Porting ViSP to Android devices

Akshay Sharma
GSoC 2018 Proposal

Synopsis

ViSP (Visual Servoing Platform) is a modular cross platform library that allows prototyping and developing applications using visual tracking and visual servoing techniques. It's developed using the researches done by Inria Lagadic team and can be useful in robotics, computer vision, augmented reality and computer animation. ViSP is already packaged for Linux, Windows, Mac and IOS. The objective of my project is to offer the ViSP community an SDK for Android, containing:

- Build scripts
- Adapting existing files
- Doxygen Documentation
- Tutorials and Videos
- Sample Executables

About Me

The Student

- **Name** Akshay Sharma
- **Email** akshay.sharma.mat16@iitbhu.ac.in
- **Github** https://github.com/AKS1996
- **Personal Website** http://www.imakshay.com

The Institute

- **University** Indian Institute of Technology (BHU), Varanasi
- **Major** Mathematics and Computing
- **Year** Sophomore
- **Degree** Integrated Master's Degree (B.Tech and M.Tech)

My Background

I have been programming since 10th grade in my school. I began coding in **Python** and later on switched to **C++** for 2 years. During this period I also learnt basics of web development and android development.

After joining IIT, I spent my first year to broaden and deepen my skills in android development and to learn computer vision. In the summer of 2017, I collaborated in a project named VOCOWA, a SLAM driven robot wheelchair for a period of 8 weeks. Towards the winter of 2017, I expanded my interests towards augmenting objects in reality, mostly using OpenCV.

My Motivation

Immersive Reality has fascinated me for past few months. Given my interest in android, I didn't want to shift to web or IOS platforms for learning it. There are propriety software available for AR/VR development in Android but I have been preferring and advocating open source. OpenCV4Android was also an option, but the hefty apk's size and slower detection and tracking were nagging problems. ViSP is a great choice for me since it's open sourced, lightweight, has many inbuilt features and is focused on visual tracking and servoing. I'll also learn about cross platform compilation.

Benefits to ViSP

Offering an SDK to Android community has several benefits. Provided that current generation of handsets have fast and efficient processors, high quality cameras, and sufficient internet availability, expanding the code base to Android phones will help ViSP reach lot of Android developers who can fascinate and enrich lives of many people by developing immersive apps.

Contributions

I made a pull request #307, in which I've added a few **CMake** files and adapted already existing ones. This enables one to build the ViSP libraries for android architecture(specifying API version, platform architecture and other build flags) using Android NDK rev16.

Build Scripts

These are used to build the SDK. I've used only **CMake** to build and make libraries.

1. android.toolchain.cmake file, adapted for ViSP build and **install.sh** script for building the SDK on Unix systems using bash command line
2. **Build_sdk.py** file and allied python service test scripts, like **test_ndk.py**

Adapting Existing files

Since ViSP wasn't initially targeted for android system, a good number of files need editing

1. Adding build flags for the files in the **CMake** folder, including **CMakeLists.txt**
2. Improving the existing ViSP source code to work on efficiently on android environment
3. Improving the existing ViSP source code to work **on Android older than API 24**, which doesn't provide support for standard IO streams(stderr, stdout, etc.) and other 64 bit operations
4. Addition of sample templates under the **CMake/template** folder
5. Addition of appropriate methods or classes, whenever required to resolve an issue

Providing Doxygen Documentation

To each method added or adapted, both in **Java** and **C++**, I'll provide either an inline or block documentation

1. Explaining the changes made in the **CMake** files
2. Explaining the changes made in the existing source code
3. Corresponding documentation for any function or class added, in **java** or **cpp**, with argument description
4. Appropriate comments for the logic in tutorials
5. Any **TODO** or **FIXME** for future edits/improvements

Tutorials and Videos

All the tutorials will be added to existing tutorials in a dedicated section called "ViSP for Android". Along with source code, they will include appropriate **readme**'s or **md** files. These will address topics like:

1. How to build ViSP Android SDK from source code.
2. How to use an Android live camera, convert images in ViSP **vpImage** data and apply simple image processing algorithms part of the **imgproc** module
3. How to interact with the user to initialize an algorithm. A good example would be the selection from the touch screen of a region of interest used to start a **blob tracking** and display the result of the tracking (position of the cog, roi...)
4. Detect one or more April tags using **vpAprilTag** class, display the position of the tag in overlay

During the final stage of evaluation, I'll try to demonstrate or provide Walkthrough videos (on YouTube) for above tutorials.
Note: I will also support bug tracking and submission as a wiki page along with a list of common issues developers might face while building.

Executables

Executable will include

1. Sample android **apk** for each tutorial specified above
2. Prebuilt SDK containing native ViSP libraries built for android (the **.so** or **.a** files), associated header files, **java** source code and 3rd party libraries.

Evaluation Benchmarks

I expect to complete the following milestones, in **chronological** order during each phase

Phase	**Deliverable**
Community Bonding Period	1. Discuss the project with mentors in detail, considering the need of a new camera framework 2. Study OpenCV4Android SDK in detail 3. Building ViSP library using NDK

May 14 - June 8, 2018	1. **Imgproc** module demo in android 2. **vpAprilTag** demo in real time
June 18 - July 6, 2018	1. Optimizing existing ViSP source code for larger device coverage and lag-free real time experience 2. Wrapper **JNI**/ **Java** code for native **C++** functions as a module named **visp_java** 3. ViSP 3rd party library support like Eigen, OpenCV
July 16 - August 6, 2018	1. Complete Doxygen documentation 2. Tutorial and demos 4. Sample executables(**apk's**) 5. Support bug tracking and submission

References

- **VOCOWA**: Voice Controlled Autonomous Wheelchair. A project that I undertook under mentorship of Dr. M.K. Meshram, Associate Prof, Dept of Electrical Engineering, IIT Varanasi. My role was to implement a simple RANSAC algorithm, to be later used in ROS for implementing a Kalman Filter.
- **android.toolchain.cmake**: Instead of using the toolchain file used to build OpenCV4Android, I'm using the official Android toolchain file, with NDK rev16.
- **visp_java:** OpenCV too has such a module, opencv_java