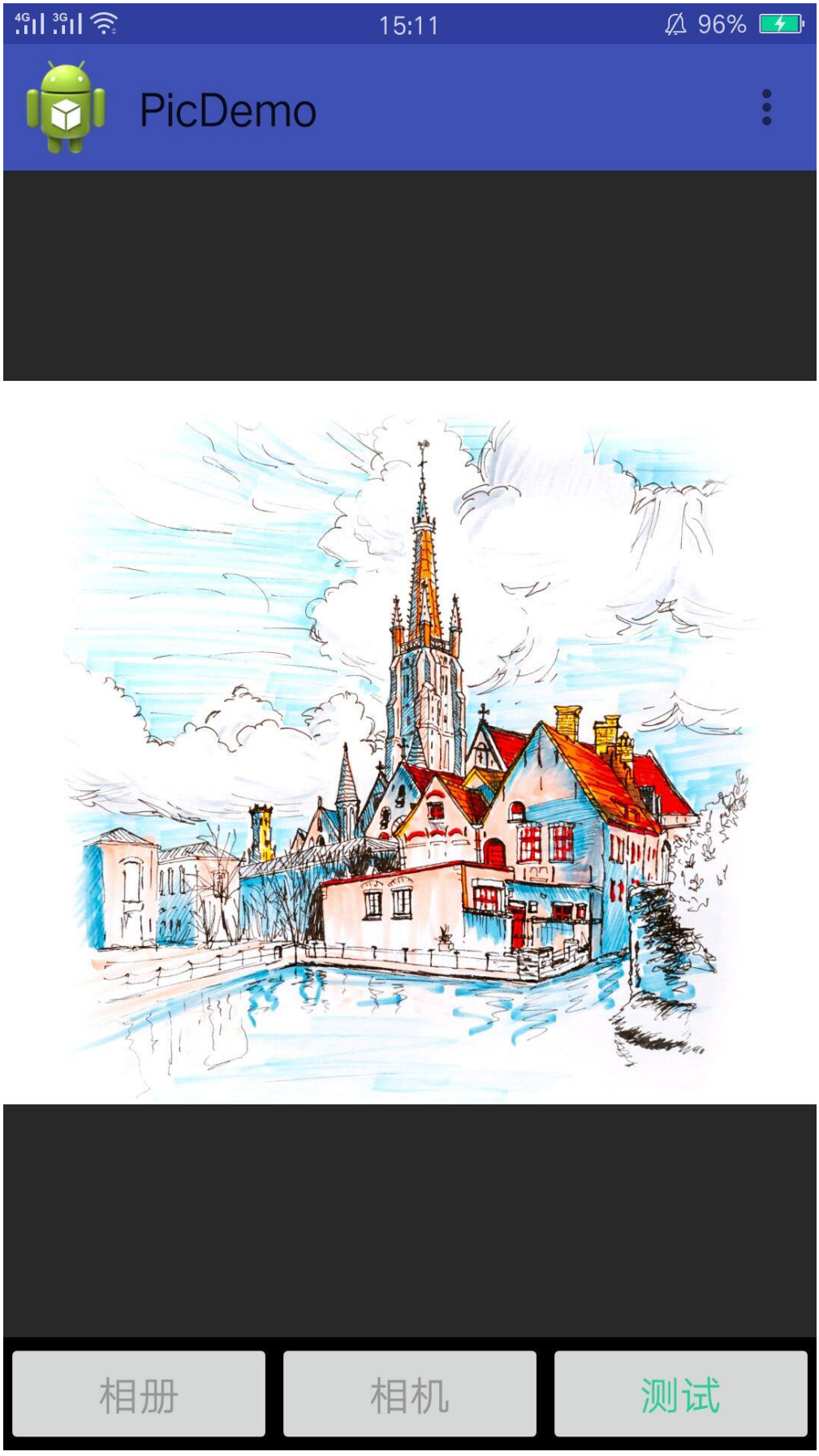
```
    /**
     * 翻转图像
     *
     * @param bit
     * 翻转原图像
     *
     * @param x
     * 翻转X轴
     *
     * @param y
     * 翻转Y轴
     *
     * @return
     * 翻转之后的图像
     *
     * 说明:
     * (1,-1)上下翻转
     * (-1,1)左右翻转
     *
     */
    public static Bitmap reverseImage(Bitmap bit,int x,int y)
    {
        Matrix matrix = new Matrix();
        matrix.postScale(x, y);

        Bitmap tempBitmap = Bitmap.createBitmap(bit, 0, 0, bit.getWidth(),
                bit.getHeight(), matrix, true);
        return tempBitmap;
    }
}
```

Basically, the essence of algorithm can be boiled down to two parts, procession of matrices and change of bitmap. This requires a bit of linear algebra skills. We develop these algorithm through trial and error though, as we actually haven't have a detailed understanding of the underlying principles of graphic processing. Yet after lengthy observation, application of these theories is no longer a problem.

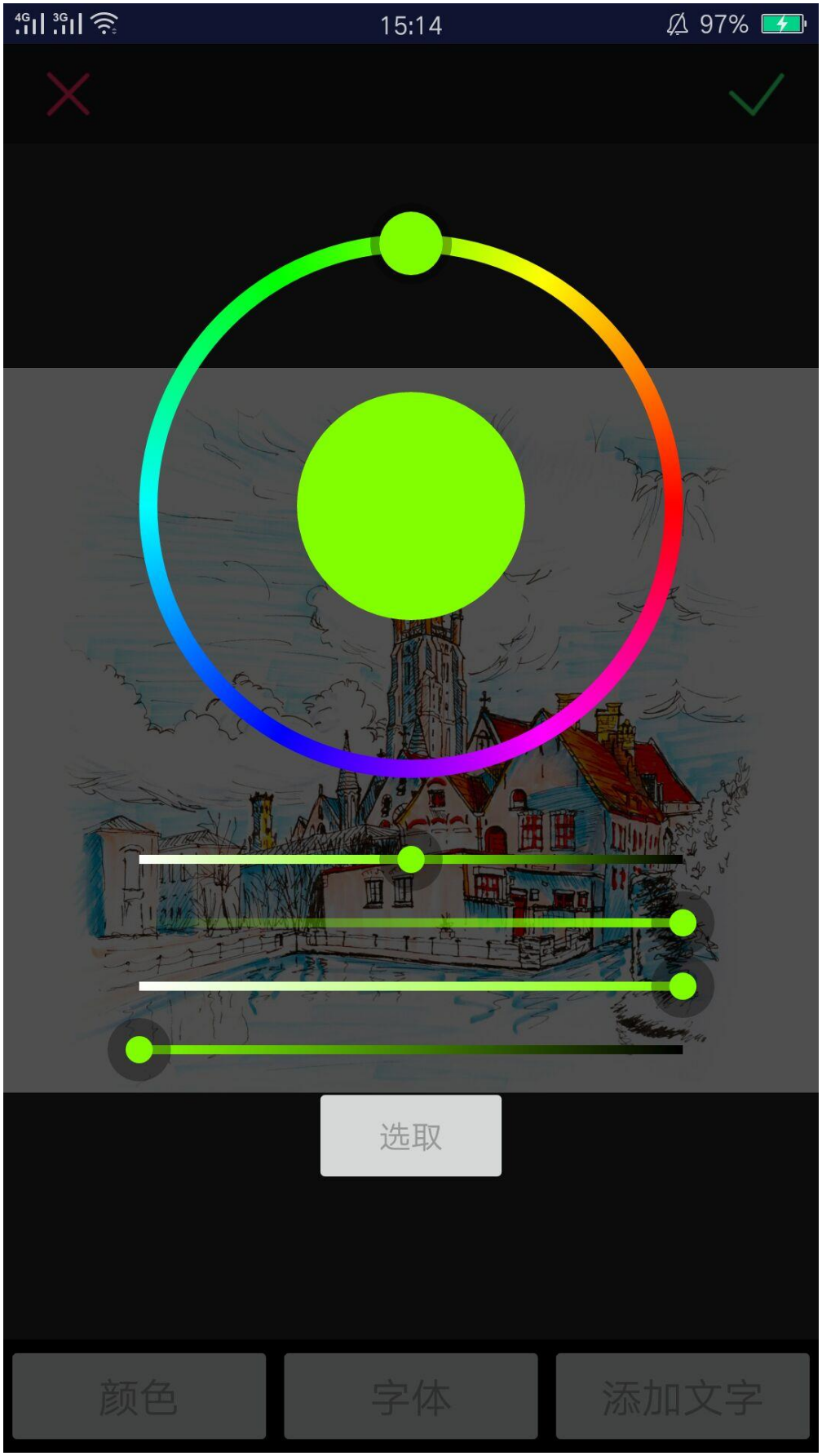
4 Screenshots

main window

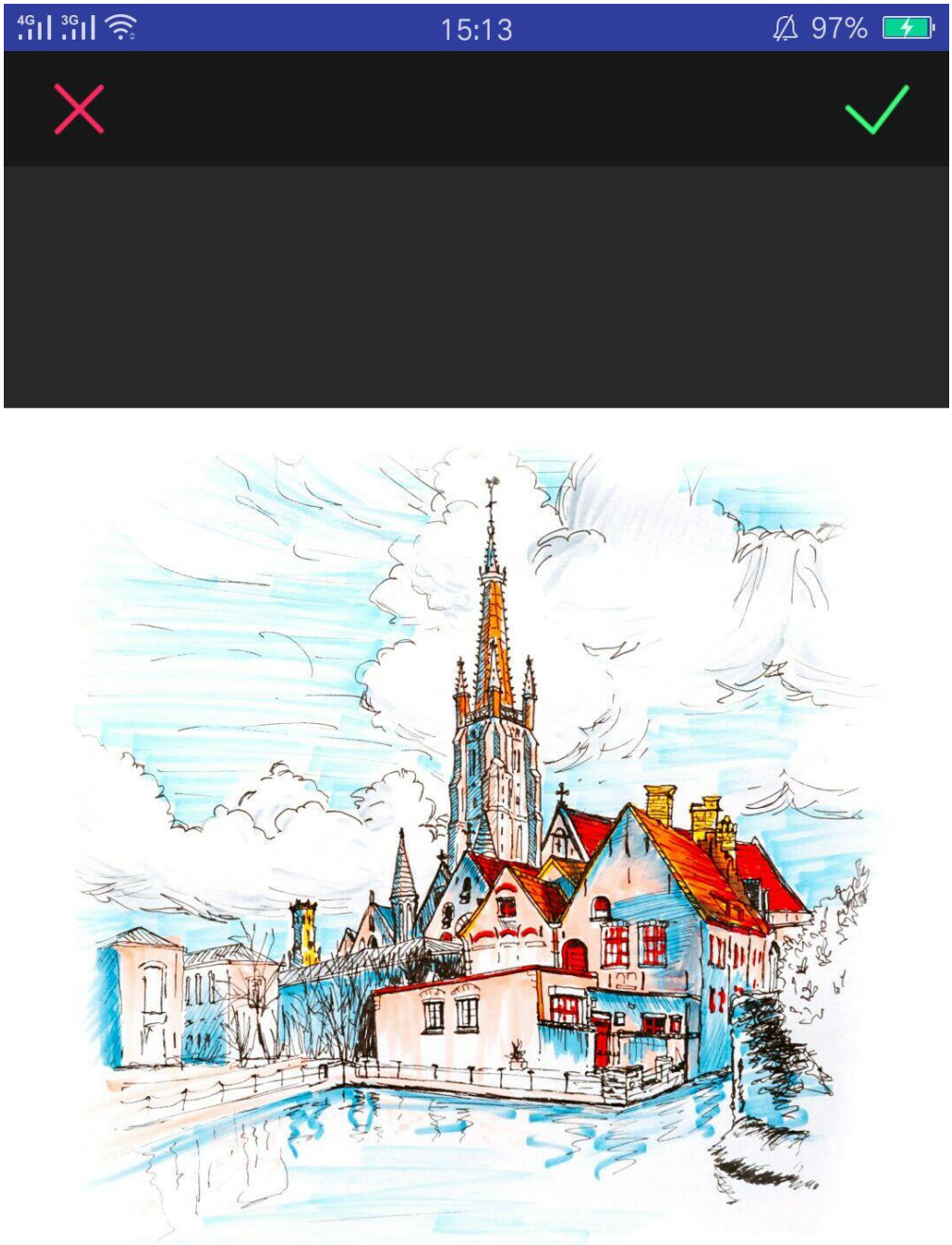


original pic

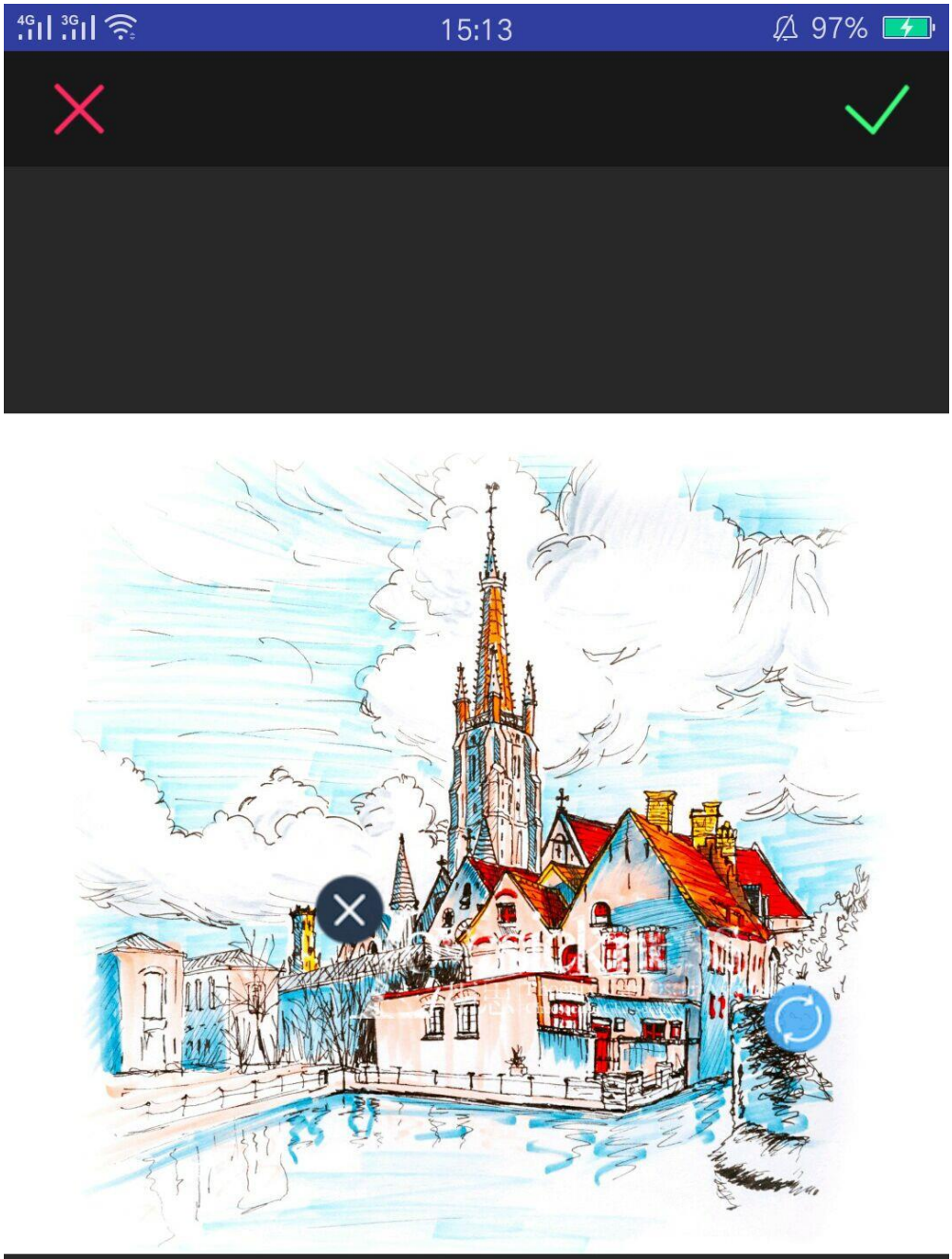
add text

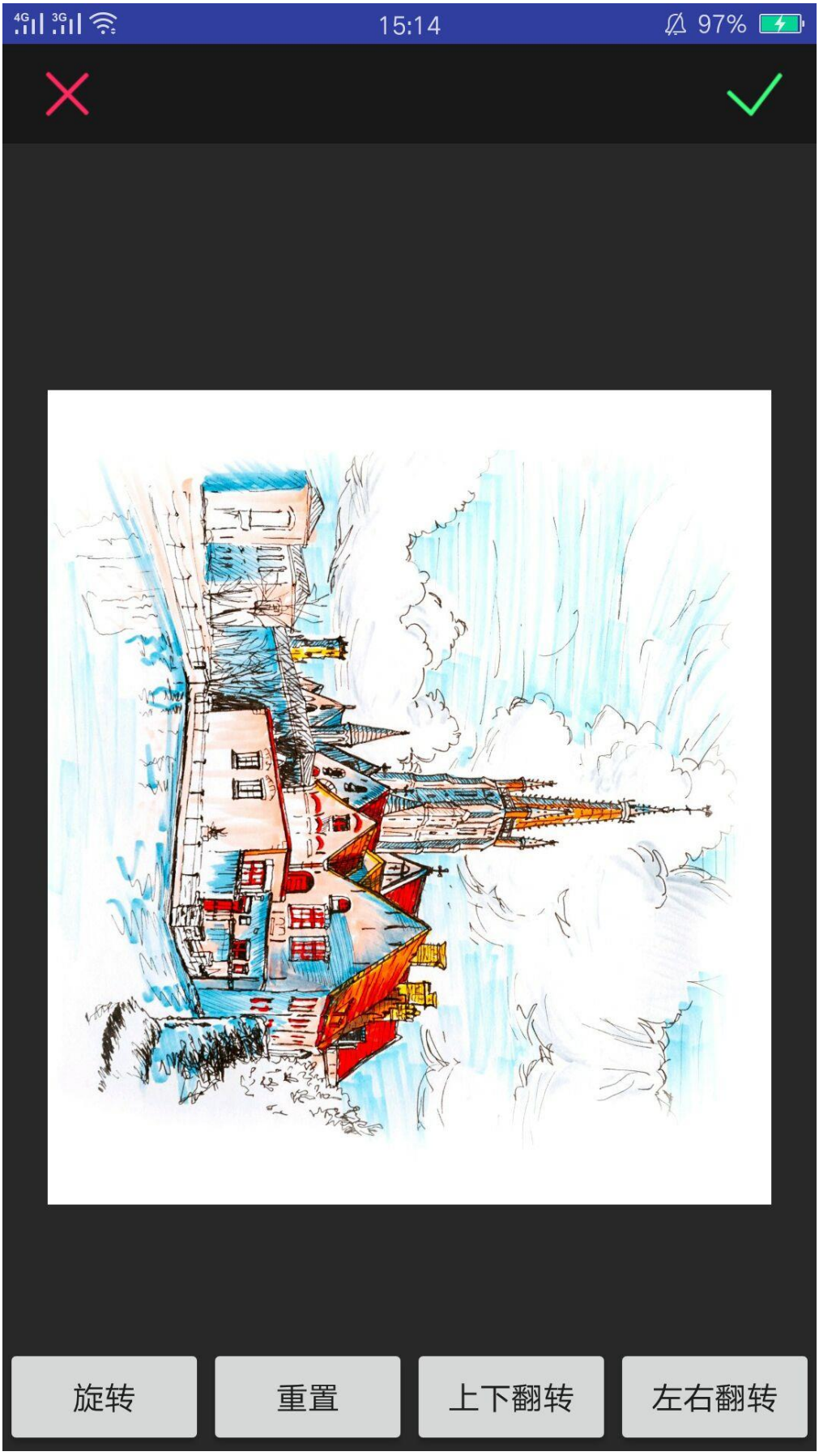


options for colors of the text to be added

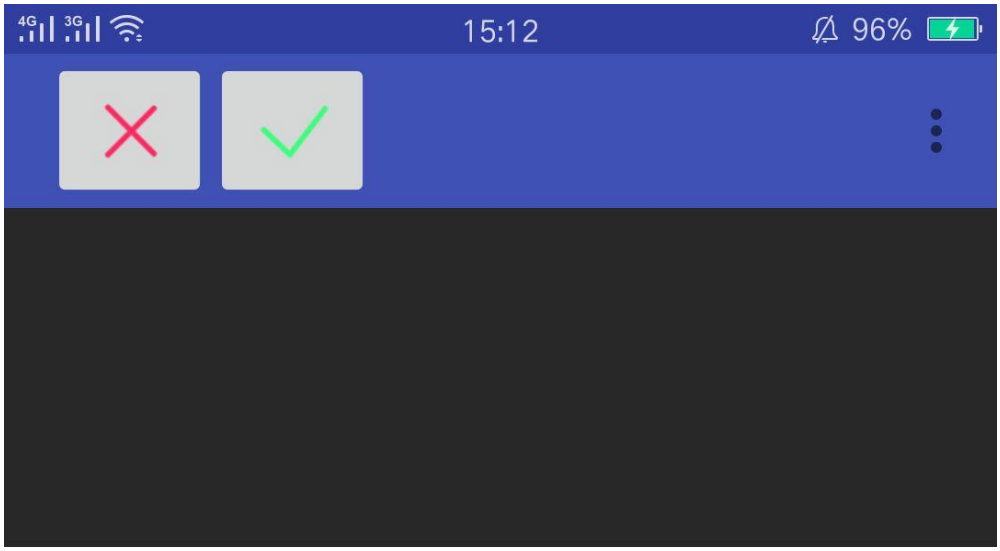


enhancement

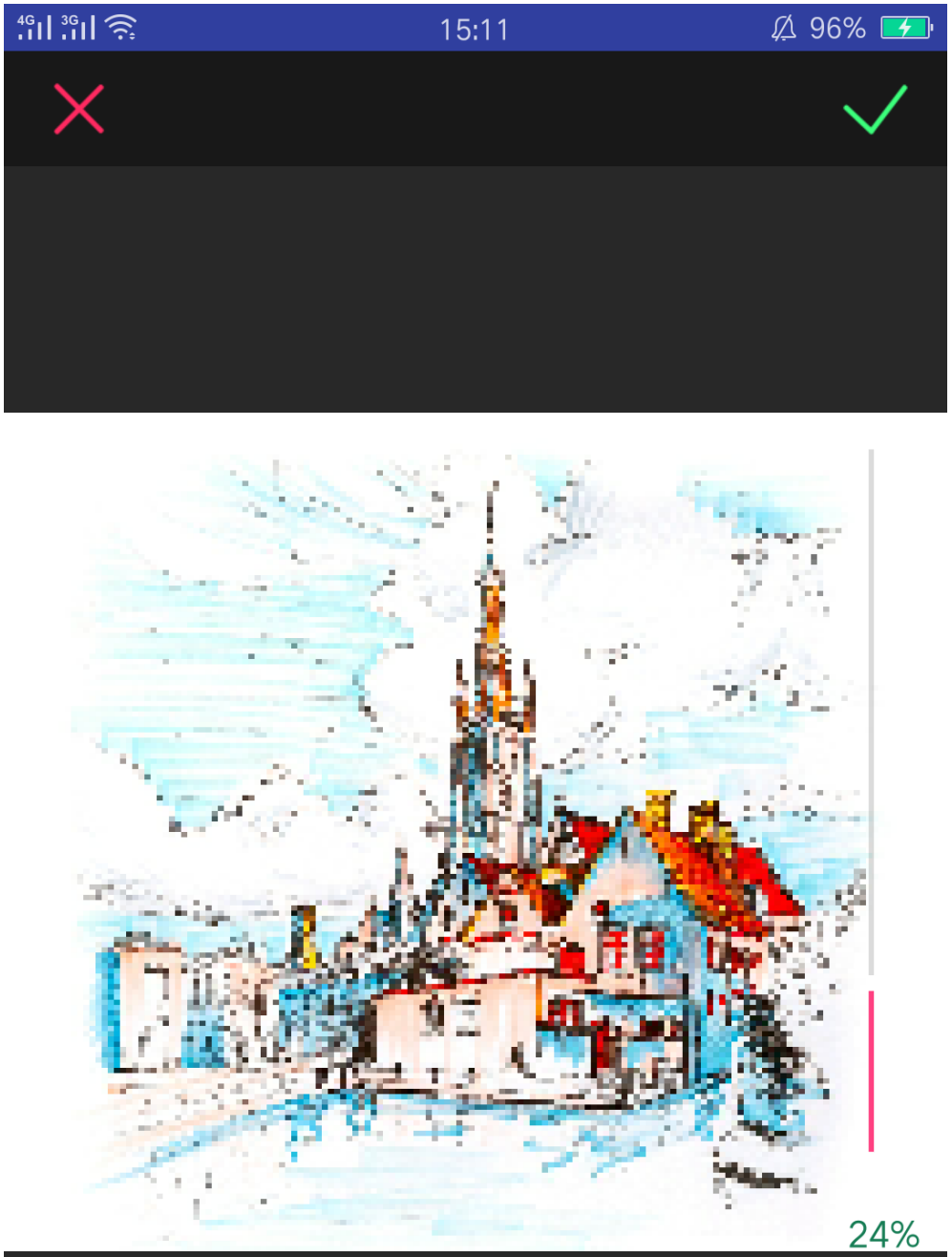




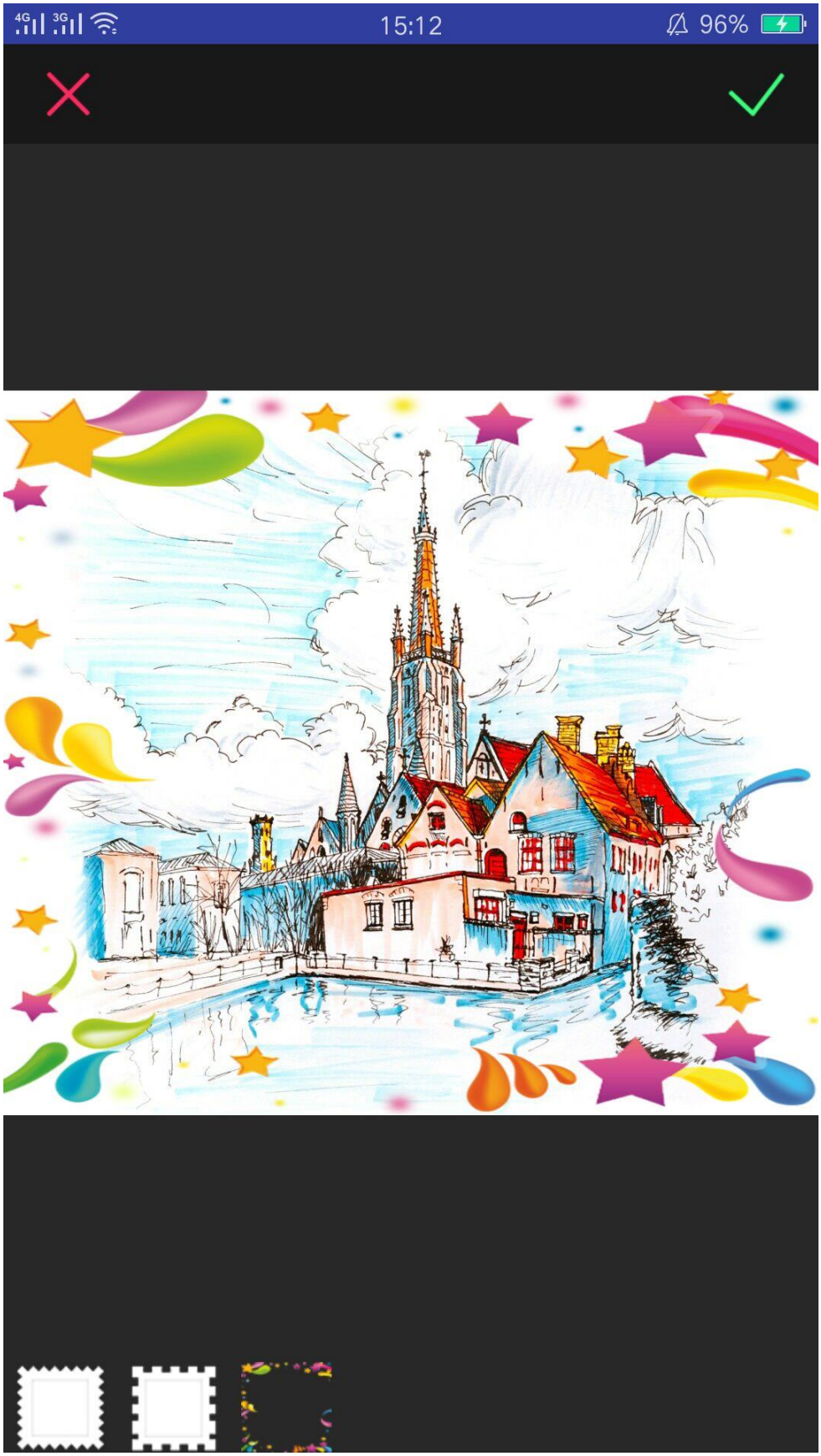
rotation



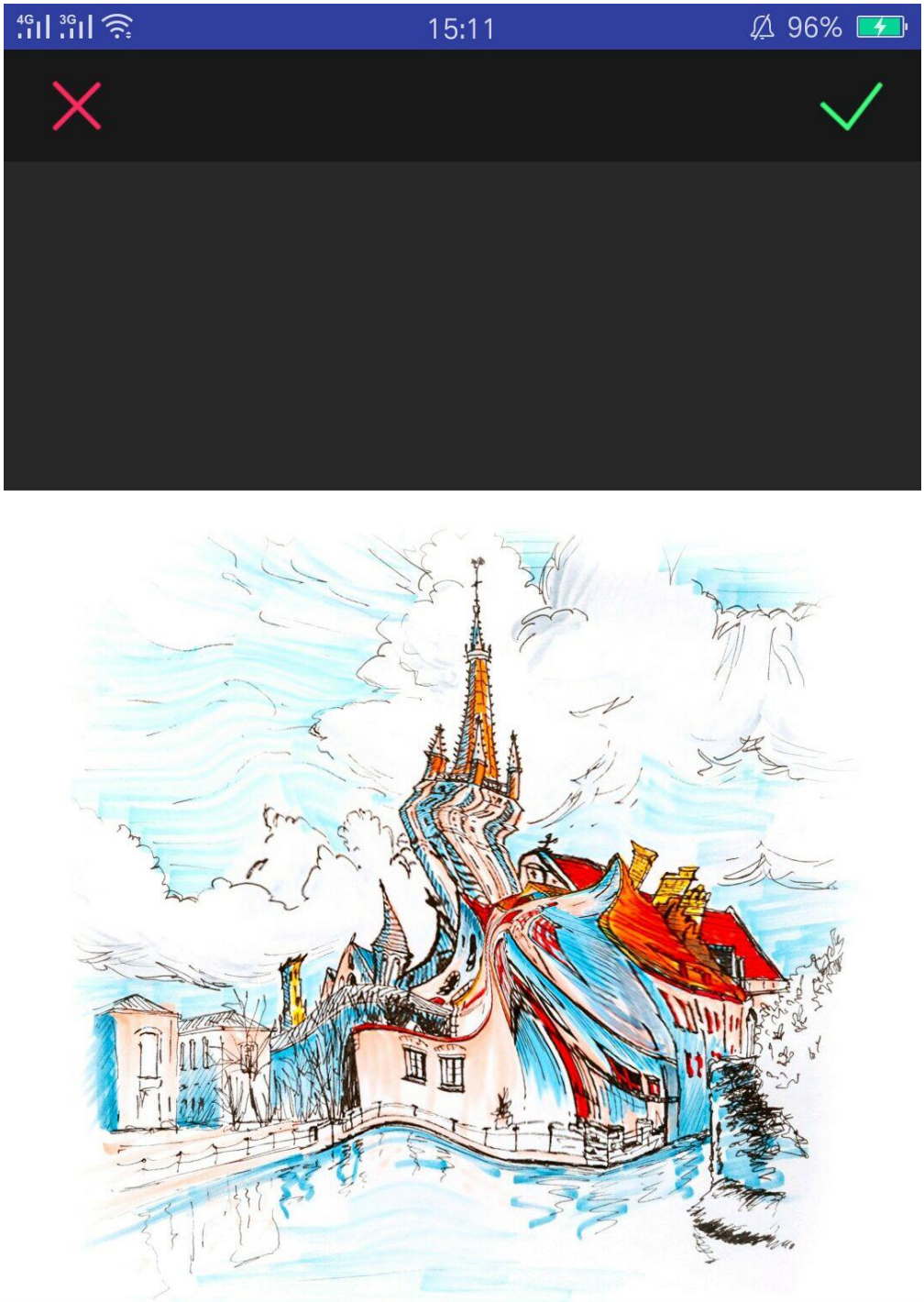
graffiti



filters



frames



5 Conclusion

The idea of developing a graph processing app comes from a teammates' love for ps. Then we thought of it as a nice start of app development to create a small and well-functioned photo-editing app of our own.

The process of building up this app is a pretty arduous journey. The project itself, as a task that involves a whole spectrum of exploration and endeavour, including learning java, ndk development setup, graph-processing algorithm learning, user interface designing, event listeners, and final integration, proves to be both challenging and inspiring. During the process, not only have we laid the ground for further learning of algorithm, UI and android app development, but also achieved a sense bordering on doing research and building up a software on our own.

Hardship twisted in the whole process of this project, from getting to understand android app, learning basic mechanism of Android Studio and the structure of its files, grammar of java, to having a grasp of graphic processing algorithm. Yet efforts did pay off. There're still, though, things to be revised and enhanced in every file of our project, for example, the 'test' button is virtually unnecessary. So adjustments are expected in the days that follow.