

 Penguin (<https://www.polarxiong.com/>)

不忘初心，方得始终

Android相机开发(六): 高效实时处理预览帧数据

文章列表 (<https://www.polarxiong.com/>)文章目录 (<https://www.polarxiong.com/list.html>)

2016-09-04 18:30

Android (<https://www.polarxiong.com/category/Android/>)动手实践 (<https://www.polarxiong.com/project.html>)相机 (<https://www.polarxiong.com/tag/%E7%9B%B8%E6%9C%BA/>), camera(<https://www.polarxiong.com/tag/camera/>), HandlerThread(<https://www.polarxiong.com/tag/HandlerThread/>), AsyncTask关于 (<https://www.polarxiong.com/about.html>)(<https://www.polarxiong.com/tag/AsyncTask/>), ThreadPoolExecutor(<https://www.polarxiong.com/tag/ThreadPoolExecutor/>)

36

Android Camera Develop: process preview frames in real time efficiently (<https://github.com/zhantong>)

概述

本篇我们暂时不介绍像相机APP增加新功能，而是介绍如何处理相机预览帧数据。想必大多数人都对处理预览帧没有需求，因为相机只需要拿来拍照和录像就好了，实际上本篇和一般的相机开发也没有太大联系，但因为仍然是在操作Camera类，所以还是归为相机开发。处理预览帧简单来说就是对相机预览时的每一帧的数据进行处理，一般来说如果相机的采样速率是30fps的话，一秒钟就会有30个帧数据需要处理。帧数据具体是什么？如果你就是奔着处理帧数据来的话，想必你早已知道答案，其实就是一个byte类型的数组，包含的是YUV格式的帧数据。本篇仅介绍几种高效地处理预览帧数据的方法，而不介绍具体的用处，因为拿来进行人脸识别、图像美化等又是长篇大论了。

本篇在Android相机开发(二): 给相机加上偏好设置 (</archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E4%BA%8C-%E7%BB%99%E7%9B%B8%E6%9C%BA%E5%8A%A0%E4%B8%8A%E5%81%8F%E5%A5%BD%E8%AE%BE%E7%BD%AE.html>)的基础上介绍。预览帧数据的处理通常会包含大量的计算，从而导致因为帧数据太多而处理效率低下，以及衍生出的预览画面卡顿等问题。本篇主要介绍分离线程优化

画面显示，以及通过利用HandlerThread、Queue、ThreadPool和Penguin (<https://www.polarxiong.com>) AsyncTask来提升帧数据处理效率的方法。

准备

文章简单起见，我们在相机开始预览的时候就开始获取预览帧并进行处理，为了能更清晰地分析这个过程，我们在UI中“设置”按钮之下增加“开始”和“停止”按钮以控制相机预览的开始与停止。

关于(<https://www.polarxiong.com/about.html>)

修改activity_main.xml，将

 (<https://github.com/zhantong>)

```
1
2     android:id="@+id/button_settings"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:text="设置" />
```

Java

替换为

Java

```

1  Penguin(https://www.polarxiong.com)
2  不忘初心，方得始终
3      android:layout_width="wrap_content"
4      android:layout_height="wrap_content"
5      android:layout_gravity="right"
6      android:orientation="Vertical">
7
8  文章列表(https://www.polarxiong.com)
9  文章目录(https://www.polarxiong.com/list.html)
10  动手实践(https://www.polarxiong.com/project.html)
11  关于我(https://www.polarxiong.com/about.html)
12
13      <Button
14          android:id="@+id/button_start_preview"
15          android:layout_width="wrap_content"
16          android:layout_height="wrap_content"
17          android:text="开始" />
18
19      <Button
20          android:id="@+id/button_stop_preview"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:text="停止" />
24  </LinearLayout>

```

这样增加了“开始”和“停止”两个按钮。

绑定事件

修改mainActivity，将原 onCreate() 中初始化相机预览的代码转移到新建的方法 startPreview() 中

Java

1 Penguin(<https://www.polarxiong.com>)

不忘初心，方得始终

```

1    final CameraPreview mPreview = new CameraPreview(this);
2
3    FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
4    preview.addView(mPreview);
5
6    SettingsFragment.passCamera(mPreview.getCameraInstance());
7
8    SettingsFragment.setDefaultValues(this, R.xml.preferences, false);
9    SettingsFragment.setDefault(PreferenceManager.getDefaultSharedPreferences(this));
10   SettingsFragment.init(PreferenceManager.getDefaultSharedPreferences(this));
11
12   Button buttonSettings = (Button) findViewById(R.id.button_settings);
13   buttonSettings.setOnClickListener(new View.OnClickListener() {
14       @Override
15       public void onClick(View v) {
16           getFragmentManager().beginTransaction().replace(R.id.camera_preview, new SettingsFragment(), null).commit();
17       }
18   })

```

文章列表 (<https://www.polarxiong.com>)

文章目录 (<https://www.polarxiong.com/post.html>)

动手实践 (<https://www.polarxiong.com/project.html>)

关于 (<https://www.polarxiong.com/about.html>)

<https://github.com/zhanlong>

同时再增加一个 `stopPreview()` 方法，用来停止相机预览

```

1    public void stopPreview() {
2        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
3        preview.removeAllViews();
4    }

```

Java

`stopPreview()` 获取相机预览所在的 `FrameLayout`，然后通过 `removeAllViews()` 将相机预览移除，此时会触发 `CameraPreview` 类中的相关结束方法，关闭相机预览。

现在 `onCreate()` 的工作就很简单了，只需要将两个按钮绑定上对应的方法就好了

Java

```

1 Penguin (https://www.polarxiong.com) {
2     super.onCreate(savedInstanceState);
3     setContentView(R.layout.activity_main);
4
5     文章列表 (https://www.polarxiong.com)
6     Button buttonStartPreview = (Button) findViewById(R.id.button_start
7     buttonStartPreview.setOnClickListener(new View.OnClickListener() {
8         @Override
9         public void onClick(View v) {
10             startPreview();
11         }
12     });
13
14     关于 (https://www.polarxiong.com/about.html)
15     Button buttonStopPreview = (Button) findViewById(R.id.button_stop_
16     buttonStopPreview.setOnClickListener(new View.OnClickListener() {
17         @Override
18         public void onClick(View v) {
19             stopPreview();
20         }
21     });
22 }

```

运行试试

现在运行APP不会立即开始相机预览了，点击“开始”按钮屏幕上才会出现相机预览画面，点击“停止”则画面消失，预览停止。

基本的帧数据获取和处理

这里我们首先实现最基础，也是最常用的帧数据获取和处理的方法；然后看看改进提升性能的方法。

基础

获取帧数据的接口是 `Camera.PreviewCallback`，实现此接口下的 `onPreviewFrame(byte[] data, Camera camera)` 方法即可获取到每个帧数据 `data`。所以现在要做的就是给 `CameraPreview` 类增加 `Camera.PreviewCallback` 接口声明，再在 `CameraPreview` 中实现 `onPreviewFrame()` 方法，最后给 `Camera` 绑定此接口。这样相机预览时每产

生一个预览帧，就会调用 `onPreviewFrame()` 方法，处理预览帧数据 `data`。

👉 Penguin (<https://www.polarxiong.com>)

忘初心，方得始终

在 `CameraPreview` 中，修改

文章列表 (<https://www.polarxiong.com>) Java

```
1 public class CameraPreview extends SurfaceView implements SurfaceHolder
```

文章目录 (<https://www.polarxiong.com/list.html>)

动手实践 (<https://www.polarxiong.com/project.html>) Java

```
1 public class CameraPreview extends SurfaceView implements SurfaceHolder
```

关于 <https://www.polarxiong.com/opencv.html> 接口声明。

加🔍 `onPreviewFrame()` 的实现

🔍 (<https://github.com/zhantong>) Java

```
1 public void onPreviewFrame(byte[] data, Camera camera) {
2     Log.i(TAG, "processing frame");
3     try {
4         Thread.sleep(500);
5     } catch (InterruptedException e) {
6         e.printStackTrace();
7     }
8 }
```

这里并没有处理帧数据 `data`，而是暂停0.5秒模拟处理帧数据。

在 `surfaceCreated()` 中 `getCameraInstance()` 这句的下面加入

1 `mCamera.setPreviewCallback(this);` Java

将此接口绑定到 `mCamera`，使得每当有预览帧生成，就会调用 `onPreviewFrame()`。

运行试试

现在运行APP，点击“开始”，一般在屏幕上观察不到明显区别，但这里其实有两个潜在的问题。其一，如果你这时点击“设置”，会发现设置界面并不是马上出现，而是会延迟几秒出现；而再点击返回键，设置界面也会过几秒才消失。其二，在logcat中可以看到输出的 "processing frame"，大约0.5秒输

出一条，因为线程睡眠设置的是0.5秒，所以一秒钟的30个帧数据只处理了2帧，剩下的28帧都被丢弃了（这里没有非常直观的方法显示剩下的28帧被丢弃了，但事实就是这样，不严格的来说，当新的帧数据到达时，如果 `onPreviewFrame()` 正在执行还没有返回，这个帧数据就会被丢弃）。
[Penguin \(https://www.polarxiong.com\)](https://www.polarxiong.com)
 忘初心，方得始终
[文章列表 \(https://www.polarxiong.com\)](https://www.polarxiong.com)

与UI线程分离

[文章目录 \(https://www.polarxiong.com/list.html\)](https://www.polarxiong.com/list.html)

问题分析

[动手实践 \(https://www.polarxiong.com/project.html\)](https://www.polarxiong.com/project.html)

现在我们来解决第一个问题。第一个问题的原因很简单，也是Android开发中经常碰到的：UI线程被占用，导致UI操作卡顿。在这里就是 `onPreviewFrame()` 会阻塞线程，而阻塞的线程就是UI线程。

`onPreviewFrame()` 在哪个线程执行？官方文档里有相关描述：

<https://github.com/zhantong>
 Called as preview frames are displayed. This callback is invoked on the event thread `open(int)` was called from.



意思就是 `onPreviewFrame()` 在执行 `Camera.open()` 时所在的线程运行。而目前 `Camera.open()` 就是在UI线程中执行的（因为没有创建新进程），对应的解决方法也很简单了：让 `Camera.open()` 在非UI线程执行。

解决方法

这里使用 `HandlerThread` 来实现。`HandlerThread` 会创建一个新的线程，并且有自己的 `loop`，这样通过 `Handler.post()` 就可以确保在这个新的线程执行指定的语句。虽然说起来容易，但还是有些细节问题要处理。

先从 `HandlerThread` 下手，在 `CameraPreview` 中加入

Java

 Penguin (<https://www.polarxiong.com>)
 不忘初心，方得始终
 文章列表 (<https://www.polarxiong.com>)
 文章目录 (<https://www.polarxiong.com/list.html>)
 动手实践 (<https://www.polarxiong.com/project.html>)
 关于 (<https://www.polarxiong.com/about.html>)
 <https://github.com/zhanlong>

```

1  CameraHandlerThread extends HandlerThread {
2      Handler mHandler;
3
4      public CameraHandlerThread(String name) {
5          super(name);
6          start();
7      }
8
9      @Override
10     synchronized void notifyCameraOpened() {
11         notify();
12     }
13
14     void openCamera() {
15         mHandler.post(new Runnable() {
16             @Override
17             public void run() {
18                 openCameraOriginal();
19                 notifyCameraOpened();
20             }
21         });
22         try {
23             wait();
24         } catch (InterruptedException e) {
25             Log.w(TAG, "wait was interrupted");
26         }
27     }
28 }

```

CameraHandlerThread 继承自HandlerThread，在构造函数中就 start() 启动这个Thread，并创建一个handler。openCamera() 要达到的效果是在此线程中执行 mCamera = Camera.open();，因此通过 handler.post() 在 Runnable() 中执行，我们将要执行的语句封装在 openCameraOriginal() 中。使用notify-wait是为安全起见，因为 post() 执行会立即返回，而 Runnable() 会异步执行，可能在执行 post() 后立即使用 mCamera 时仍为 null；因此在这里加上notify-wait控制，确认打开相机后，openCamera() 才返回。

接下来是 openCameraOriginal()，在CameraPreview中加入

Java

❗ Penguin(<https://www.polarxiong.com>)

不忘初心，方得始终 {
 3 mCamera = Camera.open();
 4 } catch (Exception e) {
 文章列表(<https://www.polarxiong.com>):
 5 Log.d(TAG, "camera is not available");
 6 }
 文章目录(<https://www.polarxiong.com/list.htm>)

这个不用解释，就是封装成了方法。
 动手实践(<https://www.polarxiong.com/project.htm>)

最后将 `getCameraInstance()` 修改为
 关于(<https://www.polarxiong.com/about.htm>)

```

1 public Camera getCameraInstance() {
2     if (mCamera == null) {
3         CameraHandlerThread mThread = new CameraHandlerThread("camera t
4     synchronized (mThread) {
5         mThread.openCamera();
6     }
7 }
8 return mCamera;
9 }
```

Java

这个也很容易理解，就是交给 `CameraHandlerThread` 来处理。

运行试试

现在运行APP，会发现第一个问题已经解决了。

处理帧数据

接下来解决第二个问题，如何确保不会有帧数据被丢弃，即保证每个帧数据都被处理。解决方法的中心思想很明确：让 `onPreviewFrame()` 尽可能快地返回，不至于丢弃帧数据。

下面介绍4种比较常用的处理方法：`HandlerThread`、`Queue`、`AsyncTask`和`ThreadPool`，针对每一种方法简单分析其优缺点。

HandlerThread

不忘初心，方得始终

简介

文章列表 (<https://www.polarxiong.com>)

采用HandlerThread就是利用Android的Message Queue来异步处理帧数据。流程简单来说就是 onPreviewFrame() 调用时将帧数据封装为Message，发送给HandlerThread，HandlerThread在新的线程获取Message，对帧数据进行处理。因为发送Message所需时间很短，所以不会造成帧数据丢失。

动手实践 (<https://www.polarxiong.com/project.htm>)

实现

关于 (<https://www.polarxiong.com/about.htm>)

新建 ProcessWithHandlerThread 类，内容为



```
Java
1 public class ProcessWithHandlerThread extends HandlerThread implements
   (https://github.com/zhanlong)
2     private static final String TAG = "HandlerThread";
3     public static final int WHAT_PROCESS_FRAME = 1;
4
5     public ProcessWithHandlerThread(String name) {
6         super(name);
7         start();
8
9     }
10
11     @Override
12     public boolean handleMessage(Message msg) {
13         switch (msg.what) {
14             case WHAT_PROCESS_FRAME:
15                 byte[] frameData = (byte[]) msg.obj;
16                 processFrame(frameData);
17                 return true;
18             default:
19                 return false;
20         }
21     }
22
23     private void processFrame(byte[] frameData) {
24         Log.i(TAG, "test");
25     }
26 }
```

`ProcessWithHandlerThread` 继承 `HandlerThread` 和 `Handler.Callback` 接口, 此接口实现 `handleMessage()` 方法, 用来处理获得的 `Message`。帧数据被封装到 `Message` 的 `obj` 属性中, 用 `what` 进行标记。 `processFrame()` 即处理帧数据, 这里仅作示例。
[Penguin \(https://www.polarxiong.com\)](https://www.polarxiong.com)
[文章列表 \(https://www.polarxiong.com/list.html\)](https://www.polarxiong.com/list.html)

下面要在 `CameraPreview` 中实例化 `ProcessWithHandlerThread`, 绑定接口, 封装帧数据, 以及发送 `Message`。

在实践 (<https://www.polarxiong.com/about.html>) 中添加新的成员变量

```
1 | private static final int PROCESS_WITH_HANDLER_THREAD = 1;
2 |
3 | private int processType = PROCESS_WITH_HANDLER_THREAD;
4 |
5 | private ProcessWithHandlerThread processFrameHandlerThread;
6 | private Handler processFrameHandler;
```

关于 (<https://www.polarxiong.com/about.html>)

在构造函数末尾增加

```
1 | switch (processType) {
2 |     case PROCESS_WITH_HANDLER_THREAD:
3 |         processFrameHandlerThread = new ProcessWithHandlerThread("processFrameHandlerThread");
4 |         processFrameHandler = new Handler(processFrameHandlerThread.getLooper());
5 |         break;
6 | }
```

注意这里的 `new Handler()` 同时也在绑定接口, 让 `ProcessWithHandlerThread` 处理接收到的 `Message`。

修改 `onPreviewFrame()` 为

```
1 | public void onPreviewFrame(byte[] data, Camera camera) {
2 |     switch (processType) {
3 |         case PROCESS_WITH_HANDLER_THREAD:
4 |             processFrameHandler.obtainMessage(ProcessWithHandlerThread.MESSAGE_PREVIEW_FRAME, data, 0, camera);
5 |             break;
6 |     }
7 | }
```

这里将帧数据 `data` 封装为 `Message`，并发送出去。
Penguin (<https://www.polarxiong.com>)
不忘初心，方得始终
运行试试

文章目录 (<https://www.polarxiong.com/list.html>)

改 `processFrame()` 进行测试。

文章目录 (<https://www.polarxiong.com/list.html>)

分析

动手实践 (<https://www.polarxiong.com/project.html>)

这种方法就是灵活套用了Android的Handler机制，借助其消息队列模型

Message Queue解决问题。存在的问题就是帧数据都封装为Message一股

脑丢给Message Queue会不会超出限度，不过目前还没遇到。另一问题就是Handler机制可能过于庞大，相对于拿来处理这个问题不太“轻量级”。

Queue (<https://github.com/zhantong>)

简介

Queue方法就是利用Queue建立帧数据队列，`onPreviewFrame()` 负责向队尾添加帧数据，而由处理方法在队头取出帧数据并进行处理，Queue就是缓冲和提供接口的角色。

实现

新建 `ProcessWithQueue` 类，内容为

Java

```

1  Penguin(https://www.polarxiong.com) Thread {
2  不忘初心, 方得始终; private static final String TAG = "Queue";
3      private LinkedBlockingQueue<byte[]> mQueue;
4
5  文章列表(https://www.polarxiong.com) public ProcessWithQueue(LinkedBlockingQueue<byte[]> frameQueue) {
6      mQueue = frameQueue;
7
8  文章目录(https://www.polarxiong.com/list.html) start() {
9      }
10
11  动手实践(https://www.polarxiong.com/project.html) @Override
12  关于(https://www.polarxiong.com/about.html) public void run() {
13      byte[] frameData = null;
14      try {
15          frameData = mQueue.take();
16      } catch (InterruptedException e) {
17          e.printStackTrace();
18      }
19      processFrame(frameData);
20  }
21  }
22
23  private void processFrame(byte[] frameData) {
24      Log.i(TAG, "test");
25  }
26  }

```

ProcessWithQueue 实例化时由外部提供Queue。为能够独立处理帧数据以及随时处理帧数据，ProcessWithQueue 继承 Thread，并重载了 run() 方法。run() 方法中的死循环用来随时处理Queue中的帧数据，mQueue.take() 在队列空时阻塞，因此不会造成循环导致的CPU占用。processFrame() 即处理帧数据，这里仅作示例。

下面要在 CameraPreview 中创建队列并实例化 ProcessWithQueue，将帧数据加入到队列中。

在 CameraPreview 中添加新的成员变量

Java

👉 [Penguin \(https://www.polarxiong.com\)](https://www.polarxiong.com)

不忘初心, 方得始终

文章列表 (<https://www.polarxiong.com>).

将

文章目录 (<https://www.polarxiong.com/list.html>)

```
1 private int processType = PROCESS_WITH_THREAD_POOL;
```

动手实践 (<https://www.polarxiong.com/project.html>)

Java

修改为

关于 (<https://www.polarxiong.com/about.html>)

```
1 private int processType = PROCESS_WITH_QUEUE;
```

Java

在构造函数的switch中加入

```
1 case PROCESS_WITH_QUEUE:
2     frameQueue = new LinkedBlockingQueue<>();
3     processFrameQueue = new ProcessWithQueue(frameQueue);
4     break;
```

Java

这里使用 `LinkedBlockingQueue` 满足并发性要求, 由于只操作队头和队尾, 采用链表结构。

在 `onPreviewFrame()` 的switch中加入

```
1 case PROCESS_WITH_QUEUE:
2     try {
3         frameQueue.put(data);
4     } catch (InterruptedException e) {
5         e.printStackTrace();
6     }
7     break;
```

Java

将帧数据加入到队尾。

运行试试

现在运行APP, 在logcat中会出现大量的"test", 你也可以自己修

✚ 改 processFrame() 进行测试。
 Penguin (<https://www.polarxiong.com>)
 忘初心，方得始终
 分析

文章种方法可以简单理解为对之前的HandlerThread方法的简化，仅用 LinkedBlockingQueue 来实现缓冲，并且自己写出队列处理方法。这种方法同样也没有避开之前说的缺点，如果队列中的帧数据不能及时处理，就会造成队列过长，占用大量内存。但优点就是实现简单方便。
 动手实践 (<https://www.polarxiong.com/project.htm>)

AsyncTask

关于 (<https://www.polarxiong.com/about.htm>)

简介

AsyncTask方法就是用到了Android的AsyncTask类，这里就不详细介绍了。
 简单来说每次调用AsyncTask都会创建一个异步处理事件来异步执行指定的方法，在这里就是将普通的帧数据处理方法交给AsyncTask去执行。

实现

新建 ProcessWithAsyncTask 类，内容为

```

1  public class ProcessWithAsyncTask extends AsyncTask<byte[], Void, String> {
2      private static final String TAG = "AsyncTask";
3
4      @Override
5      protected String doInBackground(byte[]... params) {
6          processFrame(params[0]);
7          return "test";
8      }
9
10     private void processFrame(byte[] frameData) {
11         Log.i(TAG, "test");
12     }
13 }
  
```

ProcessWithAsyncTask 继承 AsyncTask，重载 doInBackground() 方法，输入为 byte[]，返回 String。doInBackground() 内的代码就是在异步执行，这里就是 processFrame()，处理帧数据，这里仅作示例。

下面要在 `CameraPreview` 中实例化 `ProcessWithAsyncTask`，将帧数据交给 `AsyncTask`。与之前介绍的方法不一样，每次处理新的帧数据都要实例化一个新的 `ProcessWithAsyncTask` 并执行。

文章列表 (<https://www.polarxiong.com>)
在 `CameraPreview` 中添加新的成员变量

文章目录 (<https://www.polarxiong.com/list.html>)
1 | `private static final int PROCESS_WITH_ASYNC_TASK = 3;` Java

动手实践 (<https://www.polarxiong.com/project.html>)

关于 (<https://www.polarxiong.com/about.html>)
1 | `private int processType = PROCESS_WITH_QUEUE;` Java

修改为

 (<https://github.com/polarxiong>)
1 | `processType = PROCESS_WITH_ASYNC_TASK;` Java

在 `onPreviewFrame()` 的switch中加入

```
1 | case PROCESS_WITH_ASYNC_TASK:
2 |     new ProcessWithAsyncTask().execute(data);
3 |     break;
```

Java

实例化一个新的 `ProcessWithAsyncTask`，向其传递帧数据 `data` 并执行。

运行试试

现在运行APP，在logcat中会出现大量的"test"，你也可以自己修改 `processFrame()` 进行测试。

分析

这种方法代码简单，但理解其底层实现有难度。`AsyncTask`实际是利用到了线程池技术，可以实现异步和并发。其相对之前的方法的优点就在于并发性高，但也不能无穷并发下去，还是会受到帧处理时间的制约。另外根据官方文档中的介绍，`AsyncTask`的出现主要是为解决UI线程通信的问题，所以在这里算旁门左道了。`AsyncTask`相比前面的方法少了“主控”的部分，可能满足不了某些要求。

ThreadPool

不忘初心，方得始终

简介

文章列表 (<https://www.polarxiong.com>)

ThreadPool方法主要用到的是Java的ThreadPoolExecutor类，想必之前的AsyncTask就显得更底层一些。通过手动建立线程池，来实现帧数据的并发处理。

动手实践 (<https://www.polarxiong.com/project.html>)

实现

新建ProcessWithThreadPool类，内容为

```

1 public class ProcessWithThreadPool {
2     private static final String TAG = "ThreadPool";
3     private static final int KEEP_ALIVE_TIME = 10;
4     private static final TimeUnit TIME_UNIT = TimeUnit.SECONDS;
5     private BlockingQueue<Runnable> workQueue;
6     private ThreadPoolExecutor mThreadPool;
7
8     public ProcessWithThreadPool() {
9         int corePoolSize = Runtime.getRuntime().availableProcessors();
10        int maximumPoolSize = corePoolSize * 2;
11        workQueue = new LinkedBlockingQueue<>();
12        mThreadPool = new ThreadPoolExecutor(corePoolSize, maximumPoolSize,
13    }
14
15    public synchronized void post(final byte[] frameData) {
16        mThreadPool.execute(new Runnable() {
17            @Override
18            public void run() {
19                processFrame(frameData);
20            }
21        });
22    }
23
24    private void processFrame(byte[] frameData) {
25        Log.i(TAG, "test");
26    }
27    }
  
```

Java

ProcessWithThreadPool 构造函数建立线程池，corePoolSize 为并发度，这里就是处理器核心个数，线程池大小 maximumPoolSize 则被设置为并发度的两倍。post() 则用来通过线程池执行帧数据处理方法。processFrame() 即处理帧数据，这里仅作示例。
[Penguin \(https://www.polarxiong.com\)](https://www.polarxiong.com)
[文章列表 \(https://www.polarxiong.com\)](https://www.polarxiong.com)

下面要在 CameraPreview 中实例化 ProcessWithThreadPool，将帧数据交给 ThreadPoo

在实践 (https://www.polarxiong.com) 中添加新的成员变量

关于 (https://www.polarxiong.com/about) Java

```
1 | private static final int PROCESS_WITH_THREAD_POOL = 4;
2 |
3 | private ProcessWithThreadPool processFrameThreadPool;
```

将 (https://github.com/zhanlong)

1 | private int processType = PROCESS_WITH_ASYNC_TASK; Java

修改为

1 | private int processType = PROCESS_WITH_THREAD_POOL; Java

在构造函数的switch中加入

1 | case PROCESS_WITH_THREAD_POOL: Java
 2 | processFrameThreadPool = new ProcessWithThreadPool();
 3 | break;

在 onPreviewFrame() 的switch中加入

1 | case PROCESS_WITH_THREAD_POOL: Java
 2 | processFrameThreadPool.post(data);
 3 | break;

将帧数据交给ThreadPool。

运行试试

现在运行APP，在logcat中会出现大量的"test"，你也可以自己修改，processFrame()进行测试。
不忘初心，方得始终

分析

文章列表 (<https://www.polarxiong.com>).

ThreadPool方法相比AsyncTask代码更清晰，显得不太“玄乎”，但两者的意思是一致的。ThreadPool方法在建立线程池时有了更多定制化的空间，但同样没能避免AsyncTask方法的缺点。

动手实践 (<https://www.polarxiong.com/project.htm>)

一点唠叨

关于 (<https://www.polarxiong.com/about.htm>)

上面介绍的诸多方法都只是大概描述了处理的思想，在实际使用时还要根据需求去修改，但大体是这样的流程。因为实时处理缺乏完善的测试方法，所以bug也会经常存在，还需要非常小心地去排查；比如处理的帧中丢失了两三帧就很难发现，即使发现了也不容易找出出错的方法，还需要大量的测试。

上面介绍的这些方法都是根据我踩的无数坑总结出来的，因为一直没找到高质量的介绍实时预览帧处理的文章，所以把自己知道的一些知识贡献出来，能够帮到有需要的人就算达到目的了。

关于帧数据和YUV格式等的实际处理问题，可以参考我之前写的一些Android视频解码和YUV格式解析的文章，也希望能够帮到你。

DEMO

本文实现的相机APP源码都放在GitHub上，如果需要请点击 [zhantong/AndroidCamera-ProcessFrames](https://github.com/zhantong/AndroidCamera-ProcessFrames) (<https://github.com/zhantong/AndroidCamera-ProcessFrames>)。

参考

- Camera | Android Developers (<https://developer.android.com/reference/android/hardware/Camera.html>)

- Camera.PreviewCallback | Android Developers (<https://developer.android.com/reference/android/hardware/Camera.PreviewCallback.html>)
- android - Best use of HandlerThread over other similar classes - Stack Overflow (<http://stackoverflow.com/questions/18149964/best-use-of-handlerthread-over-other-similar-classes>)
- Using concurrency to improve speed and performance in Android – Medium (<https://medium.com/@ali-muzaffar/using-concurrency-and-speed-and-performance-on-android-d00ab4c5c8e3#.awqwikas5>)
- HandlerThread | Android Developers (<https://developer.android.com/reference/android/os/HandlerThread.html>)
- LinkedBlockingQueue | Android Developers (<https://developer.android.com/reference/java/util/concurrent/LinkedBlockingQueue.html>)
- AsyncTask | Android Developers (<https://developer.android.com/reference/android/os/AsyncTask.html>)
- ThreadPoolExecutor (Java Platform SE 7) (<https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html>)

已有 36 条评论

vicemiami

🕒 2017-11-23 13:56 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1495>)

博主，如果使用了MediaRecorder进行录像，此时在很多机型上将无法触发onPreviewFrame，有什么好的解决办法么

回复

Penguin

🕒 2017-11-23 16:33 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1496>)

有一个解决办法是用onPreviewFrame()结合MediaCodec来模拟MediaRecorder，具体就是从onPreviewFrame()拿到每一帧的数据（这时候你可以干自己想干的事情），然后把帧数据喂给

MediaCodec来硬编码, 这样从效率上来说是可行的, 但我不太确定视频的质量与MediaRecorder是否有区别。
Penguin (<https://www.polarxiong.com>)
不忘初心, 方得始终

回复

文章列表 (<https://www.polarxiong.com>).

vicemiami

文章目录 ([https://www.polarxiong.com/archives/Android相机开发\(六\): 高效实时处理预览帧数据](https://www.polarxiong.com/archives/Android相机开发(六): 高效实时处理预览帧数据))
动手实践 (<https://www.polarxiong.com/project.html>)
关于 (<https://www.polarxiong.com/about.html>)

(<https://github.com/zhanton>)

我现在改成了用TextureView来做相机预览了, 然后开线程自己去回调TextureView.getBitmap的结果, 就不用onPreviewFrame了, 感觉这种办法不是很好, 但是也能先凑合...MediaCodec之前也试了, 但是问题很多, 自己不太会用, 博主如果有好的例子的话希望分享一下:)

回复

Penguin

2017-11-27 13:44 ([https://www.polarxiong.com/archives/Android相机开发\(六\): 高效实时处理预览帧数据](https://www.polarxiong.com/archives/Android相机开发(六): 高效实时处理预览帧数据))

抱歉回复晚了, getBitamp()可以用但是肯定不能达到30fps, 你要是帧速率没有要求的话这个方法倒是可以用的。

例子的话网上应该有一些示例代码, 但MediaCodec确实难以调试~

回复

👤 Penguin (https://www.polarxiong.com)

不忘初心，方得始终

📅 2017-11-08 16:31 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1473>)

文章列表 (<https://www.polarxiong.com>)

博主有没有考虑写一篇基于camera2 API的实时处理预览帧的文章呢？最近要测试一个c++人脸检测库在手机上的表现，希望fps不会限制到库的表现。

动手实践 (<https://www.polarxiong.com/project.html>)

回复

关于 (<https://www.polarxiong.com/about.html>)

Penguin



📅 2017-11-08 16:40 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1474>)



(<https://github.com/zhantong>)

目前有计划写一份基于Camera2的相机开发教程，但限于时间和精力，短期内可能指望不上咯～你可以看看一些基于Camera2的相机开发DEMO（比如Google自己写的），套用上这里的处理部分就好了。

回复


hfesg

📅 2017-08-30 20:00 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1381>)

非常感谢博主！要是早点儿看到这篇博客就能省去我大量的时间了！我也需要处理预览帧中的每一帧数据，但一开始就是将所有处理结果都写在了onPreviewFrame()中，后来参考了一些人脸检测Demo中的做法，用arraycopy()方法先将数据传出再处理，掉帧的情况少了很多，现在准备试试博主的方法。

最后要咨询博主一个问题～我目前对获取到的预览帧中的某一块区域感兴趣，希望截取下来单独进行处理，但每两帧之间感兴趣区域的位置和大小可能都不固定（每一帧的位置可以确定）。谷歌、百度都不知道怎么搜索，跪

求博主大大提供一些方法或者思路！

 Penguin (<https://www.polarxiong.com>)

不忘初心，方得始终

回复

文章列表 (<https://www.polarxiong.com>)


 2017-09-10 13:06 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1391>)


文章目录 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1391>)

动手实践 (<https://www.polarxiong.com/project.html#comment-1391>)

抱歉这几天太忙了~

关于 (<https://www.polarxiong.com/about.html>)




 (<https://github.com/hfzhang>)

我写过定位和识别二维码的相关代码，说说我的理解。“截取”的实现一般就是定位的过程了，如果你一定要“截取”帧中的一部分保存下来的话，这个是有一定难度的，我现在想到的方法是定位到后把对应区域的矩形NV21数据抠出来保存，等跑完了再开始处理抠出来的数据（因为33ms不太可能把NV21压成jpg）。

如果你只是需要处理的话，还是尽量优化代码让整个处理过程控制在33ms内咯

回复


hfesg


 2017-09-10 14:30 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1395>)

非常感谢博主的回答！我现在也在考虑先将数据得到，之后再处理，目前正在获取NV21中的Y数据分量。感觉自己给自己挖的坑太深了。。哎。。

回复

Penguin

 2017-09-10 16:11 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1395>)

 Penguin (<https://www.polarxiong.com>)

忘初心，方得始终

<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1397>)

另外只用作调试的话，你不如先用相机拍视频拿到文章列表 (<https://www.polarxiong.com/project.html>) 视频文件，然后要怎么折腾怎么折腾了，说不定还能直接解成RGB帧了开始愉快玩耍~

文章目录 (<https://www.polarxiong.com/list.html>)

回复

动手实践 (<https://www.polarxiong.com/project.html>)

关于 (<https://www.polarxiong.com/about.html>)

  (<https://github.com/polarxiong>)

 2017-09-10 16:20 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1399>)

嗯，前期就是用视频文件处理，一些东西已经做好了，想实时处理，谁知道碰到这么多事儿。博主，还有一个疑问，onPreviewFrame的回调data[]的取值范围是-128~127，但是NV21的Y分量规定是0~255，我现在提取了Y分量，但是是负值，请问该如何处理？或者说有没有更好的处理方式？真是麻烦你啦~

回复

Penguin

 2017-09-10 17:22

(<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1401>)

哈哈~ Java基础不过关。因为Java的byte没有无符号类型，直

接data[i]&0xff转int就行了

 Penguin (https://www.polarxiong.com)

不忘初心，方得始终

回复

文章列表 (https://www.polarxiong.com),

hfesg

🕒 2017-09-10 16:40

文章目录 (https://www.polarxiong.com/list.html)

(https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1400)

动手实践 (https://www.polarxiong.com/project.html)

关于 (https://www.polarxiong.com/about.html)



 (https://github.com/zhaoyang1994)

https://stackoverflow.com/questions/5272388/extract-black-and-white-image-from-android-cameras-nv21-format/12702836#12702836

这个问题搞定~

回复

niuzj

🕒 2017-08-26 17:42 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1374)

博主你好，我这里现在需要拍摄1：1的视频，但是好多手机不支持1：1拍摄，所以我想拿到预览帧数据后，裁剪这些数据，请问该怎么做呢？

回复

📄 Penguin (https://www.polarxiong.com)
不忘初心，方得始终


文章列表 (https://www.polarxiong.com)


文章目录 (https://www.polarxiong.com/list.html)


动手实践 (https://www.polarxiong.com/project.html)

关于 (https://www.polarxiong.com/about.html)



 (https://github.com/zhanlong)

 Penguin (https://www.polarxiong.com)


 2017-08-30 09:00 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91%E5%85%AD%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1379)

文章列表 (https://www.polarxiong.com)


预览帧数据是NV21格式的数组，你需要做的是先得到预览帧的分辨率，这样知道数组的内容分布，然后理解NV21格式，就可以试着新建一个数组然后把需要的内容抠出来拼成1:1分辨率的NV21格式数组然后继续处理啦。


动手实践 (https://www.polarxiong.com/project.html)

不过你说要拍成视频，我就很怀疑这么做能不能满足效率要求了，关于 (https://www.polarxiong.com/about.html) 你可以先试着处理每一帧的耗时。



回复


 (https://github.com/zhantong)

 2017-09-08 16:50 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91%E5%85%AD%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1385)

先录制，录制完后一并使用ffmpeg剪裁处理可以么？

回复

Penguin

 2017-09-10 13:05 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91%E5%85%AD%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1390)

当然可以啦，你可以看看ffmpeg有没有提供相关的API，不需要实时的话思路就开阔多了

回复

👤 Penguin (<https://www.polarxiong.com>)

📅 2017-08-20 15:57 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1372>)

🔍 文章列表 (<https://www.polarxiong.com>)

📄 文章目录 (<https://www.polarxiong.com/index.html>)

📄 默认的AsyncTask不是只有一个队列在顺序执行吗

回复

📄 动手实践 (<https://www.polarxiong.com/project.html>)

Penguin

📄 关于 (<https://www.polarxiong.com/about.html>)

📅 2017-08-30 08:52 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1377>)



(<https://github.com/zhanton>)

抱歉这两天太忙没看到。

确实（很久以前）改造后的AsyncTask是串行执行的，所以我说是“旁门左道”，但是确实是在新的线程里处理呢。

回复

zhong

📅 2017-07-31 17:24 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1361>)

博主您好，看了您的文章收货很大，之前用的相机预览程序很耗内存，改成博主的效果好多了。不过我实际使用的时候有个问题，就是我需要将相机帧进行处理，希望把帧数据转换成RGB的矩阵格式。在我之前用的程序中相机帧数据格式为CvCameraViewFrame，可以直接转换成矩阵形式。不知道博主Byte格式的帧数据能不能也转换成包含RGB像素值的矩阵格式？？谢谢博主！

回复

🐧 Penguin (<https://www.polarxiong.com>)

不忘初心，方得始终

🕒 2017-07-31 23:24 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1363>)

文章列表 (<https://www.polarxiong.com>)

文章目录 (<https://www.polarxiong.com/list.html>)

其实就是NV21转RGB了，这个网上有很多相关的示例，包括

Android上的OpenCV也有API提供支持。简单点就是当场算，缺点

动手实践 (<https://www.polarxiong.com/project.html>)

是慢；复杂点就是OpenGL来加速，缺点就是OpenGL写起来有点

复杂；最后就是用OpenCV这类的框架了，缺点是太重量级。

关于 (<https://www.polarxiong.com/about.html>)

回复



cai (<https://github.com/zhanton>)

🕒 2016-12-16 13:49 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-982>)

你好楼主，有个问题想请教一下，如何在调用安卓自身视频录制方法的同时获取帧数据进行实时分析，查阅了很多文献，好像两个方法是不能同时运行的，不知道你是否有什么办法解决。现在我唯一的思路是不调用安卓本身的视频录制，自己手动对帧数据进行处理并转成视频，但这种方法的缺陷在于：

如果对每一帧实时处理，会导致掉帧。如果将从onPreviewFrame方法返回的每一帧NV21数据存在内存中，耗费太大。

望不吝赐教！

回复

Penguin

🕒 2016-12-16 17:05 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-985>)

最近有点忙，没时间尝试能不能录制时同时获取帧，你可以试试在

绑定onPreviewFrame的情况下(就是实时获取每一帧),同时调用录像的API,看着onPreviewFrame还能不能拿到数据,不能的话忘初心,方得始终,只能说目前是不行的。

文章列表(<https://www.polarxiong.com/>)
把收到的每一帧拼成视频是可行的,记得用MediaCodec拼,效率很高的。

文章目录(<https://www.polarxiong.com/list.html>)
掉帧是因为onPreviewFrame()方法耗时过长(函数内容过多),或者onPreviewFrame线程太多任务。照我那种onPreviewFrame拿到数据马上放队列的方法是不会出现掉帧的(我目前就是用的这个方法,而且我需要处理每一帧,可以保证一秒钟是有正好30帧的)。这个内存消耗在我这里就是队列的消耗了,这个是一个很头疼的问题,你要是真想要保证每一帧都处理的话,就必须保证处理过程耗时33ms之内,即保证队列一定为空,否则时间长了队列会爆掉。另一个可能的方法是把数据写在硬盘里,不过你算算需要的写入速度也是很可观的,这就要自己取舍了。



(<https://github.com/zhanton>)

回复

Cai

📅 2016-12-17 01:25 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-987>)

谢谢回复,现在情况是这样的,我有两种选择,1:将onpreviewframe返回的nv21数据存放在list中后续一并处理,这种方法肯定不掉帧,但缺点是内存开销太大,因为nv21数据格式本身太大。2:对传来的每帧实时处理,例如转成bitmap(格式更小)再存放在list中,后续再一起转成video.这种方法就会导致掉帧,因为转bitmap挺耗时间。几点疑问:1.您说mediacodec是将一个list的图像转成video对吧,应该不能写在onpreviewframe中做实时处理吧。2.将nv21转bitmap很耗时,需要0.1左右,所以掉帧,您知道有没有什么快速转换的方法吗?哪怕转成其他格式也行,因为这步唯一的目的是减少list的内存消耗,后续转视频不需要实时。万分感谢!

回复

Penguin


2016-12-17 11:18 (<https://www.polarxiong.com/archives>
<https://www.polarxiong.com>
Android%E7%9B%B8%E6%BC%BA%E5%BC%80%E5

%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6
/www.polarxiong.com).
%8D%AE.html#comment-989)

`View.onPreviewFrame()` 干的事情要尽可能少；我应该会开新的线程取queue，一份交给MediaCodec，一份

转成其他格式的话，我劝你不要这么急（不过你也可以继续尝试），因为任何压缩算法都需要消耗更多时间。`png`显得非常宝贵。或者你可以找找有没有快速的`byte[]`压缩算法；另外转`bitmap`是应对存储为`jpeg`等图片格式的，和你的需求不太吻合。




 (<https://github.com/zhantong>)

📅 2016-12-17 13:05 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-990>)

回复

([https://www.polarxiong.com/archives
/Android%E7%9B%B8%E6%9C%BA
%E5%BC%80%E5%8F%91-%E5%85
%AD-%E9%AB%98%E6%95%88%E5](https://www.polarxiong.com/archives/%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5)

Penguin (<https://www.polarxiong.com>)

不忘初心，方得始终

%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-994)

文章列表 (<https://www.polarxiong.com>) 你是说Array List？主要是没有concurrent支持。

文章目录 (<https://www.polarxiong.com/list.html>) java.util.concurrent.LinkedBlockingQueue的优点在于一是线程安全；二是

动手实践 (<https://www.polarxiong.com/project.html>) 因为队列只会从头取出从尾放入，适合链表结构，避免了数组接口动态扩展的性能问题。

关于 (<https://www.polarxiong.com/about.html>)

回复



(<https://github.com/zhenyong>)

Penguin

2016-12-17 13:58

(<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-991>)

我觉得队列好处就是简单可控，不复杂也没有额外的开销，用着方便罢了。

Android对每个APP的内存是有上限的，队列爆了APP就自动崩溃了，我比较倾向于给队列设个最大容量，这样就算丢帧也能保证APP不崩溃；还有就是注意不要内存泄漏咯，让GC抓紧回收。

回复

Cai

2016-12-18 07:20

(<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-991>)

 Penguin (<https://www.polarxiong.com>)

不忘初心，方得始终

文章列表 (<https://www.polarxiong.com>)

文章目录 (<https://www.polarxiong.com/list.html>)

动手实践 (<https://www.polarxiong.com>)

关于 (<https://www.polarxiong.com/about.html>)



(<https://github.com/zhantong>)

回复

%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-993)

但如果不做数据压缩和处理，也不考虑内存消耗，把每一帧直接放进一个List中后续再转成视频不是更直接吗？相比用List的话，用队列能帮助减少内存消耗么？

Cris

🕒2016-12-09 17:28 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-964>)

恩，谢谢！博主的回复真的是神速。

回复

Cris

🕒2016-12-09 10:14 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-964>)

%B8%A7%E6%95%B0%E6%8D%AE.html#comment-962)

👤 Penguin (<https://www.polarxiong.com>)

博主，你好！我又来问个小问题了，我在理解了你博客的基础上进行了实时预览帧处理，但发现存在一个问题，我存储了预览帧图像，将它与自定义相机拍得的图像进行对比，发现二者差距很大，预览和图片分辨率我设的都一样，其他参数也一样。评论发不了对比图片，我在CSDN上也提了这个问题。<http://ask.csdn.net/questions/350770>

文章目录 (<https://www.polarxiong.com/list.html>)

我不清楚是不是preview data和picture data是不是本身就不同，还是预览时需要指定一些参数？

动手实践 (<https://www.polarxiong.com/project.html>)

回复

关于 (<http://www.polarxiong.com/about.html>)

Penguin



🕒 2016-12-09 16:41 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-963>)



(<https://github.com/zhagtonk>)

因为一个是preview，另一个是take picture，两者调用的API不同。预览帧可以理解为拍视频，一般可以支持到30fps，分辨率最大1080P；但拍照的话是调用的拍照API，不是onPreviewFrame()，没有API提供像onPreviewFrame()那样自动拍照的功能（Camera2可能可以），支持的最大fps可能只有10+，分辨率可以达到4K。

你打开相机就是开启了预览，这时预览帧全部显示在屏幕上；你点击拍照就是调用API专门拍照，不是预览帧咯。

回复

Huiyuan

🕒 2017-10-26 14:05 (<https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91-%E5%85%AD-%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1450>)

博主，您好。想请教下，现在有需要要在扫描过程中得到分辨率很大的帧数据，应该怎么做呢？

回复

👤 Penguin (https://www.polarxiong.com)

不忘初心，方得始终

📅 2017-10-26 15:06 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91%E5%85%AD%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1456)

文章列表 (https://www.polarxiong.com)

文章目录 (https://www.polarxiong.com/list.html)

动手实践 (https://www.polarxiong.com/project.html) 这个就是把预览帧分辨率调到最大了，在开始相机预览前用getSupportedPreviewSizes()拿到支持的

最大分辨率，然后再set到相机就好了，可以参考

关于 (https://www.polarxiong.com/about.html) 之前讲偏好设置的文章。



如果你是想得到像拍照那么高的分辨率的话，单纯的预览是做不到的，那你就要试试预览时调用API拍照了，我不知道能不能行~



(https://github.com/zhantong)

回复

Huiyuan

📅 2017-10-26 15:40 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91%E5%85%AD%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1457)

如果把扫描过程当作在持续拍照，这种方式可行吗？

回复

Penguin

📅 2017-10-26 15:59 (https://www.polarxiong.com/archives/Android%E7%9B%B8%E6%9C%BA%E5%BC%80%E5%8F%91%E5%85%AD%E9%AB%98%E6%95%88%E5%AE%9E%E6%97%B6%E5%A4%84%E7%90%86%E9%A2%84%E8%A7%88%E5%B8%A7%E6%95%B0%E6%8D%AE.html#comment-1458)

Q

添加新评论

*

填写Email以便有新的回复时能够及时通知您，您的Email不会被

http://

*

提交评论

2017/12/6 下午2:34