FUFO-VP

Report on Android Camera Image Type

*Author: Vũ Minh Phong*

# Introduction

This report is about the main focus of VP project in getting image from Android Camera.

The main contents of this report are:

* How to get image frame from Camera in Android.
* What is YUV format and detail of some specific format that related to VP.

# How to get image frame from Camera in Android

## The use of public class [android.hardware.Camera]

The Camera class is used to set image capture settings, start/stop preview, snap pictures, and retrieve frames for encoding for video. This class is a client for the Camera service, which manages the actual camera hardware.

These following instructions show some basic uses of this class:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To take pictures with this class, use the following steps:

1. Obtain an instance of Camera from [open(int)](http://developer.android.com/reference/android/hardware/Camera.html#open%28int%29).
2. Get existing (default) settings with [getParameters()](http://developer.android.com/reference/android/hardware/Camera.html#getParameters%28%29).
3. If necessary, modify the returned [Camera.Parameters](http://developer.android.com/reference/android/hardware/Camera.Parameters.html) object and call [setParameters(Camera.Parameters)](http://developer.android.com/reference/android/hardware/Camera.html#setParameters%28android.hardware.Camera.Parameters%29).
4. If desired, call [setDisplayOrientation(int)](http://developer.android.com/reference/android/hardware/Camera.html#setDisplayOrientation%28int%29).
5. **Important**: Pass a fully initialized [SurfaceHolder](http://developer.android.com/reference/android/view/SurfaceHolder.html) to [setPreviewDisplay(SurfaceHolder)](http://developer.android.com/reference/android/hardware/Camera.html#setPreviewDisplay%28android.view.SurfaceHolder%29). Without a surface, the camera will be unable to start the preview.
6. **Important**: Call [startPreview()](http://developer.android.com/reference/android/hardware/Camera.html#startPreview%28%29) to start updating the preview surface. Preview must be started before you can take a picture.
7. When you want, call [takePicture(Camera.ShutterCallback, Camera.PictureCallback, Camera.PictureCallback, Camera.PictureCallback)](http://developer.android.com/reference/android/hardware/Camera.html#takePicture%28android.hardware.Camera.ShutterCallback,%20android.hardware.Camera.PictureCallback,%20android.hardware.Camera.PictureCallback,%20android.hardware.Camera.PictureCallback%29) to capture a photo. Wait for the callbacks to provide the actual image data.
8. After taking a picture, preview display will have stopped. To take more photos, call [startPreview()](http://developer.android.com/reference/android/hardware/Camera.html#startPreview%28%29) again first.
9. Call [stopPreview()](http://developer.android.com/reference/android/hardware/Camera.html#stopPreview%28%29) to stop updating the preview surface.
10. **Important:** Call [release()](http://developer.android.com/reference/android/hardware/Camera.html#release%28%29) to release the camera for use by other applications. Applications should release the camera immediately in [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause%28%29) (and re-[open()](http://developer.android.com/reference/android/hardware/Camera.html#open%28%29) it in [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume%28%29)).

To quickly switch to video recording mode, use these steps:

1. Obtain and initialize a Camera and start preview as described above.
2. Call [unlock()](http://developer.android.com/reference/android/hardware/Camera.html#unlock%28%29) to allow the media process to access the camera.
3. Pass the camera to [setCamera(Camera)](http://developer.android.com/reference/android/media/MediaRecorder.html#setCamera%28android.hardware.Camera%29). See [MediaRecorder](http://developer.android.com/reference/android/media/MediaRecorder.html) information about video recording.
4. When finished recording, call [reconnect()](http://developer.android.com/reference/android/hardware/Camera.html#reconnect%28%29) to re-acquire and re-lock the camera.
5. If desired, restart preview and take more photos or videos.
6. Call [stopPreview()](http://developer.android.com/reference/android/hardware/Camera.html#stopPreview%28%29) and [release()](http://developer.android.com/reference/android/hardware/Camera.html#release%28%29) as described above.

This class is not thread-safe, and is meant for use from one event thread. Most long-running operations (preview, focus, photo capture, etc) happen asynchronously and invoke callbacks as necessary. Callbacks will be invoked on the event thread [open(int)](http://developer.android.com/reference/android/hardware/Camera.html#open%28int%29) was called from. This class's methods must never be called from multiple threads at once.

**Caution:** Different Android-powered devices may have different hardware specifications, such as megapixel ratings and auto-focus capabilities. In order for your application to be compatible with more devices, you should not make assumptions about the device camera specifications.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Instruction on how to get frame data

Frame data are provided by this class in term of "Preview back buffer" or "Preview frame".

To access to the preview frame, simply use the interface [Camera.PictureCallback](http://developer.android.com/reference/android/hardware/Camera.PictureCallback.html)

This interface provide the following method which give us a pointer to the data sequence:

#### public abstract void onPreviewFrame (byte[] data, [Camera](http://developer.android.com/reference/android/hardware/Camera.html) camera)

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Called as preview frames are displayed. This callback is invoked on the event thread [open(int)](http://developer.android.com/reference/android/hardware/Camera.html#open%28int%29) was called from.

##### Parameters

|  |  |
| --- | --- |
| **data** | the contents of the preview frame in the format defined by [ImageFormat](http://developer.android.com/reference/android/graphics/ImageFormat.html), which can be queried with [getPreviewFormat()](http://developer.android.com/reference/android/hardware/Camera.Parameters.html#getPreviewFormat%28%29). If [setPreviewFormat(int)](http://developer.android.com/reference/android/hardware/Camera.Parameters.html#setPreviewFormat%28int%29) is never called, the default will be the YCbCr\_420\_SP (NV21) format. |
| **camera** | the Camera service object. |

## Frame data type

There are 7 types of frame we could get from the preview buffer:

|  |  |  |  |
| --- | --- | --- | --- |
| **Constants** | | | |
| int | [JPEG](http://developer.android.com/reference/android/graphics/ImageFormat.html#JPEG) | Encoded formats. |
| int | [NV16](http://developer.android.com/reference/android/graphics/ImageFormat.html#NV16) | YCbCr format, used for video. |
| int | [NV21](http://developer.android.com/reference/android/graphics/ImageFormat.html#NV21) | YCrCb format used for images, which uses the NV21 encoding format. |
| int | [RGB\_565](http://developer.android.com/reference/android/graphics/ImageFormat.html#RGB_565) | RGB format used for pictures encoded as RGB\_565 see [setPictureFormat(int)](http://developer.android.com/reference/android/hardware/Camera.Parameters.html#setPictureFormat%28int%29). |
| int | [UNKNOWN](http://developer.android.com/reference/android/graphics/ImageFormat.html#UNKNOWN) |  |
| int | [YUY2](http://developer.android.com/reference/android/graphics/ImageFormat.html#YUY2) | YCbCr format used for images, which uses YUYV (YUY2) encoding format. |
| int | [YV12](http://developer.android.com/reference/android/graphics/ImageFormat.html#YV12) | Android YUV format: This format is exposed to software decoders and applications. |

NOTE: For our FUFO application, the best format should be [YV12](http://developer.android.com/reference/android/graphics/ImageFormat.html#YV12). But [NV21](http://developer.android.com/reference/android/graphics/ImageFormat.html#NV21) is also OK. The reasons for this conclusion will be shown by the next part of this report.

# What is YUV format and detail of some specific format that related to VP

## What is YUV:

YUV is a data model for storage picture which was invented for reducing bandwidth for chrominance components compare to RGB model. Its way of encoding a color image or video is similar to human perception.

There are three fragments of each YUV pixel:

- Y: the brightness of a picture.

- U and V: The color information of the picture.

The following images show a picture with full YUV, then a picture with only Y, then only U and only V component.



YUV model has many variant for storage YUV data, such as: YUV444p, YUV420p, YUV422p, YUVY, YV12, YV21...

Each of these variant has its own way of sorting Y, U, V in its data stream. This leads to the complexity of conversion of YUV to RGB.

## YUV conversion to RGB

Today, almost of the display device understand RBG model as their only standard to display image. Therefore the conversion of YUV to RGB is very important.

The value of Y is the sum of weighted R,G,B and U and V are computed as scaled different between Y' and B and R value. Because of this, many research has been taken to define the best scale for these conversion leading into an endless debate. In our FUFO project we will just choose the following scale provided by Avery Lee:

R = Y + 1.402 (V-128)

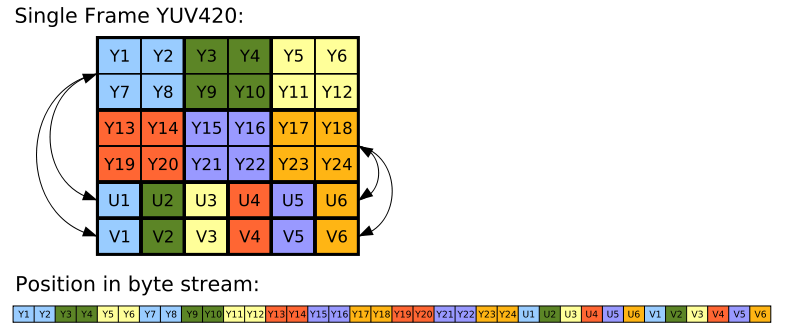
G = Y - 0.34414 (U-128) - 0.71414 (V-128)

B = Y + 1.772 (U-128)

Reference here: <http://www.fourcc.org/fccyvrgb.php>

## Detail of YV12 (YUV420 planar or Y'UV420p)

Y'UV420p is a planar format, which means that Y',U,V values are grouped together instead of interspersed. The following example picture will show the way Y',U,V values are organized:



In FUFO, the best type of image frame should be this type of YUV. The reason for this is very obvious: I have done the code fragment for converting YV12 to RGB24 and it took me 1 months to do it.

Here is the piece of code in C++:

#define clip(value)static\_cast<unsigned char>(min( 255,max( 0,value )))

// The following function will take an array of YUV420p data and return an array of RGB

// data

void ConvertUtility::ConvertYUV420ToRGB24(const unsigned char\* yuv, unsigned char\* rgb,int width, int height) {

int yLength = width \* height;

int rgbWidth = width \* 3;

const unsigned char\* pY = yuv;

const unsigned char\* pU = pY + yLength;

const unsigned char\* pV = pU + (yLength >> 2);

unsigned char\* pRgb = rgb;

int y00, y01, y10, y11, v, u;

for (int row = 0; row < height; row += 2) {

for (int col = 0; col < width; col += 2) {

y00 = pY[0];

y01 = pY[1];

y10 = pY[width];

y11 = pY[width + 1];

v = pV[0] -128;

u = pU[0] -128;

// Pixel at 00 position

pRgb[0] = clip(y00 + 1.772f \* u); //B

pRgb[1] = clip(y00 - 0.344f \* u - 0.714f \* v); //G

pRgb[2] = clip(y00 + 1.402f \* v); //R

// Pixel at 01 position

pRgb[3] = clip(y01 + 1.772f \* u); //B

pRgb[4] = clip(y01 - 0.344f \* u - 0.714f \* v); //G

pRgb[5] = clip(y01 + 1.402f \* v); //R

// Pixel at 10 position

pRgb[rgbWidth] = clip(y10 + 1.772f \* u); //B

pRgb[rgbWidth + 1] = clip(y10 - 0.344f \* u - 0.714f \* v); //G

pRgb[rgbWidth + 2] = clip(y10 + 1.402f \* v); //R

// Pixel at 11 position

pRgb[rgbWidth + 3] = clip(y11 + 1.772f \* u); //B

pRgb[rgbWidth + 4] = clip(y11 - 0.344f \* u - 0.714f \* v); //G

pRgb[rgbWidth + 5] = clip(y11 + 1.402f \* v); //R

pY += 2;

pV += 1;

pU += 1;

pRgb += 6;

}

pY += width;

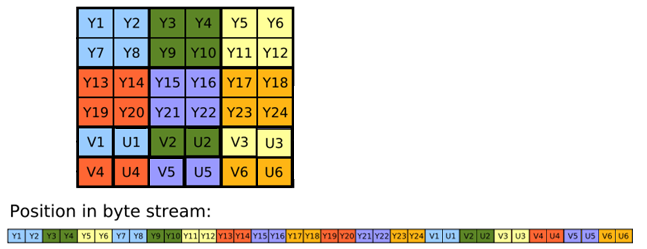
pRgb += rgbWidth;

}

}

## Detail of NV21 (YUV420sp)

YUV420 semi planar has the same information values as YUV420p, but the order of U and V value is reversely interspersed. The following example picture will describe the way Y,U,V values organized:



This format of YUV are more complicated to convert into RGB24. Therefore the effort for converting NV21 into RGB will be greater, so be the delay time. If possible, we should not use this format.

THE END