Hochschule Fulda
University of Applied Sciences

MOBILE APPLICATION ON SCENE RECONSTRUCTION AND VISUALIZATION FROM COMMUNITY PHOTO COLLECTION

Web & Mobile Application Development
MSc in Global Software Development
Winter Semester 2016, Fulda

Submitted By:

S M Touhidur Rahman
Matrikel No: 547073

Muneeb Noor
Matrikel No: 444530

Chosen Paper

Scene Reconstruction And Visualization From Community Photo Collection

By Noah Snavely, Ian Simon, Michael Goesele, Richard Szeliski, and Steven M. Seitz

Introduction

The paper is about using internet to gather a large resource of images of particular places e.g. famous tourists spots and generate models using these images. There are number of steps described such as:

- Finding features and correspondence
- Dense Scene Reconstruction
- Scene Summarization and Segmentation
- Finding Paths From Series of Photos
- Scene Visualization

The project is suitable for good hardware setup and it wouldn't be feasible to generate 3d models using the current hardware capabilities of the hand held devices. However, we took an attempt in doing so by working on the first part of the project which is related to finding features and correspondence between images. We tried to cover the steps where small computing devices can be used efficiently such as feature detection and finding correspondence, capture images with features and panoramical views.

Implementation

Used Tools, Technologies and Libraries

Tools	Usage Rights	URL
Android Studio *Version: 2.3*	Freely distributed	https://developer.android.com/studio/index.html
Android SDK *Version: 23, 25*	"	https://developer.android.com/studio/releases/platform-tools.html
Android NDK (Native Development Kit)	"	https://developer.android.com/ndk/index.html
OpenCV4Android	Open Source	http://opencv.org/platforms/android/

Version: 3.2		
Gradle Build Tool	Open Source	https://gradle.org/

App: Scenator2

Developed By: S M Touhidur Rahman

Github URL: http://github.com/touhidrahman/scenator2

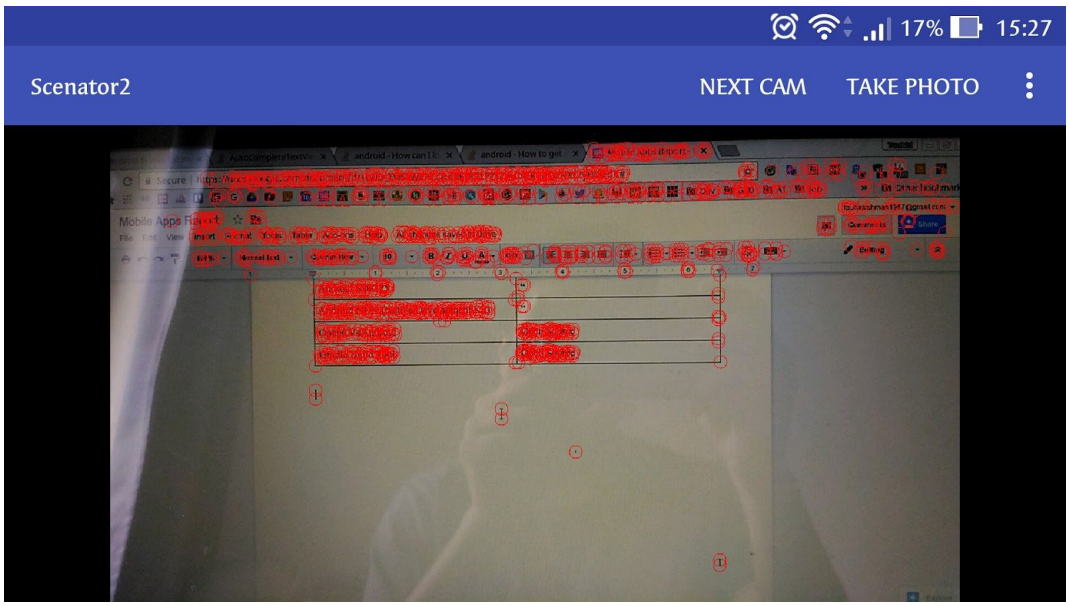

Fig: The Scenator2 app detecting features in the live camera stream

Architecture

The application is developed primarily for Android devices with ARMEABI v7a architecture. However, additional architectures such as ARM64, ARMEABI, mips, x86, x86_64 can be supported easily.

Features

The purpose of this app is to take photos of key object and detect features. The camera view displays features by red circles found in the camera stream live. The app can switch between both front and rear camera and also change resolution of output image. It can be used to capture photos of key object from different angle and later use the images for further steps such as Dense Scene Reconstruction.

Structure

Fig: Code Structure of the App (partial)

Although the app is written using Java, part of it also incorporates C++ codes which calls the OpenCV4android library for feature detection via Java Native Invocation (JNI) calls. The C++ codes are compiled at build time for each architecture.

There are few algorithms for feature detection. SIFT, SURF, FAST and ORB are few among many. In our case, the algorithm used for feature detection is 'FAST' - Features from Accelerated Segment Test.

App: Images Features Matching and Stitching

Developed by: Muneeb Noor

Github URL: https://github.com/muneebnoor/mobandweb

Introduction

Images features matching is the first step to achieve "scene reconstruction from photos". Since the mobile phones have not got the required capability to render models from images, I concentrated on creating an app which would explain the working of different algorithms in image features matching. This app can act as a starting point for students/learners who would like to know and test different feature matching algorithms.

Features

- Following algorithms (feature detectors) have been implemented to generate keypoints in a particular image and display it: Harris, FAST, ORB, AGAST, MSER, GFTTD, KAZE, AKAZE, STAR, SIFT and SURF
- Following algorithms have been implemented as a descriptor selection choice for feature matching: KAZE, AKAZE, ORB, FREAK, BRISK, BRIEF, LUCID, LATCH, DAISY, SIFT and SURF
- FlannBased, BruteForce L1 and BruteForce L2 matchers have been implemented to match float descriptors while BruteForce Hamming and BruteForce HammingLUT have been implemented to match uchar descriptors.
- Image stitching has been implemented using openCV function as well as using custom technique. The custom implementation of the image stitcher is achieved by using SURF feature detector and descriptor, along with FlannBased matcher to find the matches between the images. After the matches are found, we find the homography by using findhomography function of openCV. We use homography matrix to warp the matching images.
- Functions to load, process and display images using mat object has also been implemented as part of the app.

Implementation:

The app is built using the android platform libraries of OpenCV 3.2.0. Since we needed to test SIFT and SURF, it was required to build the OpenCV_contrib library. To do that, I manually compiled both OpenCV and openCV_contrib library. All other tools and technologies have been same as mentioned in the "Implementation" section.

All the algorithms have been implemented using both Java Wrappers and native code. Only native code implementation has been added in the app because the performance of implementation using Java wrappers was much worse than performance of native code implementation. The reason for bad performance is that multiple JNI calls are needed when java wrappers are used, while only a single JNI call is enough in the case of native implementation.

Future Work:

Although the app is a great starting point for the computer vision learners, there are some improvements which would further improve the app. There is no option to enter input parameters for algorithms in the current version, which would stop the learners from testing the algorithms using different input parameters. The app also lacks the ability to offer different implementations of algorithms to the users, which may restrict the users in selecting the best option according to their requirements.

Work is also needed to be done on GUI of the app, which would greatly improve the user experience. The app doesn't help users in selecting the valid and invalid combinations for the features matching, which may frustrate the new users because selecting the invalid combination results in closing of the app.

Observations:

- Although the OpenCV community is vast, there is not much help available and new users of OpenCV may find it hard to start working with the library.
- There are inadequate resources available online which may help the Windows users to compile and use the library.
- Comparative to Python OpenCV developers, there is very little help available online for C++ developers.