# 鸿蒙开源第三方组件——uCrop\_ohos图片裁剪组件

## 前言

基于安卓平台的图片裁剪组件uCrop（  
https://github.com/Yalantis/uCrop），实现了鸿蒙化迁移和重构。目前代码已经开源到（https://gitee.com/isrc\_ohos/u-crop\_ohos），欢迎各位下载使用并提出宝贵意见！

## 背景

uCrop组件是开源的图片裁剪库，支持对图片的缩放和裁剪等操作，是安卓平台比较受欢迎的组件，在Github上已有1万多个Star和近2千个Fork。uCrop组件具有封装程度高、使用流畅、自定义程度高的优点，被广泛应用于多种APP中。

## 组件效果展示

安卓和鸿蒙UI组件的差异较大，uCrop\_ohos的实现完全重构了安卓版uCrop的UI部分，所以uCrop\_ohos的组件效果看上去会和uCrop完全不同。

本组件的效果展示可分为两个步骤：图片选择和图片裁剪。下面依次对其进行讲解和展示。

### uCrop\_ohos图片选择

uCrop\_ohos支持裁剪系统选择相册图片或网络图片，用户可以在主菜单中选择对应的功能，如图1所示。



图 1主菜单界面

1. uCrop\_ohos读取相册图片

当用户赋予组件相应权限后，uCrop\_ohos可以自动读取手机相册中每一张图片，并将它们的缩略图作为一个列表呈现在UI界面上，用户可以上下滑动列表寻找目标图片，如图2所示。当用户点击某张缩略图时，会跳转到uCrop\_ohos的裁剪界面，执行后续操作。

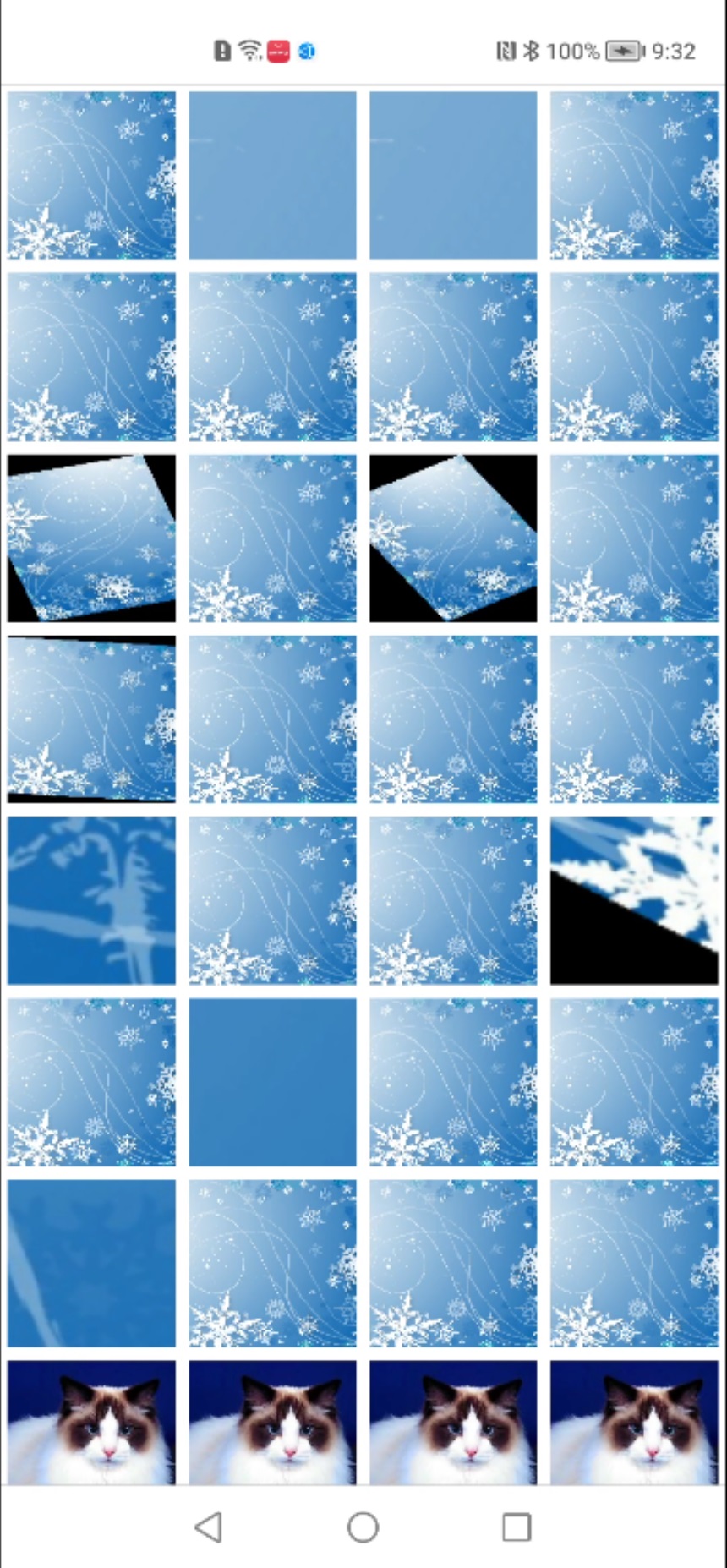


图 2 选择系统相册图片

1. uCrop\_ohos读取网络图片

用户需要将图片网址键入到输入框内并点击确定按钮，如图3所示。uCrop\_ohos会自动下载图片并跳转到裁剪界面，执行后续操作。

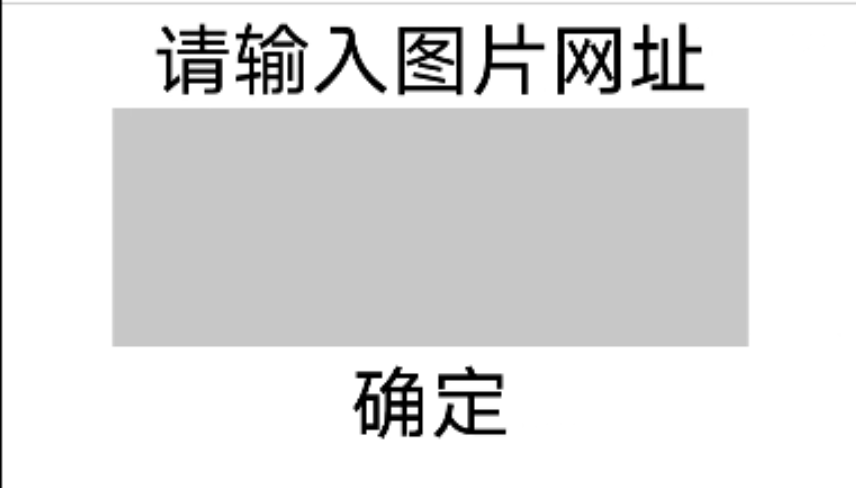


图 3 选择网络图片

### uCrop\_ohos图片裁剪

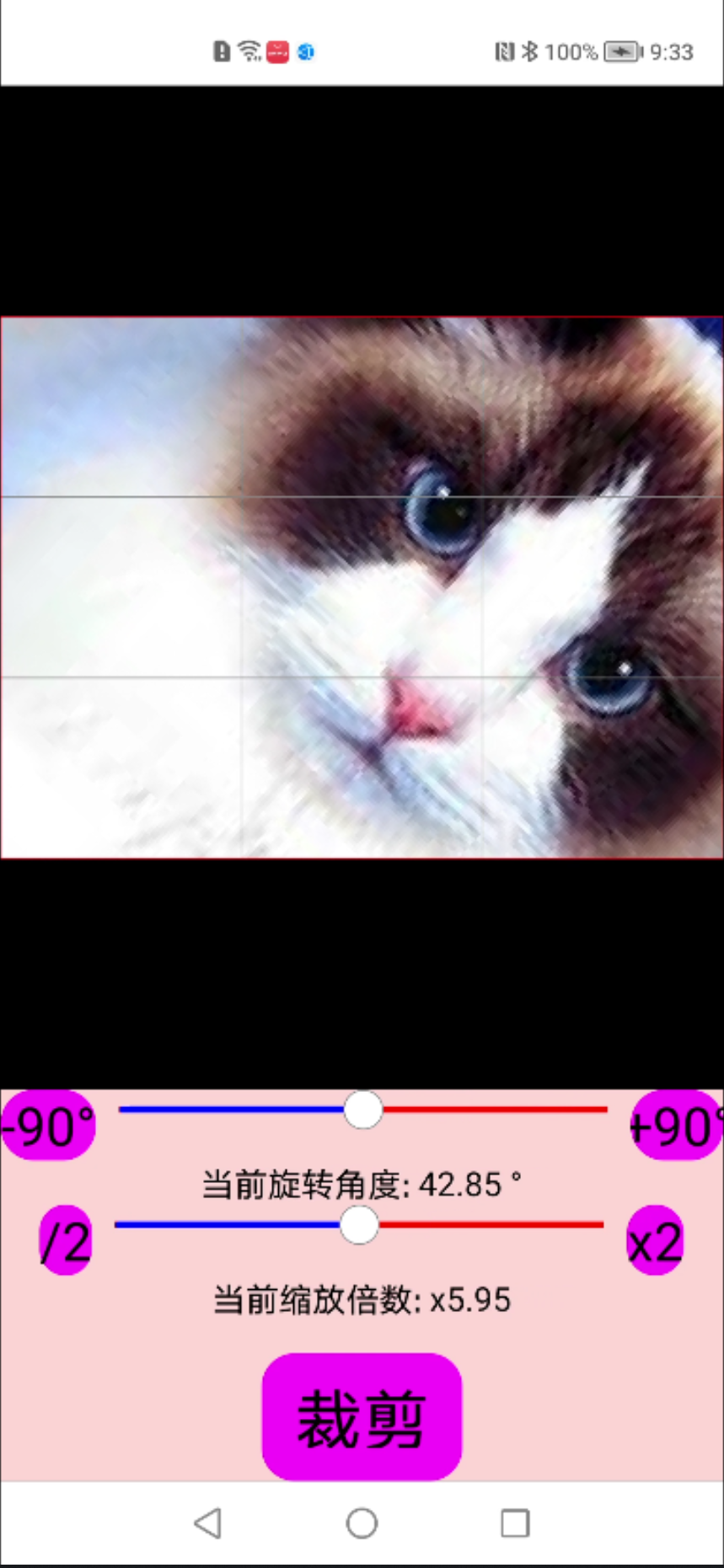


图4 uCrop\_ohos的裁剪界面

图4是uCrop\_ohos的裁剪界面。使用者可以通过手势对图片进行缩放、旋转和平移的操作，也可以通过按钮、滑块等控件进行相应操作。将图片调整至满意状态时，点击裁剪按钮即可获得裁剪后的新图片，并将其保存至手机相册。且本组件的图片与裁剪框具有自适应能力，能够保证裁剪框时刻在图片范围内，防止由于裁剪框的范围大于图片导致的一系列问题。

## Sample解析

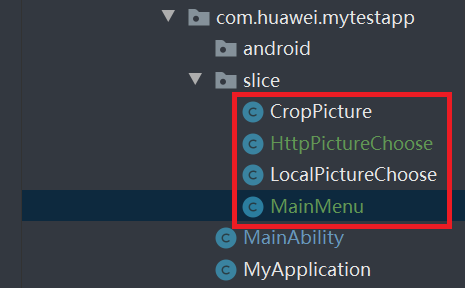


图5 Sample的工程结构

uCrop\_ohos的核心能力都由其Library提供，Sample主要用于构建UI，并调用Library的接口。从图5可以看出Sample的工程结构较为简单，主要由4个文件构成，下面进行详细的介绍。

### CropPicture

CropPicture文件提供了裁剪界面，其最主要的逻辑是通过图片Uri实例化Library中UCropView类。由于uCrop\_ohos的逻辑是先将用户选择的原图创建一个副本，然后对副本执行裁剪，所以为了将图片传入UCropView需要两个Uri：一个名为uri\_i，从intent中获得，标识的是用户选择的原图，可以是本地图片也可以是网络图片；另一个名为uri\_o，标识的是原图副本，一定是一张本地图片。代码如下：

//URI\_IN  
Uri uri\_i = intent.getUri();

//URI\_OUT  
String filename = "test.jpg";  
PixelMap.InitializationOptions options = new PixelMap.InitializationOptions();  
options.size = new Size(100,100);  
PixelMap pixelmap = PixelMap.create(options);  
Uri uri\_o = saveImage(filename, pixelmap);

//UcropView  
UCropView uCropView = new UCropView(this);  
try {  
 uCropView.getCropImageView().setImageUri(uri\_i, uri\_o);  
 uCropView.getOverlayView().setShowCropFrame(true);  
 uCropView.getOverlayView().setShowCropGrid(true);  
 uCropView.getOverlayView().setDimmedColor(Color.*TRANSPARENT*.getValue());  
  
} catch (Exception e) {  
 e.printStackTrace();  
}

Library给开发者提供了public接口，使得开发者易于封装自己的UI功能。例如本文件中的旋转和缩放滑块、旋转和缩放按钮、当前旋转和缩放状态的显示都是调用Library接口实现的。以如下功能的实现为例：创建了一个按钮，当用户触碰这个按钮之后就可以将图片右旋90度。其核心能力就是依靠调用Library中postRotate()函数实现的，非常简单。

//右旋90度的Button  
Button button\_plus\_90 = new Button(this);  
button\_plus\_90.setText("+90°");  
button\_plus\_90.setTextSize(80);  
button\_plus\_90.setBackground(buttonBackground);  
button\_plus\_90.setClickedListener(new Component.ClickedListener() {  
 @Override  
 public void onClick(Component component) {  
 float degrees = 90f;  
 //计算旋转中心  
 float center\_X = uCropView.getOverlayView().getCropViewRect().getCenter().getPointX();  
 float center\_Y = uCropView.getOverlayView().getCropViewRect().getCenter().getPointY();  
 //旋转  
 uCropView.getCropImageView().postRotate(degrees,center\_X,center\_Y);  
 //适配  
 uCropView.getCropImageView().setImageToWrapCropBounds(false);  
 //显示旋转角度  
 mDegree = uCropView.getCropImageView().getCurrentAngle();  
 text.setText("当前旋转角度: " + df.format(mDegree) + " °");  
 }  
});

### LocalPictureChoose & HttpPictureChoose

由上文可知，uri\_i是通过intent得到的，这个intent就是由 LocalPictureChoose或HttpPictureChoose传递的。LocalPictureChoose提供选择相册图片的能力，HttpPictureChoose提供选择网络图片的能力。

LocalPictureChoose提供的功能是将相册中的全部图片读取出来，做成缩略图排列在UI上，然后将每个缩略图绑定一个触摸监听器，一旦使用者选中某个缩略图，就会将这个缩略图对应的原图uri放在intent中传给CropPicture。具体代码如下：

private void showImage() {  
 DataAbilityHelper helper = DataAbilityHelper.*creator*(this);  
 try {  
 // columns为null，查询记录所有字段，当前例子表示查询id字段  
 ResultSet resultSet = helper.query(AVStorage.Images.Media.*EXTERNAL\_DATA\_ABILITY\_URI*, new String[]{AVStorage.Images.Media.*ID*}, null);  
 while (resultSet != null && resultSet.goToNextRow()) {

//创建image用以显示系统相册缩略图  
 PixelMap pixelMap = null;  
 ImageSource imageSource = null;  
 Image image = new Image(this);  
 image.setWidth(250);  
 image.setHeight(250);  
 image.setMarginsLeftAndRight(10, 10);  
 image.setMarginsTopAndBottom(10, 10);  
 image.setScaleMode(Image.ScaleMode.CLIP\_CENTER);  
 // 获取id字段的值  
 int id = resultSet.getInt(resultSet.getColumnIndexForName(AVStorage.Images.Media.*ID*));  
 Uri uri = Uri.appendEncodedPathToUri(AVStorage.Images.Media.*EXTERNAL\_DATA\_ABILITY\_URI*, String.valueOf(id));  
 FileDescriptor fd = helper.openFile(uri, "r");  
 ImageSource.DecodingOptions decodingOptions = new ImageSource.DecodingOptions();  
 try {

//解码并将图片放到image中  
 imageSource = ImageSource.create(fd, null);  
 pixelMap = imageSource.createPixelmap(null);  
 int height = pixelMap.getImageInfo().size.height;  
 int width = pixelMap.getImageInfo().size.width;  
 float sampleFactor = Math.max(height /250f, width/250f);  
 decodingOptions.desiredSize = new Size((int) (width/sampleFactor), (int)(height/sampleFactor));  
 pixelMap = imageSource.createPixelmap(decodingOptions);  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 if (imageSource != null) {  
 imageSource.release();  
 }  
 }  
 image.setPixelMap(pixelMap);  
 image.setClickedListener(new Component.ClickedListener() {  
 @Override  
 public void onClick(Component component) {  
 gotoCrop(uri);  
 }  
 });  
 tableLayout.addComponent(image);  
 }  
 } catch (DataAbilityRemoteException | FileNotFoundException e) {  
 e.printStackTrace();  
 }  
}

//uri放在intent中

private void gotoCrop(Uri uri){  
 Intent intent = new Intent();  
 intent.setUri(uri);  
 present(new CropPicture(),intent);  
}

HttpPictureChoose的功能主要是将用户输入的网络图片地址解析为Uri传递给CropPicture，目前只支持手动输入地址。

### MainMenu

一个简单的主菜单界面，用户可以通过点击不同的按钮选择裁剪相册图片还是网络图片。

## Library解析

鸿蒙和安卓存在较多的能力差异，即二者在实现同一 种功能时，方法不同，这不仅体现在工程结构上，也体现在具体的代码逻辑中。以下将对uCrop\_ohos和uCrop的工程结构进行对比，并介绍几个在uCrop\_ohos移植过程中遇到的安卓和鸿蒙的能力差异。

### 工程结构对比

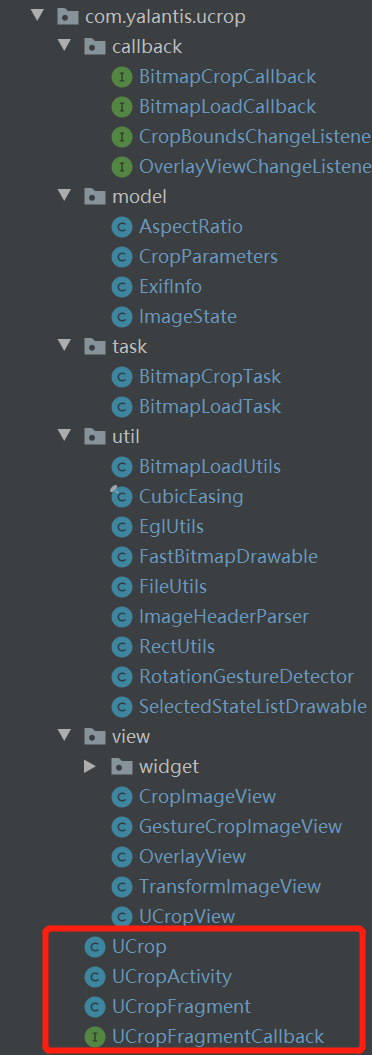
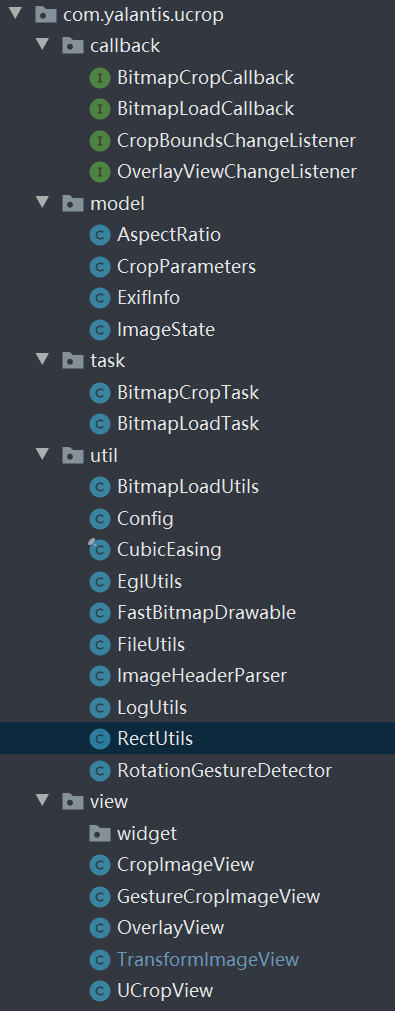


图 6 uCrop\_ohos(左)与uCrop(右)的工程结构对比

可以看出uCrop\_ohos相比uCrop少封装了一层Activity与Fragment，原因有3个：

（1）安卓的Activity与鸿蒙的Ability还是有差别的，强行复现会导致代码复用率低。

（2）这一层与UI强耦合，由于鸿蒙尚不支持安卓中许多控件，例如Menu等，这就导致难以原样复现UCropActivity中的UI。

（3）封装程度越高，可供开发者自定义的程度就越小。

### 能力差异

#### （1）图片加载&保存

不论是加载网络图片还是相册图片，在uCrop和uCrop\_ohos内部都是通过解析图片的Uri实现的，所以需要有一个识别Uri种类的过程，即通过分析Uri的Scheme来实现Uri的分类。如果Uri的Scheme是http或https则会被认为是网络图片，调用okhttp3的能力执行下载操作；如果Uri的Scheme是content（安卓）或dataability（鸿蒙）就会被认为是本地图片，执行复制操作。下载或复制的图片将作为被裁剪的图片。代码如下所示：

private void processInputUri() throws NullPointerException, IOException {  
 String inputUriScheme = mInputUri.getScheme();

//Scheme为http或https即为网络图片，执行下载  
 if ("http".equals(inputUriScheme) || "https".equals(inputUriScheme)) {  
 try {  
 downloadFile(mInputUri, mOutputUri);  
 } catch (NullPointerException e) {  
 LogUtils.*LogError*(*TAG*, "Downloading failed:"+e);  
 throw e;  
 }

//安卓中Scheme为content即为本地图片，执行复制  
 } else if ("content".equals(inputUriScheme)) {  
 try {  
 copyFile(mInputUri, mOutputUri);  
 } catch (NullPointerException | IOException e) {  
 LogUtils.*LogError*(*TAG*, "Copying failed:"+e);  
 throw e;  
 }

//鸿蒙中Scheme为dataability即为本地图片，执行复制  
 } else if("dataability".equals(inputUriScheme)){  
 try {  
 copyFile(mInputUri, mOutputUri);  
 } catch (NullPointerException | IOException e) {  
 LogUtils.*LogError*(*TAG*, "Copying failed:"+e);  
 throw e;  
 }

图片文件准备完成后，还需要将其解码成Bitmap（安卓）或PixelMap（鸿蒙）格式以便实现uCrop后续的各种功能。在解码之前还需要通过Uri来获取文件流，在这一点上安卓和鸿蒙的实现原理不同。对于安卓，可以通过openInputStream()函数获得输入文件流InputStream：

InputStream stream = mContext.getContentResolver().openInputStream(mInputUri);

对于鸿蒙则需要调用DataAbility，通过DataAbilityHelper先拿到FileDescriptor，然后才能得到InputStream：

InputStream stream = null;  
DataAbilityHelper helper = DataAbilityHelper.creator(mContext);  
FileDescriptor fd = helper.openFile(mInputUri, "r");  
stream = new FileInputStream(fd);

同样地，对于图片保存需要的输出文件流OutputStream，安卓和鸿蒙获取方式也存在不同，具体代码如下。

//安卓获取OutputStream

outputStream = context.getContentResolver().openOutputStream(Uri.fromFile(new File(mImageOutputPath)));

//鸿蒙获取OutputStream  
valuesBucket.putInteger("is\_pending", 1);  
DataAbilityHelper helper = DataAbilityHelper.creator(mContext.get());  
int id =helper.insert(AVStorage.Images.Media.*EXTERNAL\_DATA\_ABILITY\_URI*, valuesBucket);  
Uri uri = Uri.appendEncodedPathToUri(AVStorage.Images.Media.*EXTERNAL\_DATA\_ABILITY\_URI*, String.valueOf(id));  
//这里需要"w"写权限  
FileDescriptor fd = helper.openFile(uri, "w");  
OutputStream outputStream = new FileOutputStream(fd);

#### （2）裁剪的实现

在安卓版的uCrop中，裁剪功能的实现原理是将原图（位图1）位于裁剪框内的部分创建一个新的位图（位图2），然后将新的位图保存成图片文件（图片文件1）。如图7所示：

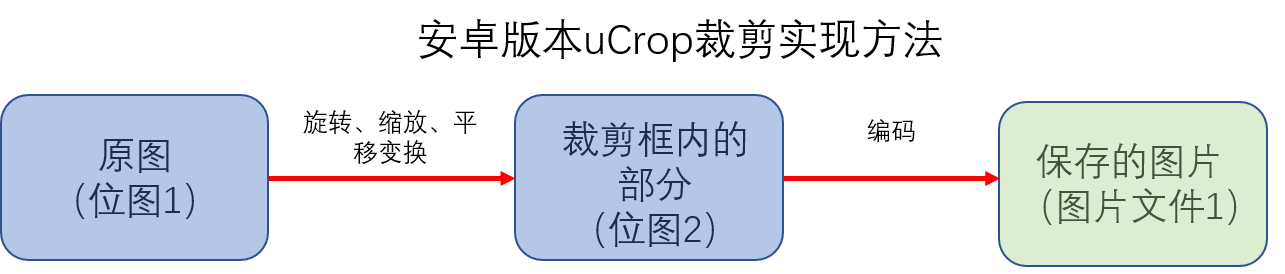


图 7 uCrop裁剪功能的实现方法

而在鸿蒙版uCrop\_ohos中，裁剪功能的实现原理发生了变化。鸿蒙系统API虽不支持对位图的旋转操作，但图像的解码API提供了旋转能力，所以鸿蒙的裁剪过程是这样的：

首先将原图（位图1）保存为一个临时的图片文件（图片文件1），通过相对旋转角度对临时图片文件进行读取，此时读取出的位图（位图2）就包含了正确的旋转信息。然后再通过相对缩放和位移创建一个新的位图（位图3），这个位图还会因为API的特性发生压缩和错切等形变，所以还需要再创建最后一个位图（位图4）来修正形变，最后再将位图4保存成图片文件（图片文件2）。如图8所示：

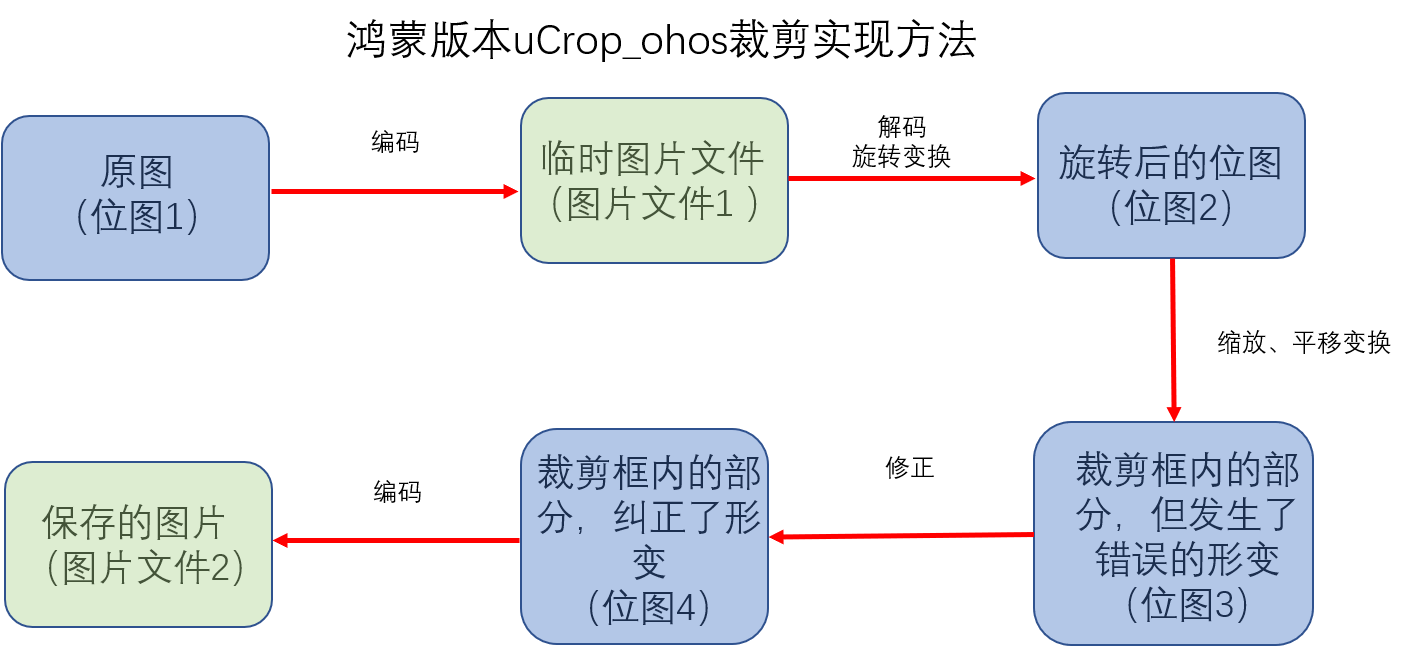


图 8 uCrop\_ohos裁剪功能的实现方法

#### （3）异步任务处理

由于图片的读取、裁剪和保存这些操作都是比较消耗系统性能的，直接导致的问题就是卡顿，所以需要使用异步任务将这些操作放到后台操作，减少UI线程的负担。下面以裁剪任务为例进行介绍。

在uCrop中使用的是BitmapCropTask类继承AsyncTask类的方法：

public class BitmapCropTask extends AsyncTask<Void, Void, Throwable>

然后在其中重写doInBackground()和onPostExecute()函数，分别实现后台裁剪任务的处理与回调：

@Override  
@Nullable  
protected Throwable doInBackground(Void... params) {  
 if (mViewBitmap == null) {  
 return new NullPointerException("ViewBitmap is null");  
 } else if (mViewBitmap.isRecycled()) {  
 return new NullPointerException("ViewBitmap is recycled");  
 } else if (mCurrentImageRect.isEmpty()) {  
 return new NullPointerException("CurrentImageRect is empty");  
 }  
  
 try {  
 crop();  
 mViewBitmap = null;  
 } catch (Throwable throwable) {  
 return throwable;  
 }  
  
 return null;  
}

@Override  
protected void onPostExecute(@Nullable Throwable t) {  
 if (mCropCallback != null) {  
 if (t == null) {  
 Uri uri = Uri.*fromFile*(new File(mImageOutputPath));  
 mCropCallback.onBitmapCropped(uri, cropOffsetX, cropOffsetY, mCroppedImageWidth, mCroppedImageHeight);  
 } else {  
 mCropCallback.onCropFailure(t);  
 }  
 }  
}

鸿蒙中没有搭载类似安卓的AsyncTask类，所以uCrop\_ohos修改了后台任务的处理方案，首先将后台任务的处理与回调合并写在一个Runnable中，然后鸿蒙原生的多线程处理机制EventHandler搭配EventRunner新开一个线程用于处理这个Runnable，实现了图片裁剪任务的异步处理。

public void doInBackground(){   
 EventRunner eventRunner = EventRunner.*create*();  
 EventHandler handler = new EventHandler(eventRunner);  
 handler.postTask(new Runnable() {  
 @Override  
 public void run() {  
 if (mViewBitmap == null) {  
 Throwable t = new NullPointerException("ViewBitmap is null");  
 mCropCallback.onCropFailure(t);  
 return;  
 } else if (mViewBitmap.isReleased()) {  
 Throwable t = new NullPointerException("ViewBitmap is null");  
 mCropCallback.onCropFailure(t);  
 return;  
 } else if (mCurrentImageRect.isEmpty()) {  
 Throwable t = new NullPointerException("ViewBitmap is null");  
 mCropCallback.onCropFailure(t);  
 return;  
 }  
 try {  
 crop();  
 mViewBitmap = null;  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
}

## 项目贡献人

吴圣垚 郑森文 朱伟 陈美汝 王佳思