Aerial Photography using a Nokia N95

Mariano I. Lizarraga† David M. Ilstrup* Gabriel Hugh Elkaim† James Davis *

Abstract—**A low cost UAV for aerial photography is constructed and tested. The UAV transfers images as they are captured via 802.11g to a ground station. Photo-mosaics are immediately assembled using SIFT keypoints for automatic image registration.**

Keywords: Aerial Photography, UAV, SIFT

1 Introduction

The availability of current aerial imagery is important for many applications. "Current" may mean a few minutes for scenarios such as forest fire tracking, while a few hours might be acceptable for missing person searches over a region or intelligence gathering for military and law enforcement operations. Obtaining minutes- to hours-old aerial imagery has typically been a capability outside all but the most well funded organizations.

Unmanned Aerial Vehicles (UAVs) are often thought of as strictly military devices, but in fact they are rapidly permeating the space of civilian applications (driven by lower entry-level prices and a skilled and active community of remote/radio control (RC) airplane hobbyists). Current applications for UAVs include border surveillance, whale and other marine mammal tracking, power-line verification, and search and rescue missions in disaster areas [1].

Many of the commercially available UAVs are currently able to relay video and photographs to users on the ground. The usefulness of the imagery that these flying robots relay to their controlling ground station is directly related to how much information the operator can "extract" from the frames being watched. Image quality of video relay, RF noise rejection, and image stabilization have come to play important roles in the overall performance measure of these missions.

While many of the mainstream UAVs have sophisticated equipment onboard to take care of these problems, the price tag has traditionally been on the order of millions of dollars, making them unaffordable for all but military applications.

Recent advances in solid-state sensors and overall reduction of the electronics have made it possible to greatly improve the capabilities of "hobbyist level" systems. Autopilots are now as cheap as $350.00 [2] but video downlink capabilities and automatic high resolution image registration/photomosaicing are generally far beyond the restricted budget of such a project.

In this project we develop a solution that provides high quality continuous (photo-mosaic) view from an aerial platform to small UAVs with small budgets, using a Nokia N95 wireless phone, a personal computer and open source or free software.

The aircraft is flown under pilot control from the ground while position, heading, and imagery from the phone are received within seconds of image capture. The results are assembled into a photo-mosaic and geographically linked using a GoogleEarth^{TM}KML file. If internet is available on site, a visual overlay of the flight course is immediately viewable.

The rest of this document is organized as follows: Section 2 presents previous work and how it relates to this project. Section 3 describes in detail each component of the system and how they interact. Section 4 shows flight test results. Section 5 presents conclusions, discusses some of the weakness of this approach, and recommends some directions for further work.

2 Related Work

The idea of using a wireless phone inside an RC airplane to take pictures is not new. The company Pict'Earth [3] has a yet unreleased commercial proposal very similar to the one described here, but again, the price keeps it out of reach of the typical restricted budget. As evidenced by the demo of their commercial product, the opportunity for improved image registration exists.

Image registration and photomosaicing of related imagery from UAVs is an active research topic. Majumdar, et.al. [4] presented a method for offline registration and mosaicing of collected images from UAVs. This method has proven to give great results but is numerically intensive and somewhat slow. Instead, we make use of the open source *Panorama Tools* [10] to do the mosaicing of the flight images.

Our contribution is to provide a complete system to acquire aerial images, geographically tag them, download them via Wi-Fi to a ground station, and stitch them to-

*University of California Santa Cruz, Baskin School of Engineering, Computer Science Department davei@soe.ucsc.edu, davis@soe.ucsc.edu. † Computer Engineering Department malife@soe.ucsc.edu, elkaim@soe.ucsc.edu.

gether to create a mosaic. All these steps are done with no intervention from the end-user aside from launching the applications in the ground PC and the phone. If internet is available, the downloaded images as well as the airplane's trajectory can be observed in a geo-referenced form in Google Earth. The system is based solely on open source or free tools and can be put together for as little as $170.00 for the RC airplane, assuming the end user already owns the Nokia N95 phone and a wireless router.

3 System Description

The complete setup, shown in Figure 1 consists of an RC airplane that has been modified to hold the Nokia N95 wireless phone inside its fuselage (hand-launched and controlled during flight by a pilot via radio control). Once the airplane is in the air, two applications run on the phone. One takes pictures and modifies their EXIF tags [5] to include the GPS information; the other transmits them down to the ground station through a TCP socket via 802.11g.

The ground station computer runs a server that receives the images. These are handed off to a set of applications which assemble mosaics from sets of three pictures. At the time the pictures are received, the appropriate KML files are also modified to display the received images and update the flight path for display in Google Earth™. The wireless link between the computer on the ground and the phone is serviced by a wireless (802.11g) router with high gain (7dB) antennas.

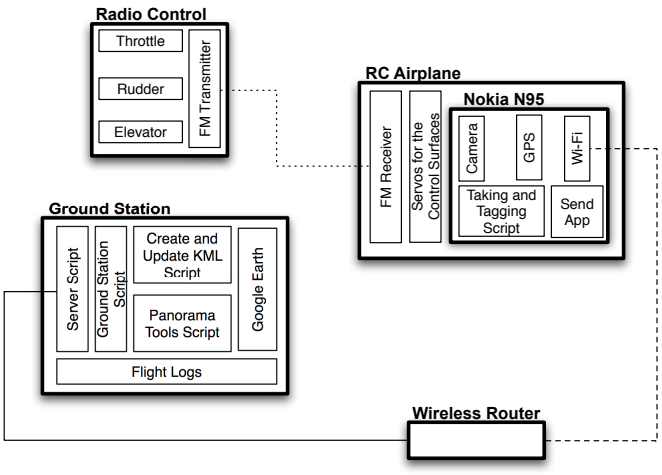

Figure 1: Complete System Setup

There are three main components to the overall system: **(1)** The modified airplane to fly with the phone installed, **(2)** the applications residing on the phone itself, and **(3)** the set of applications comprising the ground station.

The following subsections describe each of the three main components in more detail.

3.1 Airplane Modification and Balancing

The first step to get the phone airborne was to modify a hobby RC airplane to be able to safely house the phone inside its fuselage. For this purpose we tried two different airplanes: FlyZone's SkyFly [6] and HobbyZone's SuperCub [7], both shown on Figure 2. Both proved to be sturdy and relatively easy to fly, but the Skyfly proved to be under-powered and became very unstable in preliminary flights with an attached 130 gm weight simulating the phone. The SuperCub was chosen for the main test flights at UCSC.

The SuperCub's fuselage was modified to safely house the phone during flight (see Figure 2). Since the phone's GPS antenna is located under the keyboard, the fuselage was molded to house the phone with the keyboard extended. Extra precautions were taken to keep the plane aerodynamically stable.

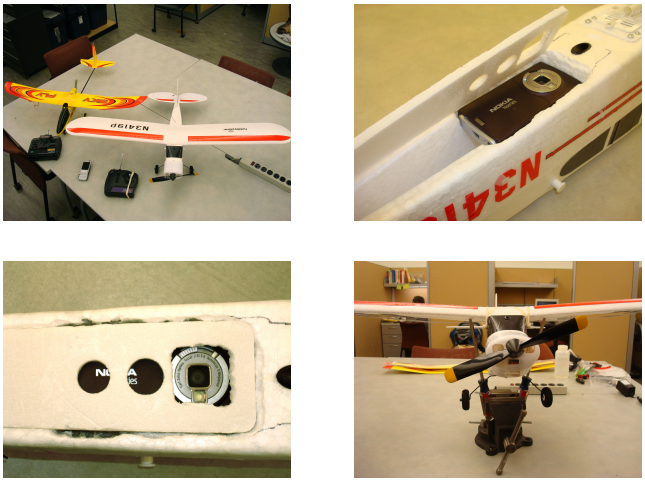

Figure 2: Airplane Modification. *Top Left:* FlyZone's SkyFly (left) and HobbyZone's SuperCub. *Top Right:* Phone installed in the housing. *Bottom Left:* Cover placed on the fuselage. *Bottom Right:* Balancing of the airplane

3.2 The Phone Applications

Two independent applications were developed for the phone. A block diagram of these can be seen in Figure 3. One is a Python script (*PhoneApp.py*) in charge of initializing the Camera and the GPS, taking pictures and geographically-tagging them and saving them on the SD card. The other is a Symbian OS SIS application (SendApp.sisx) that connects to the ground PC via Wi-Fi and periodically scans a local directory to determine whether a new picture has been taken. If so, it sends the picture to the ground PC and tags it as sent.

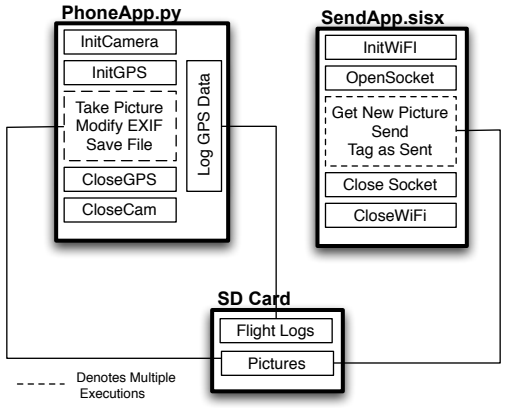

Figure 3: General Architecture of the Phone Applications

3.3 The Ground Station

The ground station application (Figure 4) manages the configuration of the host server for tagged image reception, on-the-fly image registration and update of the flight path in a KML file.

This is the first application run when starting a flight, as the image server must be available when the camera application begins image transmission. The ground stations creates a unique flight directory to contain all flight images, logs and KML files.

The main process of the ground station monitors the progress of the image server. When a new image has been completely received, the KML file's path information is updated and if sufficient files are available, image registration, in sets of three, is performed.

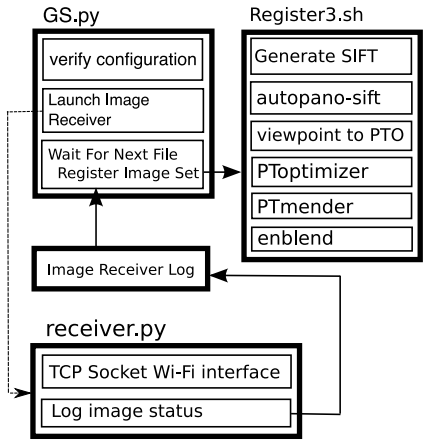

Figure 4: General Architecture of the Ground Station

In image registration, perhaps the most interesting and challenging aspect is establishing correspondence points or *keypoints* between images. The open source software

Table 1: Mean and standard deviation for GPS accuracy

	Mean	Std. Dev
Picture Taking Frequency	3.4886	0.8164
Vertical Accuracy	28.1294	22.8255
Horizontal Accuracy	32.2818	21.0033

autopano-sift [8] automatically establishes these correspondences. It uses Scale Invariant Feature Tracking (SIFT) [9] to identify *keypoints* in each image. The identified correspondences of the processed images are then stored in a '*.pto' file which the ground station software modifies slightly to insert Point Of View (POV) information specific to the phone's camera.

Two tools from the Panorama Tools Library [10] are run to find linear image transformations that minimize the distance error between keypoints (PToptimizer) and then to perform the image transformations (PTmender) required to create the photo-mosaic.

Finally, the *Enblend* application [11] is used to smooth the pixel shading so the seams where images are joined become unnoticeable.

The application is robust, so that all available files will be processed even if a communication interruption occurs.

4 Results

To test the complete setup several flights were completed mainly in two locations: UCSC East Remote parking lot and in McMillan Airfield in Camp Roberts, CA. In the following subsections we present our findings from these flights.

4.1 GPS Accuracy and Picture Taking Latency

One of the key factors to successfully create a photo-mosaic was the ability of the camera to take pictures quickly enough so that pictures were able to overlap. From the flight log we obtained data of how frequently a picture was taken. Figure 5 shows a plot of the separation (in seconds) between pictures.

Since the information being presented in Google Earth^{TM}was very dependent on the accuracy of the phone's GPS module, then we decided to plot the data from the flight logs to establish the expected accuracy of the phone's GPS. Figure 5 shows a plot of the vertical and horizontal accuracy (in meters) of the GPS data collected. Table 1 shows the mean and variance of the image-capture frequency, vertical accuracy, and horizontal accuracy of the GPS data.

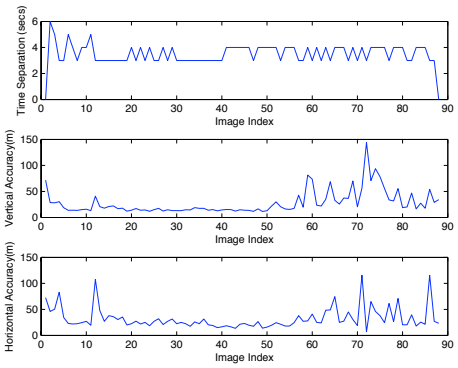

Figure 5: Time separation between pictures. Vertical and horizontal accuracy of the GPS

Figure 6: Multiple pictures of UCSC's east remote parking lot

4.2 Aerial Imagery

Three flights tests with an average of two launches per flight were performed in Fall 2007 on the UCSC campus. Figure 6 shows four pictures of UCSC's east remote parking taken from the airplane.

From the pictures taken in those flights it became clear that the creation of the photo-mosaic was dependent of the skill of the pilot to keep the airplane in level flight and on a straight line. Figure 7 shows a photo-mosaic assembled from a set of pictures taken, with no user interaction. Although it is clear that the "stitching" is not perfect, it is reasonable and clearly the end user can extract plenty of information from the view.

Several flights tests were conducted during Winter 2008 at the McMillan Airfield in Camp Roberts, CA. Figure 8 shows a photo-mosaic produced from some of the images acquired during these tests which were done from one of the UAVs that is part of the Naval Postgraduate School's Rapid Flight Test Prototyping System [13]. Having the phone take pictures from a stabilized platform noticeably improves the results. Finally as images are received Google Earth's^{TM}KML file is modified to display the last known position of the airplane and includes a link to the image taken in that position. The image can be seen in a pop-up "balloon" inside Google Earth^{TM}or by selecting the link in the figure, in a standard image viewer. Figure 9 shows a screenshot of one of the flights.

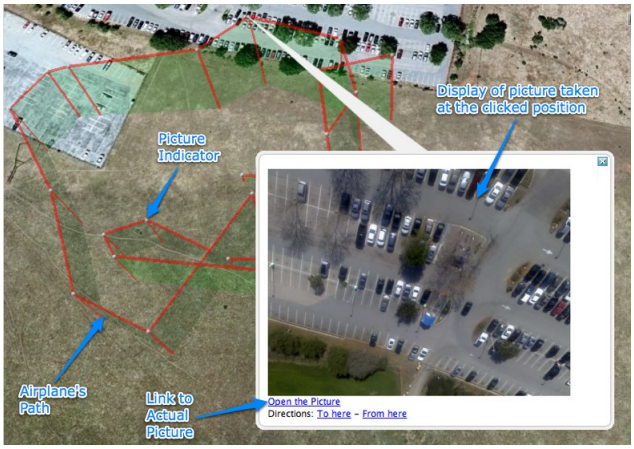

Figure 9: Airplane's path displayed in Google Earth^{TM}

5 Conclusions and Further Work

The system presented successfully takes aerial pictures, on average, every four seconds. Image registration to create the photo mosaics is reasonable provided that the airplane is flying so that successive pictures overlap. The presentation of the pictures and the mosaics in Google Earth^{TM}proves to be very useful to analyze the received pictures.

Although we have presented a complete system that is able to take aerial pictures and create photo-mosaics with them, it is far from complete. Much work is needed on the ground computer software. Three different scripting languages are currently used on the ground station (shell, Python and Perl) which ideally should be merged into one. Currently there is no installer package to put these tools onto a different computer than those used for development. The system remains a fragile prototype, suitable for research, but not yet ready for wide-scale deployment.

System performance is best in straight and level flight, which is difficult under manual control of the UAV. Results from the UAV flight confirm this observation.

The image registration process is limited by the nature of the transformations used by the tools employed. Some method that takes account of parallax effects present in images with large variations in ground distance would also be an improvement.

Figure 7: Final panorama assembled from a picture triplet with no user interaction. East Remote Parking Lot UCSC

Adding roll and pitch to the current EXIF tags of the images could potentially improve the process of image correction and registration.

6 Acknowledgments

Many thanks are due to Natasha Gelfand at Nokia Research in Palo Alto, CA. Thanks to Isaac Kaminer, Vladimir Dobrokhodov, Kevin Jones, and Don Meeks at the NPS for their flight support and interesting ideas. Finally we profoundly thank Andrew Adams at Stanford for providing us with a very efficient Python Module to make use of the phone's camera. This work was partially funded by the Mexican National Science and Technology Council (CONACyT).

References

[1] Nonami, K. ; *Prospect and Recent Research & Development for Civil Use Autonomous Unmanned Aircraft as UAV and MAV*, Journal of System Design and Dynamics, Vol. 1, No. 2, 2007.

[2] UNAV, LLC; *Pico Pilot: World's Smallest Autopilot*, http://www.u-nav.com/picopilot.html.

[3] Pict'Earth, *World on Live: Pict'Earth*, http://www.pictearth.com/.

[4] Majumdar, J., Vinay, S., Selvi, S.; *Registration and Mosaicing for Images Obtained from UAV*, IEEE International Conference on Signal Processing and Communications (SPCOM), 2004.

[5] Technical Standardization Committee on AV & IT Storage Systems and Equipment; *Exchangeable image file format for digital still cameras: Exif Version 2.2* , Japan Electronics and Information Technology Industries Association, 2002.

[6] FlyZone, *FlyZone SkyFly Ready to Fly Radio Control Airplane*, http://www.flyzoneplanes.com/airplanes/hcaa1961_index.html

[7] HobbyZone, *HobbyZone SuperCub Ready to Fly Radio Control Airplane*, http://www.hobbyzone.com/rc_planes_hobbyzone_super_cub.htm

[8] Sabastian Nowozin, *Autopano-Sift, Making panoramas fun*, http://user.cs.tu-berlin.de/~nowozin/autopano-sift/, University of British Columbia, patents apply.

[9] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

[10] Open Source Project, *Panorama Tools - libpano13-2.9.12*, http://panotools.sourceforge.net/.

[11] Open Source Project; Andrew Mihal, *enblend*, http://enblend.sourceforge.net/.

[12] Nokia, *Python for S60*, http://wiki.opensource.nokia.com/projects/PyS60.

Figure 8: Panoramas assembled from picture triplets with no user interaction. Camp Roberts, CA

[13] Dobrokhodov, V.N. and Yakimenko, O.A. and Jones, K.D. and Kaminer, I.I. and Bourakov, E. and Kitsios, I. and Lizarraga, M.., *New Generation of Rapid Flight Test Prototyping System for Small Unmanned Air Vehicles*, in Proceedings of the AIAA Modeling and Simulation Technologies Conference 2007.