Android + OpenCV 3.1.0

1. Download Android Studio
http://developer.android.com/sdk/index.html

Install Android Studio + install Android SDK Platform.

> Open Android Studio > Configure > SDK Manager : install Android 14, 21, 22, 23 SDK

> Configure > Preferences > Appearance > Darcula theme (if you want dark theme)

1. Download Android Studio latest version (alternative to old guide - **for NDK** based projects): https://developer.android.com/studio/

Install & run it, then open the android project (ex: “Drive assist” - which has ndk and jni).

Install **NDK** from within Android Studio, then download the old NDK version:
`Android-ndk-r15c-linux-x86_64.zip` (for Ubuntu)

Download old NDK from here: https://developer.android.com/ndk/downloads/older_releases

Copy it to the default NDK folder (replace with your user on Ubuntu):
`/home/razvan/Android/Sdk/ndk-bundle`

Project should run fine.

2. Download OpenCV for Android
http://opencv.org/downloads.html

You can view some of the existing OpenCV for Android samples, for example the app provided is based on the image manipulations sample. To open it, do the following: Android Studio > Import project (Eclipse) > Open extracted archive of OpenCV > samples > image manipulations

3. Open the downloaded project in Android Studio
Android Studio > Open Project > Open the downloaded project

Usual problems: build.gradle > minimum version: 14, target version 21, compile version 21 (set these values for both build.gradle files) & make sure you download these Android versions form the SDK Manager.

Download and install OpenCV Manager application for Android (a dialogue pop-up will open when you launch the sample app and it will redirect you to the Play Store from where you can download the OCV Manager app).

Basic Android intro:

What you see on the sceen is contained in an Android activity. Activities have lifecycles and they can contain multiple fragments (small re-usable components that have their own lifecycles). Activities and fragments can be designed programatically by instantiating custom views, or by using existing android layouts and views that can be described easily in xml files.

Projects structued into packages, java classes are in src > main > java > package name > MyActivity.java

Layouts and other xml resources are found: src > main > res > drawable (images and other xml resources used for drawing), layout (xml layouts = what you see on the screen), values (xml resources for strings and texts used in the apps)

AndroidManifest.xml - important configuration file containing information about the current app, information that is used by the Android OS; here you can specify the minimum Android version that the app can run on, defines the package name of the application, specifies the components used in the applications (the activities, services, broadcast receivers, content providers), declares which permissions the application uses (ex: permission to access the device camera, the sd card storage to read/write files, permission to use the internet, etc.), lists which libraries the app uses. Read more about AndroidManifest here: http://developer.android.com/guide/topics/manifest/manifest-intro.html

The main Android app components:

1. Activities: dictate the user interface and will handle the user's interaction with the mobile device screen.

2. Services: used to handle the background processing within the application.

3. Broadcast receivers: handle the communication between Android OS and applications.

4. Content providers: handle database and data management.

Other components:

Fragments: represent part of UI within an Activity.
Views: the basic UI elements that can be drawn on the device screen (buttons, lists, etc).
Layouts: view hierarchies used to control the appearance and order of the views.

Intents: messages used to wire and connect multiple components together.
Resources: external elements like strings, constants, drawable pictures or layout files.
Manifest: the configuration file for the Android app.

Read more about app components here:
http://developer.android.com/guide/components/index.html

Basic OpenCV intro:

Images are stored into Mat objects that represent matrices. They are bi-dimensional and feature a width and height.

onCamerFrame() - callback that will contain the current frame captured from the camera
The frame is in a custom opencv object format (CvCameraViewFrame) and the RGBa (RGB + alpha channel) Mat can be accessed easily: Mat rgba = inputFrame.rgba();

> width and height are accessed as follows: rgba.height() (can be saved as int), rgba.width().

Go through an image pixel by pixel (use only for single image processing, not recommended for video processing -- also remove the logs to speed up the app):

```
for (int i = 0; i < rows; i++)
   for (int j = 0; i < cols; j++) {
       double[] pixelValue = rgba.get(i, j);
       if (pixelValue != null) {
           for (int x = 0; x < channels; x++) {
               pixelValue[x] = pixelValue[x] + 50;
               Log.i(TAG, "Pixel value: " + pixelValue[x]);
           }
           Log.i(TAG, "Pixel value -----------");
           rgba.put(i, j, pixelValue);
       }
   }
```

Or you can use the following version (tested and working):

```
int size = (int) (rgba.total() * rgba.channels());
double[] temp = new double[size];          // check out for out of memory errors here when allocating
such a large array
rgba.get(0, 0, temp);
for (int i = 0; i < size; i++) {
    if (temp[i] < 120) {
        temp[i] = 0;    // black
    } else {
        temp[i] = 255;  // white
    }
}
tmp.put(0, 0, temp);
rgba = tmp.clone();
```

The code above will do a simple binarization of the input image contained in the rgba Mat object.

Project basics:

There are 3 activities:

The first one is *MainActivity* and it containts 2 buttons which will redirect the user to the other 2 activities:

- *ImageManipulationsActivity*: the base activity example from the OpenCV fragmework, this class contains methods and callbacks to process the live camera feed (process images acquired form the camera), it has a button used to select the processing type (filter) that can be applied: RGB, Canny, Sobel, etc.

- *ImageActivity*: simple activity used to process single images; you will see a button named "MENU" which will open a file picker from where you can select an image from the phone memory (internal memory or sd card); the images are opened using a "picker" (this is a separate library that implements the functionality of picking/selecting a file from the memory); once an image is opened it will be processed in 2 methods (processImage() and processBitmap() - default selected will be processBitmap())

The processing is done in Java, therefore some functions of the OpenCV library will not be the same as in the C++ version. There is the possibility of using Android & OpenCV in C++ using NDK: you can check out the sample named "mixed processing" from the OpenCV for Android archive that you have extracted (or download OpenCV 2.4.x - where you will find the sample project). You will also need to install Android NDK and configure it, for this please consult:

http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/android_dev_intro.html#native-development-in-c

NOTE: this document and the sample application might be updated during the course of this semester, try to check the website and this document periodically.

For questions: razvan.itu@cs.utcluj.ro
More info at: http://users.utcluj.ro/~razvanitu/teaching.html#pi