Android 多媒体进阶

ByteDance 字节跳动

目录

- 视频介绍
- 相机拍照
- 简单录制

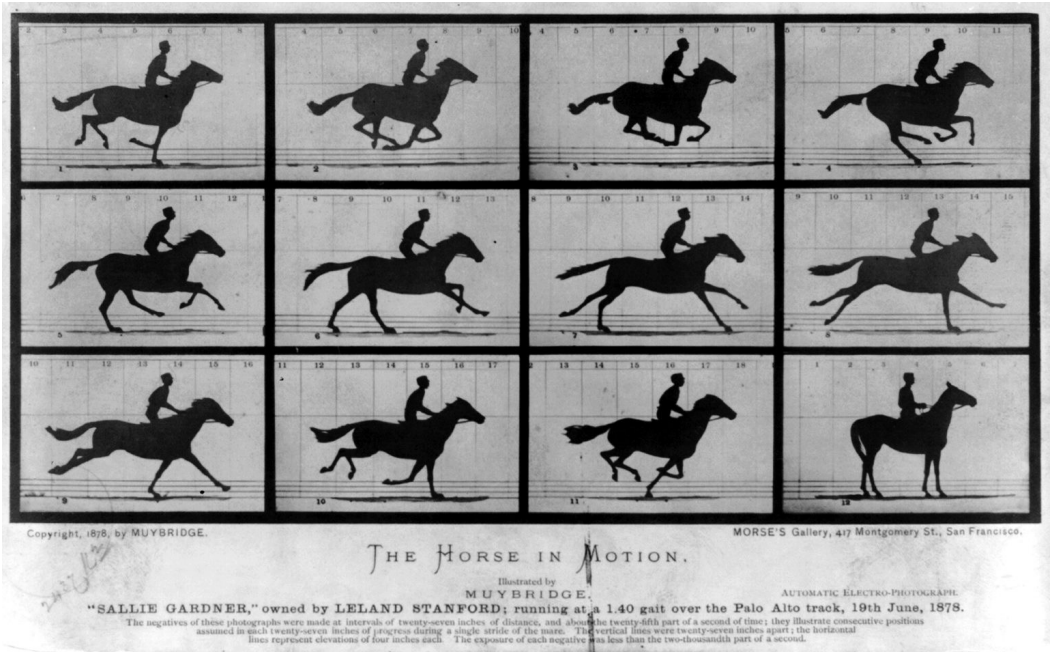
视频介绍

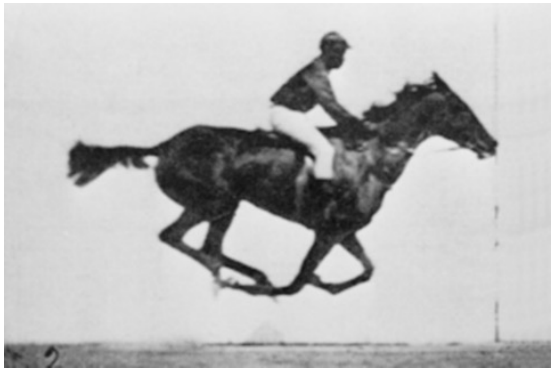
视频介绍

- 视频是什么？
- 帧率/分辨率/码率
- 为什么要进行视频编码？
- 视频封装格式

视频是什么？

- 连续的图像
- 视觉暂留

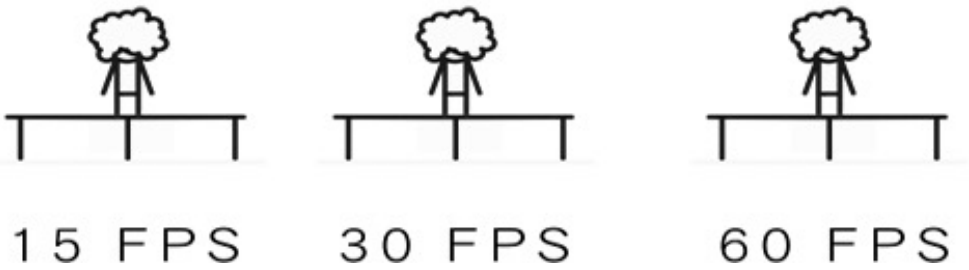




电影的先驱：迈布里奇的马，1878

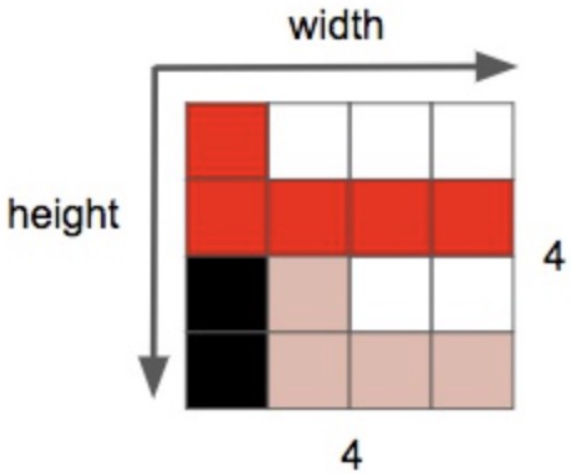
帧率

每秒钟播放的图片数量



分辨率

图像内像素的数量，通常使用宽*高表示



码率

视频文件在单位时间内使用的数据流量，单位是kbps即千位每秒

视频大小 = duration 时长(s) x kbps 千位每秒/ 8 = xx MB

为什么要视频编码？

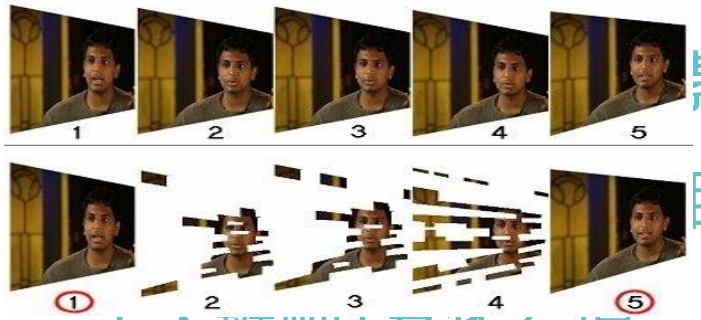
分辨率1920*1080，帧率是30，未压缩情况下码率是多少？

每一帧是1920*1080*24=49766400bit

每一秒是49766400*30/8 ~ =186.6MB

90分钟视频的大小大概是1000GB

视频编码



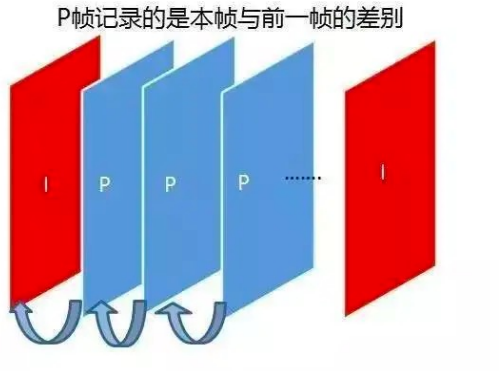
帧之间有较强的相关性

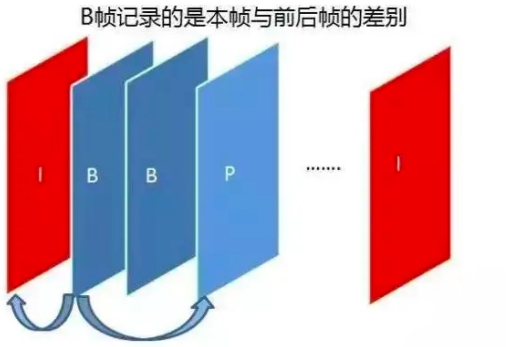
相邻图像之间内容相似，可以压缩成

1个关键帧+变化量值

I/B/P帧

I帧：关键帧
P帧：前向预测帧
B帧：双向预测帧





编码格式

目前主要使用的编码格式有H264和H265

H264可以达到百倍的压缩率

H265比H264的压缩率增加一倍

封装格式

把视频码流和音频码流按照一定的格式存储在一个文件中，与编码格式无关。

常用的格式有MP4，AVI，FLV，RMVB等

相机拍照

相机拍照

- 调起系统相机
- 接收数据
- 保存图片
- 显示图片

调起系统相机

- 打开系统相机

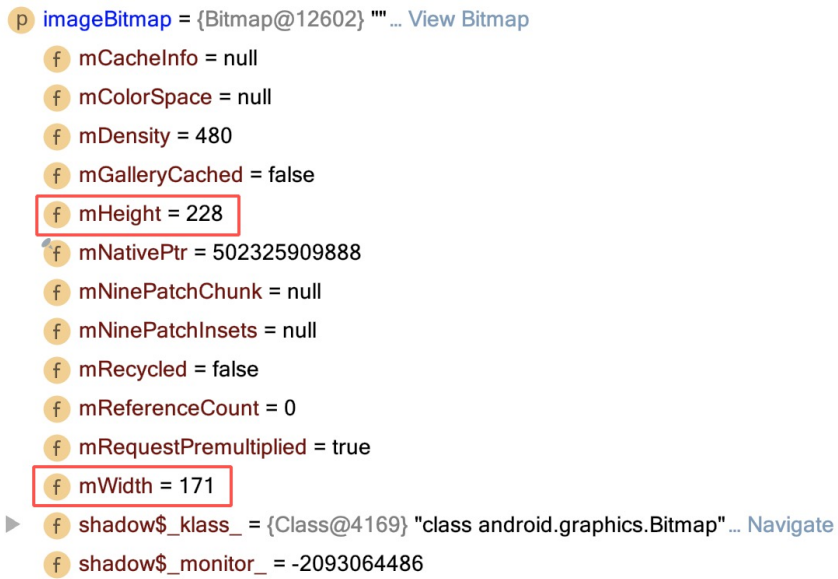
```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
```

- 接收数据，拿到返回的bitmap，并显示在屏幕上

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode ==  RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

接收数据

- 为啥图像这么小？



自定义存储路径（一）

❑ 申请存储权限

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

❑ 创建文件

```
private File createImageFile() throws IOException {
    // 获取当前时间作为文件名
    String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";

    // 获取应用文件存储路径 Android/data/com.bytedance.camera.demo/files/Pictures
    File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(imageFileName,  suffix: ".jpeg", storageDir);

    // 保存文件路径
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}
```

自定义存储路径（二）

- 获取content:// URI，7.0以上手机不允许使用file:// URI跳出应用

```
<provider
    android:authorities="com.bytedance.camera.demo.fileprovider"
    android:name="androidx.core.content.FileProvider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths" />
</provider>
```

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-files-path
        name="my_images"
        path="Pictures" />
</paths>
```

自定义存储路径（三）

- 设置存储地址

```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File photoFile = null;
    try {
        photoFile = createImageFile();
    } catch (IOException ex) {
        // error
    }
    if (photoFile != null) {
        // 获取存储图片的URI
        Uri photoURI = FileProvider.getUriForFile( context: this,
                authority: "com.bytedance.camera.demo.fileprovider", photoFile);
        takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

显示图片

- 获取view的宽高
- 获取图片的宽高
- 计算缩放比例
- 获取bitmap
- 显示在屏幕上

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode ==  RESULT_OK) {
        // View的宽高
        int targetW = mImageView.getWidth();
        int targetH = mImageView.getHeight();

        BitmapFactory.Options bmOptions = new BitmapFactory.Options();
        bmOptions.inJustDecodeBounds = true;

        // 图片的宽高
        int photoW = bmOptions.outWidth;
        int photoH = bmOptions.outHeight;

        int scaleFactor = Math.min(photoW/targetW, photoH/targetH);

        bmOptions.inJustDecodeBounds = false;
        bmOptions.inSampleSize = scaleFactor;
        bmOptions.inPurgeable = true;

        // 根据View的大小解码图片的大小
        Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
        mImageView.setImageBitmap(bitmap);
    }
}
```

显示效果

- 图片为什么旋转了
- 如何进行转正呢

图片为啥旋转了？

- 读取图片的旋转角度
- 在matrix中设置要旋转的角度
- 旋转图片

```
public static Bitmap rotateImage(Bitmap bitmap, String path) throws Exception {
    ExifInterface srcExif = new ExifInterface(path);
    Matrix matrix = new Matrix();
    int angle = 0;
    int orientation = srcExif.getAttributeInt(ExifInterface.TAG_ORIENTATION, ExifInterface.ORIENTATION_NORMAL);
    switch (orientation) {
        case ExifInterface.ORIENTATION_ROTATE_90:
            angle = NUM_90;
            break;
        case ExifInterface.ORIENTATION_ROTATE_180:
            angle = NUM_180;
            break;
        case ExifInterface.ORIENTATION_ROTATE_270:
            angle = NUM_270;
            break;
        default:
            break;
    }
    matrix.postRotate(angle);
    return Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix, true);
}
```

随堂练习

- 给自己来个自拍
  - 解决权限申请
  - 存储到sd卡
  - 图片预览方向正确
- 拓展-在相册中能扫描到自拍照片

简单录制

视频录制

- 调起相机录像
- 接收录制视频
- 显示视频
- 查看视频文件

调起系统相机

- 调起相机的录像页面

```
1 Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
2 if (takeVideoIntent.resolveActivity(getPackageManager()) != null) {
3     startActivityForResult(takeVideoIntent, REQUEST_VIDEO_CAPTURE);
4 }
```

显示录制视频

- 获取拍摄的视频，并显示在页面上，开始播放

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_VIDEO_CAPTURE && resultCode == RESULT_OK) {
        Uri videoURI =  data.getData();
        mVideoView.setVideoURI(videoURI);
        mVideoView.start();
    }
}
```

查看数据

- 视频的封装格式

  .mp4

- 视频的分辨率是多大？

  720 * 1280

- 视频的文件大小和录制时长

  7.15MB / 8秒

- 计算视频的码率

  7.15 * 1024 * 1024 * 8 / 8 = 7497.3 kbps

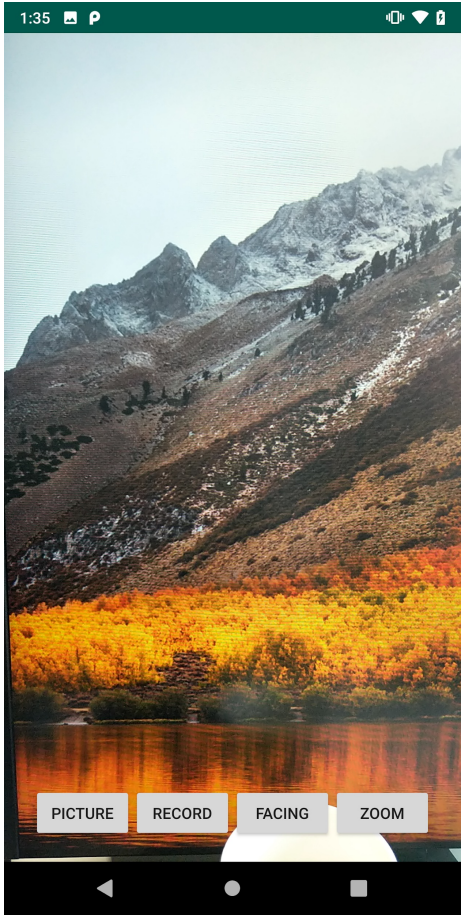
随堂练习

- 录制一段自拍视频
  - 解决权限申请
  - 默认存储
  - 相机拍摄后在页面上播放
  - 点击暂停，再次点击恢复播放

自定义录制

效果展示

- 调起相机
- 屏幕实时显示画面
- 拍个照片
- 录一段视频



获取Camera实例

- 申请权限

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

- 一共几个摄像头

  Camera.getNumberOfCameras

- 怎么获取后置摄像头

```
releaseCameraAndPreview();
Camera cam = Camera.open(Camera.CameraInfo.CAMERA_FACING_BACK);
rotationDegree = getCameraDisplayOrientation(position);
cam.setDisplayOrientation(rotationDegree);
```

摄像头数据实时显示

- 用什么控件？

  SurfaceView

- 几个关键类

  Camera

  SurfaceView

  SurfaceHolder

  SurfaceHolder.Callback

```
Camera mCamera = getCamera();
SurfaceView mSurfaceView = findViewById(R.id.img);
SurfaceHolder surfaceHolder = mSurfaceView.getHolder();
surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
surfaceHolder.addCallback(new SurfaceHolder.Callback() {
    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        mCamera.setPreviewDisplay(holder);
        mCamera.startPreview();
    }
    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {}
    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        mCamera.stopPreview();
        mCamera.release();
        mCamera = null;
    }
});
```

处理生命周期对预览的影响

```
1  @Override
2  protected void onResume() {
3      super.onResume();
4      if (mCamera == null) {
5          initCamera();
6      }
7      mCamera.startPreview();
8  }
9
10 @Override
11 protected void onPause() {
12     super.onPause();
13     mCamera.stopPreview();
14 }
```

拍摄一张实时照片

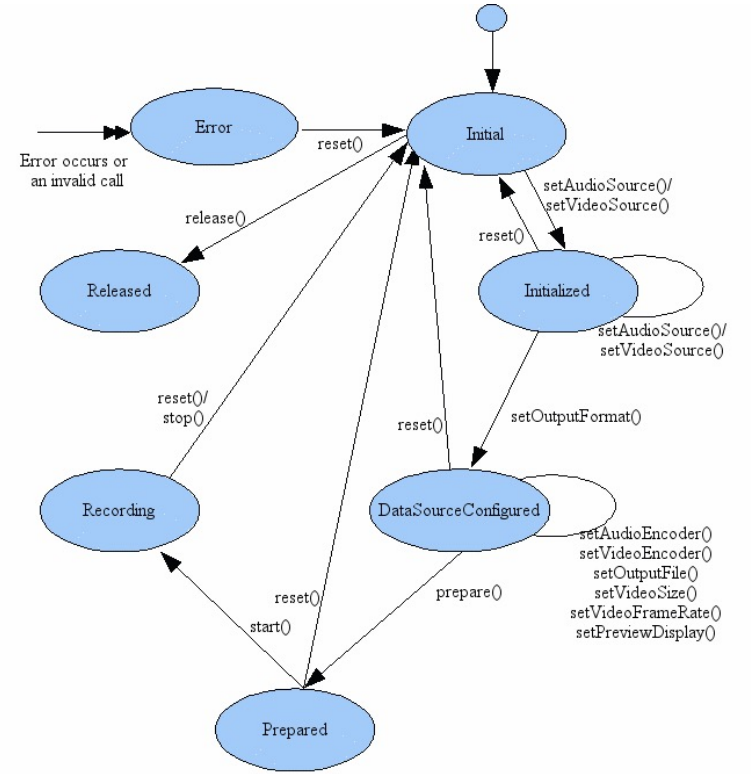
- 怎么用 camera api 拍照

  mCamera.takePicture(null, null, mPicture)

- 拍照后继续预览

```
1  private Camera.PictureCallback mPicture = (data, camera) -> {
2      File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
3      try {
4          FileOutputStream fos = new FileOutputStream(pictureFile);
5          fos.write(data);
6          fos.close();
7      } catch (IOException e) {
8          Log.d("mPicture", "Error accessing file: " + e.getMessage());
9      }
10     mCamera.startPreview();
11 };
```

认识MediaRecorder



MediaRecorder state diagram

开始录制（按部就班）

- Unlock the Camera
- Configure MediaRecorder
  - setCamera()
  - setAudioSource()
  - setVideoSource()
  - setProfile
  - setOutputFile()
  - setPreviewDisplay()
- Prepare MediaRecorder
- Start MediaRecorder

```
mMediaRecorder = new MediaRecorder();
// Step 1: Unlock and set camera to MediaRecorder
mCamera.unlock();
mMediaRecorder.setCamera(mCamera);
// Step 2: Set sources
mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);
mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
// Step 3: Set a CamcorderProfile (requires API Level 8 or higher)
mMediaRecorder.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH));
// Step 4: Set output file
mMediaRecorder.setOutputFile(getOutputMediaFile(MEDIA_TYPE_VIDEO).toString());
// Step 5: Set the preview output
mMediaRecorder.setPreviewDisplay(mSurfaceView.getHolder().getSurface());
mMediaRecorder.setOrientationHint(rotationDegree);
// Step 6: Prepare configured MediaRecorder
try {
    mMediaRecorder.prepare();
    mMediaRecorder.start();
}  catch (Exception e) {
    releaseMediaRecorder();
    return false;
}
```

结束录制（按部就班）

- Stop MediaRecorder
- Reset MediaRecorder
- Release MediaRecorder
- Lock the Camera

```
1 mMediaRecorder.stop();
2 mMediaRecorder.reset();
3 mMediaRecorder.release();
4 mMediaRecorder = null;
5 mCamera.lock();
```

随堂练习

- 录制一段视频
  - 解决权限申请
  - 存储到sd卡
  - 视频预览正确
  - 视频存储后。预览正确
- 拓展-在相册中能扫描到该视频

THANKS

ByteDance字节跳动