Final Project Report

GP-Cam

Kyle Rush, Ben Sourbeer

Advised by Professor Spletzer

**Table of Contents**

Executive Summary

Introduction and Motivation

Project Planning

Software Design

Implementation Results

Verification and Validation

Documentation

Future Work

Discussion

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Ben Sourbeer  and Kyle Rush | 12/14/2013 | Initial Draft | 1.0 |

**1. Executive Summary**

The GP-Cam project was created with the intentions of developing a pan/tilt/zoom camera controller that would be able to focus on a dynamic target using its GPS coordinates. We wanted to essentially create an autonomous camera man so that an individual could capture whatever they desire without continuous manual camera adjustment. This was to be achieved by creating a Matlab interface under Ubuntu that will receive GPS coordinates over a wifi ad-hoc connection and move the camera accordingly. The GPS coordinates are sent from an Android smartphone with an application created specifically for this purpose. The Matlab software also functions with any device that communicates it’s GPS coordinates in the correct format.

We succeeded in creating the software to perform this task given perfectly accurate GPS coordinates. At this current time, GPS accuracy on smartphones is not perfectly accurate, as we show in our testing results. The result is a system that works well in theory, however is not currently a practical solution. In the near future, this software will become much more viable whenever GPS technology in smartphones becomes more reliable.

**2. Introduction and Motivation**

2.1. The Problem

Current motion tracking devices utilize graphical algorithms to capture any moving target, and the capture of only a specific target is not possible.

2.2. Why is it interesting/important?

The desire to capture moments of our lives has become exponentially more popular with the rise of compact video capture devices such as smartphones and other portable devices.

There is also a large demand for devices to capture ourselves doing something without the need for a camera man. A prime example of this would be the sharp rise in popularity of the GoPro camera:

* GoPro sales have at least doubled every year since their debut in 2004
* 2012 sales: 2.3 million cameras; grossing $521 million(*Forbes* March, 2013)

2.3. Proposed Solution

This project will utilize the GPS capabilities of the common smartphone in order to video record a particular target with a pan/tilt/zoom camera. The smartphone will be located on the desired target, which will allow for the pan/tilt/zoom camera to locate the specific target. This will allow for the video capture of a specific target autonomously.

**3. Project Planning**

3.1. Project Organization

Both team members will be interfacing with our client. We will meet every Tuesday and Thursday in the VADER Lab from 1-4 PM. Our client is typically there to answer any questions about the requirements. If there are no deliverables for a week, we will submit a progress report to our client.

3.2. Deliverables

9/27: Project Planning Requirement

10/4: Weekly Progress Report

10/11: Project Design

10/18: Weekly Progress Report, Working Android Application

10/25: Weekly Progress Report

11/1: Weekly Progress Report

11/8: Project Implementation

11/15: Weekly Progress Report

11/22: Weekly Progress Report

11/26: Poster (For Presentation)

12/4: Poster (Final Copy)

3.3. Technical Requirements

* Development Platforms: Ubuntu and Android
* Development Software / languages: Matlab and Android SDK (Eclipse)

3.4. Performance Requirements

* USB interface for camera video
* Acquire and display images at 30Hz in Matlab
* tracking software agnostic to source of GPS coordinates
* Non-blocking tracking
* Tracking controller should run at GPS update speed
* Tracking in UTM coordinates
* Calibration accuracy will be within 1 degree

3.5. Minimum scope requirements

* An android application which can successfully calibrate a camera’s location and broadcast it’s own location via wireless technology. Converts GPS to UTM coordinates.
* An application for the pan/tilt/zoom camera that can utilize the smartphone’s data to track the target
* Specifically wifi for wireless technology
* Needs to regulate the user specified field of view based on calculated distance

3.6. Remaining Requirements

* Accuracy
  + Calibration accuracy within 1 degree
* Speed
  + Camera at 30Hz
  + Tracking controller running at GPS update speed
* Non-blocking tracking

3.7. Risk Areas

* GPS Altitude: GPS might not be able to accurately track altitude. Preliminary tests have been inconclusive.
* Accuracy: Between the altitude and the general accuracy of GPS, we might not be able to very accurately track a target.

3.8. Interfaces

* Camera has an existing interface within Matlab. We’ll be using that for basic camera control.
* We will be using the Android API and SDK

3.9. Self Evaluation

* Every week we will determine what needs to be done within the next week, and reflect on our progress thus far. If we’re falling behind schedule, more lab time will need to be scheduled to supplement our existing 6 hours a week.

3.10. Internal Deliverables

* Basic Android app that accesses phone’s GPS data and translates that data to UTM coordinates
* Matlab camera pan/tilt controller that takes in a set of coordinates and points the camera to that point.
* Matlab driver for analog video capture through USB video converter
* Final Android app that sends GPS coordinates to PC via peer to peer connection
* Final Matlab controller that receives GPS coordinates from peer to peer connection and points the camera to that point

**4. Software Design**

**4.1. System Overview**

There are two very separate components of this project.

1. Android app that sends its GPS coordinates to a computer via wireless ad-hoc network.
2. Software written in Matlab that will analyze the given coordinates and control the PTZ camera appropriately.

This is a very high level view of the system, and there are many more components at the lower levels that will make these two major components possible.

The Android app is a much smaller component than the camera controller. The Android app essentially performs three tasks.

1. Attain device’s GPS data via Android SDK
   1. the functions to accomplish this task are predefined in the SDK. The only aspect that is yet to be determined is the rate at which we will be requesting GPS coordinates. These requests are very intensive on the mobile device’s battery, and there is also concern on the speed of the device’s GPS signal acquisition.
2. Convert Latitude/Longitude coordinates to UTM(Universal Transverse Mercator) coordinates
   1. This is accomplished through some quick arithmetic.
3. Send converted coordinates to laptop via mobile ad-hoc network
   1. The Android SDK has a class for handling peer to peer connections, which is called WifiP2pManager. We can use this to communicate with other devices through wifi using socket communication.

The camera controller will be substantially more complex, as it is the core of this software system. We are fortunate that there is an existing Matlab framework for the camera that enables us to control its movement. The following components are listed in order of completion priority.

1. Gain access to analog video signal from camera and save video to a local file.
   1. The analog video signal is converted to digital using a Hauppauge 610 USB device. This is a device that is compatible with the Video for Linux 2 (V4L2) API that is included in linux kernels starting with 2.5.X.
2. Receive GPS data from peer to peer connection
   1. Use Matlab to connect to ad-hoc network and communicate with GPS device.
3. Perform algorithm to determine relative position of target GPS device
   1. There needs to first be a function to receive the calibration location of the camera.
   2. subsequent locations will be of the target, so every time data is received, there will be a calculation of what the angular difference is between the target and the camera.
   3. This angular difference will be converted into the position changes that need to be applied to the camera. This will be straightforward, as the existing functions to move the camera accept parameters in units of degrees.
   4. The required camera zoom also needs to be calculated by calculating the linear distance between the camera and the target.

**5. Implementation Results**

5.1Android Application

First, upon starting the Android application, it runs a thread to connect to the server running in Matlab. Communication is handled using Java socket connections. After the connection is established, the application will enable the two buttons, labeled “Broadcast GPS Data” and “Calibrate”, as well as display a message on the screen notifying the user that the connection is established.

The user will then press the Calibrate button. The Calibrate button creates a GPSTracker class, which deals with getting the location data from hardware. The GPSTracker class also handles the conversion between Latitude Longitude and UTM coordinates. The last known GPS coordinates are obtained and sent to the server.

Next the user will press Broadcast GPS Data. This button starts a timer which grabs the GPS data from the sensor every 10 milliseconds, and sends it to the server. The function of the Android application was accomplished in full.

5.2 Matlab Application

When the Matlab application is first opened, a Java server is started. The application then waits for a connection to be established with an Android device. Once established, the Application waits for UTM coordinates to be transmitted. Once UTM coordinates are received, the points are saved in the corresponding camera or target location.

If a target location is received, the application will calculate the relative pan and tilt angles it needs to move. This is accomplished by two functions findPanAngle and findTiltAngle. The vector math was implemented with success.

Finally, after the angles are calculated, the Matlab application calls the functions that actually move the camera. These functions were already established prior to beginning this project.

**6. Verification and Validation**

Before the Verification and Validation phase, we set out our plans for testing, which are listed below.

Acquire and display images at 30hz in Matlab

* This is a given statistic by the VLC media player. It has been confirmed that our video is being captured at 30hz

Non-blocking tracking

* Test: provide gps coordinates at varying extremes of pan capabilities
  + verify that new commands are overwriting old commands

Source-agnostic GPS coordinates

* create “dummy” process that provides gps coordinates
  + verify that there are no dependencies on source device

Stability And Robustness

* Run in Matlab application for at least 1 hour to ensure stability
* Open and close the application within Android and Matlab multiple times
* Test every different combination of inputs available to user, make sure Android application functions as intended

Angular Accuracy within 1 degree

* Provide dummy GPS coordinates and measure the direction the camera is pointing versus the direction it should be pointing.

Running at GPS update speeds

* Test (using timestamps) how fast GPS is updating.
* Measure the speed that the application is sending commands to the camera, if too slow we need to correct for the added time

All of these above tests were completed, and they were all verified other than a single requirement. We were unable to measure an angular accuracy within 1 degree, however given perfectly accurate GPS coordinates, there should be no barriers to achieving this. In our tests, we measured an average of 1.92 degrees over a series of 10 still images at varying distances. We were using Google Maps data to test with in order to rule out the inaccuracy of our gps device. This 1.92 degree error is most certainly due to the fact that Google Maps coordinates are rounded to the nearest meter.

**7. Documentation**

See user manual for complete installation instructions and system configuration.

**8. Future Work**

The project has been finished by our project definition, however there are still substantial features that can be added for a more robust and organized implementation. Most significantly is the fact that the video from the camera is currently being captured using VLC media player and not in Matlab. This was attempted to be implemented during this past semester, however we were unable to complete it by the project deadline. Additionally, there is room to smooth out the setup process and the ad-hoc wifi connection startup. More specifically, it would be nicer to be able to start the connection using a Matlab function rather than being forced to open the network manager and a terminal in Ubuntu.

**9. Discussion**

This project has certainly shown that this is a possible solution for camera motion tracking. We know that our software works given exact GPS coordinates, however as previously mentioned, there is a large dependency on the accuracy of GPS devices. Assuming that this technology will progress, our software will become more applicable to real world applications. With that being said, it may have been a wise decision to test the smartphone’s GPS accuracy in our area before venturing into this project. Had we known how inaccurate that these devices are, we may have chosen a different project, or a variation of this project.

We had an opportunity to experience programming in Matlab and Android, which we were not exposed to previously. At the end of the day, both contributors to this project acquired a great deal of invaluable experience in working on a long term project and cooperating with a partner, which is sure to be useful in our futures.