大眼，瘦臉，大長腿美顏特效在Android 上的實現

北斗星_And 安卓開發 今天

> **作者：北斗星_And**
> **鏈接：https://juejin.im/post/5d5ff49bf265da03d42fae9c**

序言

本篇文章是代碼撸彩妝的第二篇， 主要介紹在Android上怎麼進行圖片的局部變形，並實現抖音上比較火的大眼，瘦臉，大長腿特效.

在開始之前我們先來回顧上一篇的主要內容.

使用代碼畫一半的效果如下

```
public enum Region {

   FOUNDATION("粉底"),
   BLUSH("腮红"),
   LIP("唇彩"),
   BROW("眉毛"),

   EYE_LASH("睫毛"),
   EYE_CONTACT("美瞳"),
   EYE_DOUBLE("双眼皮"),
   EYE_LINE("眼线"),
   EYE_SHADOW("眼影");

   private String name;
   Region(String name) {
      this.name = name;
   }
}
```

使用代碼畫出各種效果. 上一篇的文章地址 Android：讓你的“女神”逆襲，代碼擼彩妝（畫妝）
https://juejin.im/post/5d4bd2536fb9a06b1d212f72

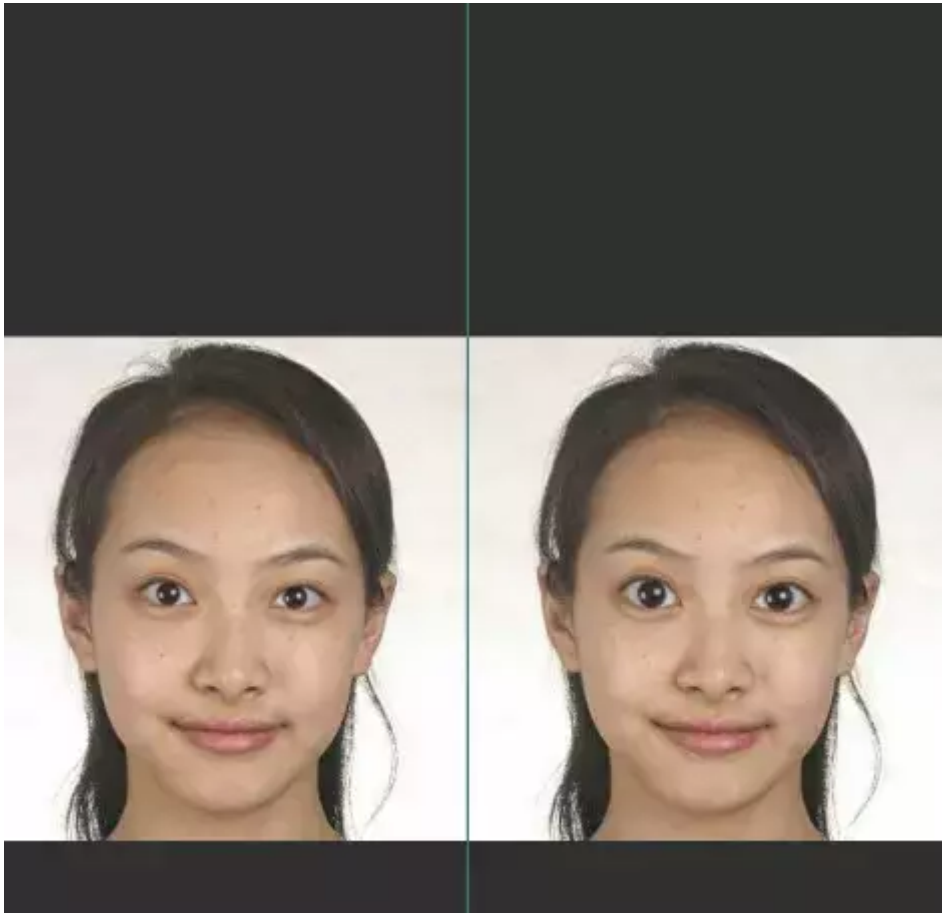
上一篇和本篇的代碼所在地址一致，都已經託管到github,如果你喜歡，歡迎給一個star，謝謝

https://github.com/DingProg/Makeup

現在開始我們今天的主題，人體（圖像）的局部變形，如果要直接看效果的話，可以點擊目錄快速滑到效果區域.

大眼

效果



實現

圖片局部縮放原理

我們知道，圖片的放大縮小，是比較容易的事，相應的庫已經封裝好了，可以直接使用（我們並不需要關注圖形放大縮小的插值處理等). 但是圖片的局部放大縮小，並沒有直接封裝好，比如Android裡面的bitmap，並沒有直接局部處理放大縮小的API.

那我們先來看一下什麼是圖形的局部縮放?

局部的縮放，我們可以想像成中心點被縮放的比例比較小，而邊緣的地方被縮放的比例很小，或者邊界區域幾乎沒有變化，這樣就可以達到一種平滑的效果。如果直接只對選中的圓形區域，變化的話，那邊緣就變成了斷裂式的縮放.

借用1993年的一篇博士論文 Interactive Image Warping 對局部圖片進行縮放
http://www.gson.org/thesis/warping-thesis.pdf

$$f_s(r) = \left(1 - \left(\frac{r}{r_{\max}} - 1\right)^2 a\right) r$$

其中a為縮放因子，當a=0時，不縮放。

代碼實現

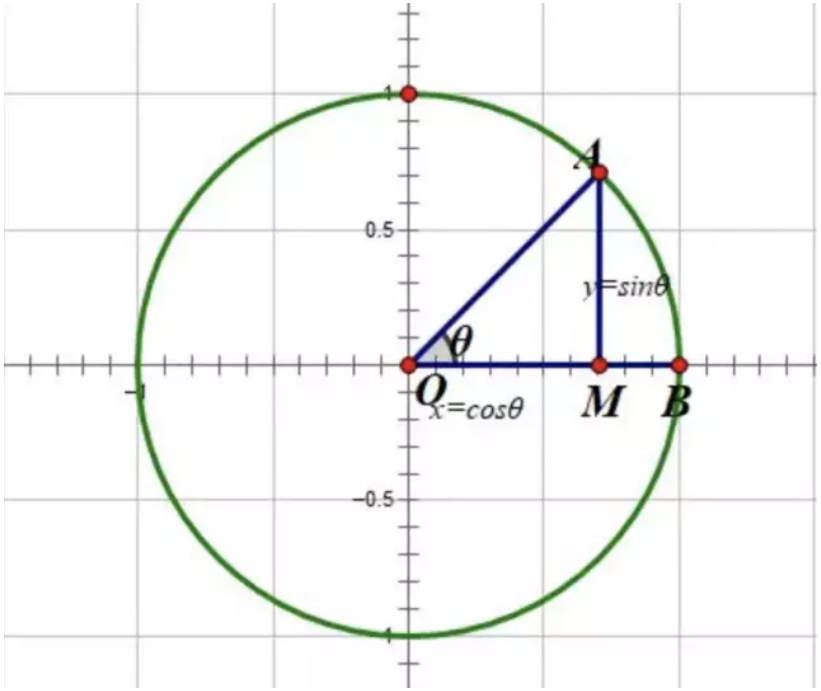
既然要讓眼睛放大，那麼我們就把對應的近圓心的點的值賦給遠心點。按照論文裡所提到的思路，進行部分修改，實現如下.

```
/**
 * 眼睛放大算法
 * @param bitmap     原来的bitmap
 * @param centerPoint 放大中心点
 * @param radius     放大半径
 * @param sizeLevel   放大力度 [0,4]
 * @return 放大眼睛后的图片
 */
public static Bitmap magnifyEye(Bitmap bitmap, Point centerPoint, int radius, float sizeLevel) {
    TimeAopUtils.start();
```

```
        Bitmap dstBitmap = bitmap.copy(Bitmap.Config.RGB_565, true);
        int left = centerPoint.x - radius < 0 ? 0 : centerPoint.x - radius;
        int top = centerPoint.y - radius < 0 ? 0 : centerPoint.y - radius;
        int right = centerPoint.x + radius > bitmap.getWidth() ? bitmap.getWidth() - 1 : centerPoint.x + radius;
        int bottom = centerPoint.y + radius > bitmap.getHeight() ? bitmap.getHeight() - 1 : centerPoint.y + radius;
        int powRadius = radius * radius;

        int offsetX, offsetY, powDistance, powOffsetX, powOffsetY;

        int disX, disY;

        //当为负数时，为缩小
        float strength = (5 + sizeLevel * 2) / 10;

        for (int i = top; i <= bottom; i++) {
          offsetY = i - centerPoint.y;
          for (int j = left; j <= right; j++) {
            offsetX = j - centerPoint.x;
            powOffsetX = offsetX * offsetX;
            powOffsetY = offsetY * offsetY;
            powDistance = powOffsetX + powOffsetY;

            if (powDistance <= powRadius) {
              double distance = Math.sqrt(powDistance);
              double sinA = offsetX / distance;
              double cosA = offsetY / distance;

              double scaleFactor = distance / radius - 1;
              scaleFactor = (1 - scaleFactor * scaleFactor * (distance / radius) * strength);

              distance = distance * scaleFactor;
              disY = (int) (distance * cosA + centerPoint.y + 0.5);
              disY = checkY(disY, bitmap);
              disX = (int) (distance * sinA + centerPoint.x + 0.5);
              disX = checkX(disX, bitmap);
              //中心点不做处理
              if (!(j == centerPoint.x && i == centerPoint.y)) {
                dstBitmap.setPixel(j, i, bitmap.getPixel(disX, disY));
              }
            }
          }
        }
        TimeAopUtils.end("eye","magnifyEye");
        return dstBitmap;
    }

    private static int checkY(int disY, Bitmap bitmap) {
```

```
        if (disY < 0) {
            disY = 0;
        } else if (disY >= bitmap.getHeight()) {
            disY = bitmap.getHeight() - 1;
        }
        return disY;
    }

    private static int checkX(int disX, Bitmap bitmap) {
        if (disX < 0) {
            disX = 0;
        } else if (disX >= bitmap.getWidth()) {
            disX = bitmap.getWidth() - 1;
        }
        return disX;
    }
```

其中里面計算縮放前後後的點，使用的是如下圖所示的計算規則計算.



有了這個方法，我們藉助人臉識別的結果，把眼睛中心部分傳入進去就可以實現自動大眼的效果了.

```
Bitmap magnifyEye = MagnifyEyeUtils.magnifyEye(bitmap,
Objects.requireNonNull(FacePoint.getLeftEyeCenter(faceJson)),
FacePoint.getLeftEyeRadius(faceJson) * 3, 3);
```

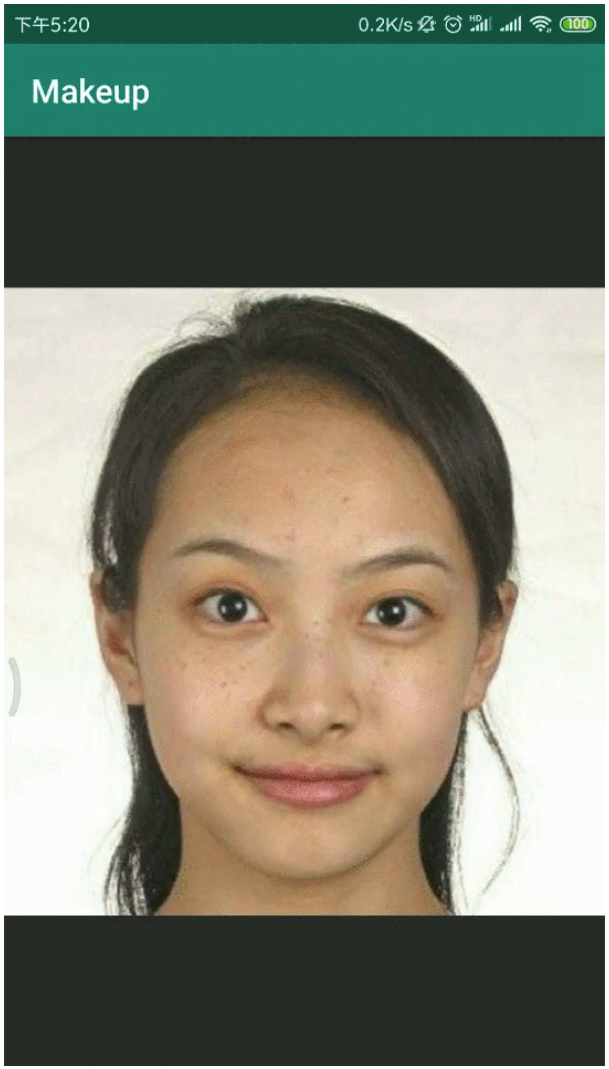
略有不足

代碼所示部分沒有使用插值（代碼直接使用了值替代，而不是使用兩個點，三個點，進行插值計算），如果放大的比例很大，可能會出現模糊的效果

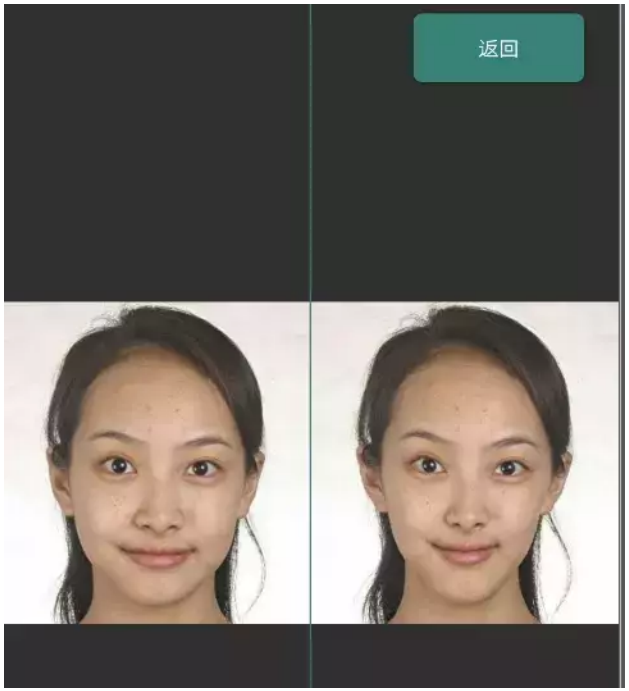
Android Bitmap直接獲取像素，效率低，正確的方式應該是一次全部獲取對應的像素，然後在數組上進行操作（考慮內容，就直接採用了每次去讀取/設置），操作完之後，在設置回去。

瘦臉

手動模式



自動模式



實現

大眼效果，使用了bitmap直接去操作像素點，效率有點低，所以在實現瘦臉和打長腿時，採用了另外的實現方式實現.

Cavans的drawBitmapMesh方法

```
public void drawBitmapMesh(@NonNull Bitmap bitmap, int meshWidth, int meshHeight,
        @NonNull float[] verts, int vertOffset, @Nullable int[] colors, int colorOffset,
        @Nullable Paint paint) {
    super.drawBitmapMesh(bitmap, meshWidth, meshHeight, verts, vertOffset, colors, colorOffset,
            paint);
}
```

這個方法，大概說的是，將圖片使用網格的方式先進行分割，然後操作這些網格，就可以讓圖片達到扭曲的效果.

代碼實現

Gif中拖動就可以進行自動瘦臉功能，這是一個自定義的View，在View上通過手勢操作，去改變那個網格，然後在調用重繪.

第一步，初始化圖片，把圖片放在View的中心

```
private void zoomBitmap(Bitmap bitmap, int width, int height) {
    if(bitmap == null) return;
    int dw = bitmap.getWidth();
    int dh = bitmap.getHeight();

    float scale = 1.0f;

    // 图片的宽度大于控件的宽度，图片的高度小于空间的高度，我们将其缩小
    if (dw > width && dh < height) {
        scale = width * 1.0f / dw;
    }

    // 图片的宽度小于控件的宽度，图片的高度大于空间的高度，我们将其缩小
    if (dh > height && dw < width) {
        scale = height * 1.0f / dh;
    }

    // 缩小值
    if (dw > width && dh > height) {
        scale = Math.min(width * 1.0f / dw, height * 1.0f / dh);
    }

    // 放大值
    if (dw < width && dh < height) {
        scale = Math.min(width * 1.0f / dw, height * 1.0f / dh);
    }

    //缩小
    if (dw == width && dh > height) {
        scale = height * 1.0f / dh;
    }
```

```
        dx = width / 2 - (int) (dw * scale + 0.5f) / 2;
        dy = height / 2 - (int) (dh * scale + 0.5f) / 2;

        mScale = scale;
        restoreVerts();
    }
```

接著初始化網格

```
//将图像分成多少格
private int WIDTH = 200;
private int HEIGHT = 200;

//交点坐标的个数
private int COUNT = (WIDTH + 1) * (HEIGHT + 1);

//用于保存COUNT的坐标
//x0, y0, x1, y1......
private float[] verts = new float[COUNT * 2];

//用于保存原始的坐标
private float[] orig = new float[COUNT * 2];
private void restoreVerts() {
    int index = 0;
    float bmWidth = mBitmap.getWidth();
    float bmHeight = mBitmap.getHeight();
    for (int i = 0; i < HEIGHT + 1; i++) {
        float fy = bmHeight * i / HEIGHT;
        for (int j = 0; j < WIDTH + 1; j++) {
            float fx = bmWidth * j / WIDTH;
            //X轴坐标 放在偶数位
            verts[index * 2] = fx;
            orig[index * 2] = verts[index * 2];
            //Y轴坐标 放在奇数位
            verts[index * 2 + 1] = fy;
            orig[index * 2 + 1] = verts[index * 2 + 1];
            index += 1;
        }
    }
    showCircle = false;
    showDirection = false;
}
```

那最後一步把這個圖片畫上去

```
@Override
protected void onDraw(Canvas canvas) {
  super.onDraw(canvas);
  if(mBitmap == null) return;
  canvas.save();
  canvas.translate(dx, dy);
  canvas.scale(mScale, mScale);
  if (isShowOrigin) {
    canvas.drawBitmapMesh(mBitmap, WIDTH, HEIGHT, orig, 0, null, 0, null);
  } else {
    canvas.drawBitmapMesh(mBitmap, WIDTH, HEIGHT, verts, 0, null, 0, null);
  }

  canvas.restore();
  if (showCircle && isEnableOperate) {
    canvas.drawCircle(startX, startY, radius, circlePaint);
    canvas.drawCircle(startX, startY, 5, directionPaint);
  }
  if (showDirection && isEnableOperate) {
    canvas.drawLine(startX, startY, moveX, moveY, directionPaint);
  }
}
```

那麼接下來，就來操作網格，然後產生一些變形的效果了.添加事件監聽

```
@Override
public boolean onTouchEvent(MotionEvent event) {
  if (!isEnableOperate) return true;
  switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
      //绘制变形区域
      startX = event.getX();
      startY = event.getY();
      showCircle = true;
      invalidate();
      break;
    case MotionEvent.ACTION_MOVE:
      //绘制变形方向
      moveX = event.getX();
```

```
            moveY = event.getY();
            showDirection = true;
            invalidate();
            break;
        case MotionEvent.ACTION_UP:
            showCircle = false;
            showDirection = false;

            //调用warp方法根据触摸屏事件的坐标点来扭曲verts数组
            if(mBitmap != null && verts!= null && !mBitmap.isRecycled()) {
                warp(startX, startY, event.getX(), event.getY());
            }

            if (onStepChangeListener != null) {
                onStepChangeListener.onStepChange(false);
            }
            break;
    }
    return true;
}
```

這裡重點，看我們的wrap方法，來操作網格的變形.先簡述一下思路，我們剛才看到眼睛的放大，就是中心部分，操作幅度大，離的遠的地方基本不操作.

來看一下代碼

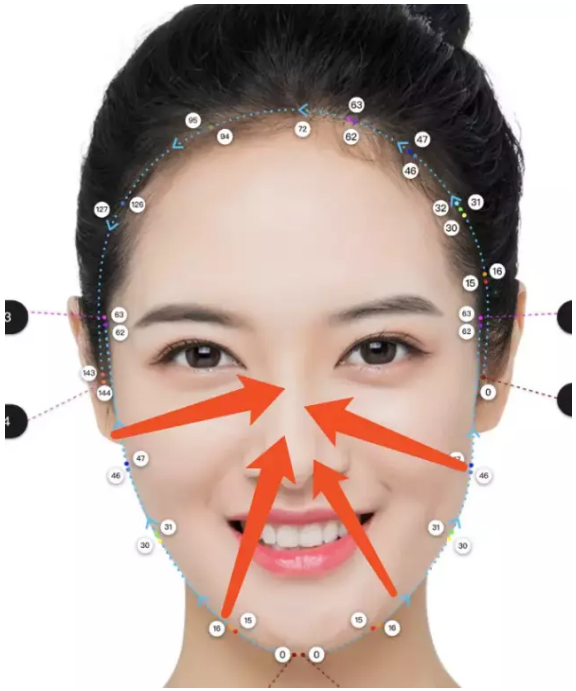
```
private void warp(float startX, float startY, float endX, float endY) {
    startX = toX(startX);
    startY = toY(startY);
    endX = toX(endX);
    endY = toY(endY);

    //计算拖动距离
    float ddPull = (endX - startX) * (endX - startX) + (endY - startY) * (endY - startY);
    float dPull = (float) Math.sqrt(ddPull);
    //dPull = screenWidth - dPull >= 0.0001f ? screenWidth - dPull : 0.0001f;
    if (dPull < 2 * r) {
        if (isSmllBody) {
            dPull = 1.8f * r;
        } else {
            dPull = 2.5f * r;
        }
```

```
    }

    int powR = r * r;
    int index = 0;
    int offset = 1;
    for (int i = 0; i < HEIGHT + 1; i++) {
      for (int j = 0; j < WIDTH + 1; j++) {
        //边界区域不处理
        if(i < offset || i > HEIGHT - offset || j < offset || j > WIDTH - offset){
          index = index + 1;
          continue;
        }
        //计算每个坐标点与触摸点之间的距离
        float dx = verts[index * 2] - startX;
        float dy = verts[index * 2 + 1] - startY;
        float dd = dx * dx + dy * dy;

        if (dd < powR) {
          //变形系数，扭曲度
          double e = (powR - dd) * (powR - dd) / ((powR - dd + dPull * dPull) * (powR - dd + dPull * dPull));
          double pullX = e * (endX - startX);
          double pullY = e * (endY - startY);
          verts[index * 2] = (float) (verts[index * 2] + pullX);
          verts[index * 2 + 1] = (float) (verts[index * 2 + 1] + pullY);

          // check
          if(verts[index * 2] < 0){
            verts[index * 2] = 0;
          }
          if(verts[index * 2] > mBitmap.getWidth()){
            verts[index * 2] =  mBitmap.getWidth();
          }

          if(verts[index * 2 + 1] < 0){
            verts[index * 2 +1] = 0;
          }
          if(verts[index * 2 + 1] > mBitmap.getHeight()){
            verts[index * 2 + 1] = mBitmap.getHeight();
          }
        }
        index = index + 1;
      }
    }
    invalidate();
  }
```

只要在操作半徑內，對X和Y進行不同的變形即可.

自動瘦臉實現

其實有了上面的拖動，要實現自動瘦臉就容易得多，我們對幾個關鍵點進行模擬拖動即可。



實現代碼如下

```
/**
 * 瘦脸算法
 *
 * @param bitmap     原来的bitmap
 * @return 之后的图片
 */
public static Bitmap smallFaceMesh(Bitmap bitmap, List<Point> leftFacePoint,List<Point> rightFacePoint,Point centerPoint, int level) {
    //交点坐标的个数
```

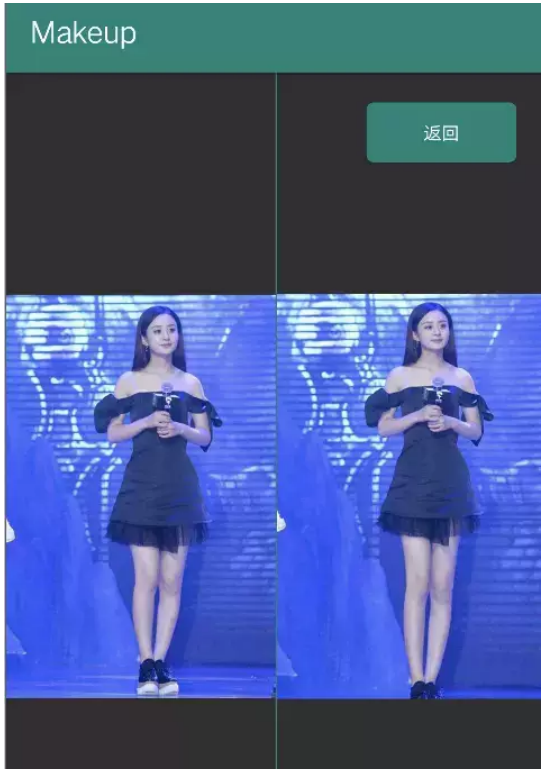
```
    int COUNT = (WIDTH + 1) * (HEIGHT + 1);
    //用于保存COUNT的坐标
    float[] verts = new float[COUNT * 2];
    float bmWidth = bitmap.getWidth();
    float bmHeight = bitmap.getHeight();

    int index = 0;
    for (int i = 0; i < HEIGHT + 1; i++) {
      float fy = bmHeight * i / HEIGHT;
      for (int j = 0; j < WIDTH + 1; j++) {
        float fx = bmWidth * j / WIDTH;
        //X轴坐标 放在偶数位
        verts[index * 2] = fx;
        //Y轴坐标 放在奇数位
        verts[index * 2 + 1] = fy;
        index += 1;
      }
    }
    int r = 180 + 15 * level;
    warp(COUNT,verts,leftFacePoint.get(16).x,leftFacePoint.get(16).y,centerPoint.x,centerPoint.y,r);
    warp(COUNT,verts,leftFacePoint.get(46).x,leftFacePoint.get(46).y,centerPoint.x,centerPoint.y,r);

    warp(COUNT,verts,rightFacePoint.get(16).x,rightFacePoint.get(16).y,centerPoint.x,centerPoint.y,r);
    warp(COUNT,verts,rightFacePoint.get(46).x,rightFacePoint.get(46).y,centerPoint.x,centerPoint.y,r);

    Bitmap resultBitmap = Bitmap.createBitmap(bitmap.getWidth(),bitmap.getHeight(), Bitmap.Config.ARGB_8888);
    Canvas canvas = new Canvas(resultBitmap);
    Paint paint = new Paint();
    canvas.drawBitmapMesh(bitmap,WIDTH, HEIGHT,verts,0,null,0,null);
    return resultBitmap;
}
```

大長腿

看代碼有些累吧，下面來看一個明星美女，有人知道這是誰嗎？問了兩三個程序員朋友，要么不知道，要么說這是楊冪嗎？哎，感嘆程序員認識的明星就那麼多嗎？

效果


Makeup
对比
此区域为选中区域



實現

上面的瘦臉操作需要對x和y兩個地方進行操作，那大長腿就繪變得容易一些，僅僅操作Y方向即可.

第一張圖，上面的覆蓋層為一個自定義View,下層直接使用了瘦臉功能的那個View，把圖片放在中心,只是不允許手勢操作圖片.

```
smallFaceView.setEnableOperate(false);
```

上層View核心代碼

```
//AdjustLegView 绘制部分
```

```
@Override
protected void onDraw(Canvas canvas) {
  super.onDraw(canvas);
  //line
  canvas.drawRect(0, topLine, getWidth(), topLine + LINEHIGHT, paint);
  //line
  canvas.drawRect(0, bottomLine, getWidth(), bottomLine + LINEHIGHT, paint);

  if (selectPos != -1) {
    swap();
    rect.set(0, topLine + LINEHIGHT, getWidth(), bottomLine);
    canvas.drawRect(rect, bgPaint);
    if(tipStr != null){
      @SuppressLint("DrawAllocation") Rect textRect = new Rect();
      textPaint.getTextBounds(tipStr,0,tipStr.length()-1,textRect);
      canvas.drawText(tipStr,rect.left + (rect.width()/ 2 -textRect.width()/2),
           rect.top + (rect.height()/ 2 -textRect.height()/2),textPaint);
    }
  }
}
```

手勢交互部分

```
//AdjustLegView
@Override
public boolean onTouchEvent(MotionEvent event) {
  float y = event.getY();
  switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN:
      selectPos = checkSelect(y);
      lastY = y;
      if(selectPos != -1 && listener != null){
        listener.down();
      }
      break;
    case MotionEvent.ACTION_MOVE:
      if (selectPos == 1) {
        // 最小 20 的偏移量
        topLine += checkLimit(y - lastY);
        invalidate();
      }
      if (selectPos == 2) {
        bottomLine += checkLimit(y - lastY);
```

```
            invalidate();
          }
          lastY = y;
          break;
        case MotionEvent.ACTION_UP:
        case MotionEvent.ACTION_CANCEL:
          selectPos = -1;
          invalidate();
          if( listener != null){
            listener.up(rect);
          }
          break;
      }
      return true;
    }

    private float checkLimit(float offset) {
      if (selectPos == 1) {
        if(topLine + offset > minLine && topLine + offset < maxLine){
          return offset;
        }
      }
      if (selectPos == 2) {
        if(bottomLine + offset > minLine && bottomLine + offset < maxLine){
          return offset;
        }
      }
      return 0;
    }

    private int checkSelect(float y) {
      selectPos = -1;
      RectF rect = new RectF(0, y - OFFSETY, 0, y + OFFSETY);
      float min = -1;
      if (topLine >= rect.top && topLine <= rect.bottom) {
        selectPos = 1;
        min = rect.bottom - topLine;
      }

      if (bottomLine >= rect.top && bottomLine <= rect.bottom) {
        if (min > bottomLine - rect.top || min == -1) {
          selectPos = 2;
        }
      }
      return selectPos;
    }
```

大長腿

那麼怎麼把腿部拉長呢？直接看一下算法部分

```
private static void warpLeg(int COUNT, float verts[], float centerY,int totalHeight,float region,float strength) {
   float  r = region / 2; //缩放区域力度

   for (int i = 0; i < COUNT * 2; i += 2) {
      //计算每个坐标点与触摸点之间的距离
      float dy = verts[i + 1] - centerY;
      double e = (totalHeight - Math.abs(dy)) / totalHeight;
      if(Math.abs(dy) < r){
         //拉长比率
         double pullY = e * dy * strength;
         verts[i + 1] = (float) (verts[i + 1] + pullY);
      }else if(Math.abs(dy) < 2 * r || dy > 0){
         double pullY = e * e * dy * strength;
         verts[i + 1] = (float) (verts[i + 1] + pullY);
      }else if(Math.abs(dy) < 3 * r){
         double pullY = e * e * dy * strength /2;
         verts[i + 1] = (float) (verts[i + 1] + pullY);
      }else {
         double pullY = e * e * dy * strength /4;
         verts[i + 1] = (float) (verts[i + 1] + pullY);
      }
   }
}

Canvas canvas = new Canvas(resultBitmap);
canvas.drawBitmapMesh(bitmap, WIDTH, HEIGHT, verts, 0, null, 0, null);
   return resultBitmap;
```

依然使用的是drawBitmapMesh，算法部分，只對Y進行了操作，X部分不操作，並且距離越遠，操作幅度越小. 盡量只拉長腿部，其他部分保持原有不動.

總結

本篇主要是介紹了，在Android上，使用原生API，怎麼去實現一些酷炫的效果. 文中的所有代碼都託管在github上，如果有需要，歡迎star，Github Makeup ，非常感謝，後續更新都會在此庫中進行.

https://github.com/DingProg/Makeup

●編號578 ，輸入編號直達本文

●輸入m獲取到文章目錄

閱讀原文