Autonomous Flight using a Smartphone as On-Board Processing Unit in GPS-Denied Environments

Michael Leichtfried
michael.leichtfried@tuwien.ac.at

Christoph Kaltenriner
christoph.kaltenrinner@tuwien.ac.at

Annette Mossel
mossel@ims.tuwien.ac.at

Hannes Kaufmann
kaufmann@ims.tuwien.ac.at

Interactive Media Systems Group
Vienna University of Technology
Favoritenstr. 9-11/188/2, 1040 Vienna, Austria

ABSTRACT

In this paper, we present a low-weight and low-cost Unmanned Aerial Vehicle (UAV) for autonomous flight and navigation in GPS-denied environments using an off-the-shelf smartphone as its core on-board processing unit. Thereby, our approach is independent from additional ground hardware and the UAV core unit can be easily replaced with more powerful hardware that simplifies setup updates as well as maintenance. The UAV is able to map, locate and navigate in an unknown indoor environment fusing vision based tracking with inertial and attitude measurements. We choose an algorithmic approach for mapping and localization that does not require GPS coverage of the target area, therefore autonomous indoor navigation is made possible. We demonstrate the UAVs capabilities of mapping, localization and navigation in an unknown 2D marker environment. Our promising results enable future research on 3D self-localization and dense mapping using mobile hardware as the only on-board processing unit.

Categories and Subject Descriptors

Information systems [**Information Systems Applications**]: Mobile Information Processing Systems; Computing Methodologies [**Artificial Intelligence**]: Robotics; Image Processing and Computer Vision [**Scene Analysis**]: Tracking

General Terms

Application

Keywords

Mobile Computing, Aerial Robotics, Mobile Information Processing Systems, Localization and Tracking, Autonomous Flight, GPS-denied Environments

MoMM2013, 2-4 December, 2013, Vienna, Austria

1. MOTIVATION

A UAV (Unmanned Aerial Vehicle) is an aircraft flying without a human pilot on board. They can be deployed e.g. in areas that are too dangerous or unreachable for humans in case of environmental catastrophes. Quadcopters, as a subsytem of UAVs, are compact rotor craft air vehicles with vertical takeoff and landing capability. They can hover like a conventional helicopters but have significant advantages such as mechanical simplicity with no need of a swash-plate mechanism and ease of piloting.

In recent years, there has been extensive research in autonomous flight control of UAVs by either tracking the UAV from outside or by monitoring the environment from the UAV itself (Section 2). Thereby, the UAV's 3D position and orientation are estimated to allow for simultaneous localization and mapping (SLAM) as well as collision free navigation. With an autonomous flight approach, remote control by humans can be omitted and no line of sight between controller and UAV is required.

In outdoor environments, a UAV can easily be localized using GPS[1] data. In GPS-denied areas, such as indoor environments, recent approaches demonstrated autonomous flight fusing vision with inertial measurement data to estimate the UAVs 3D position and orientation. Many of these approaches require a workstation as additional ground hardware for data processing and steering control computation. To increase flexibility of the system, localization, mapping and navigation as well as steering control generation have to be performed on board of the UAV. Recent work demonstrated powerful on-board SLAM approaches by using customized embedded computing platforms. However, the applied hardware platforms are mostly designed for a single use case and can hardly be extended or easily replaced with emerging and more powerful modules.

To increase flexibility and cost efficiency, we demonstrate a UAV based on open source hard- and software that uses an off-the-shelf smartphone as its central on-board processing unit to allow for autonomous flight in indoor environments with no GPS coverage. We present a SLAM and navigation approach that fuses monocular vision data with inertial as well as altitude data. All data processing and steering command generation is performed by a smartphone running Android. The phone is connected to an open hardware micro-

[1]Global Positioning System

controller that ensures stable attitude balancing and sends navigation commands to the motor controllers. For testing and safety reasons, the autonomous navigation can be replaced during run-time with a standard remote control to be able to manually intervene at every moment in time. For evaluation purposes, data analysis as well as visualization, the smartphone sends flight data via a wireless connection to a portable workstation. We demonstrate autonomous localization, mapping and navigation in an unknown 2D environment. Due to the compact construction and accurate localization, precise exploration can be performed within an indoor area of only $1.86\,x\,2.25m$. Automatic lift-off and landing on a user-defined area of $200\,x\,200mm$ is presented.

With the demonstrated work, we provide a technological low-cost flight platform that only requires a commercially available mobile device as core processing unit and hence can play an important role in areas with poor infrastructure (e.g. developing countries) to autonomously perform tasks for search and rescue, inspection and measurements of buildings as well as observation.

1.1 Contribution

This paper presents the following contributions. Firstly, we demonstrate that off-the-shelf mobile hardware such as a smartphone can be applied as central light-weight but computationally powerful on-board processing unit to enable autonomous flight. Hence, our setup does not require any computing ground hardware and control of the UAV is independent from any steering signals sent by a ground station via wireless data transmission. Thus, error prone transmission or transmission failure can be avoided which improves robustness and autonomy of the UAV steering and overall flight control.

Secondly, we show that open-source hardware projects can be used to build a low-cost high-performance indoor quadcopter for robotic research.

Thirdly, since most of the necessary hardware for autonomous flight is already included in a state-of-the-art smartphone, hardware complexity as well as costs can be minimized. Additionally, the entire system that provides autonomous flight can be quickly migrated to more powerful mobile devices with low effort which increases ease of use and flexibility compared to competing approaches.

2. RELATED WORK

Simultaneous localization and mapping (SLAM) in an unknown environment and without any human interaction is an active research area in mobile robotics [12]. In related work, autonomous localization of the UAV can be distinguished into two different approaches:

Outside-In-localization and *Inside-Out*-localization. The first uses fixed sensors, such as cameras, to estimate the 3D position of an object, e.g. the UAV. Therefore, this system needs to extract the object from the scene. For *Inside-Out*-localization, the sensor(s) such as cameras, laser range finder and inertial measurement units (IMU) are mounted onto the UAV and estimate the UAV's 3D position by observing the surrounding environment. Furthermore, related work for autonomous flight can be separated into *off-board* and *on-board* approaches. *Off-board* solutions require additional stationary hardware (ground station) to enable UAV localization and autonomous flight. All data processing for position estimation and steering control generation is performed by the ground station; this data is sent via a wireless connection to the UAV. Since all data is processed on the ground station, there is no limitation in size and weight of the hardware components. Furthermore, complex SLAM approaches that are based on dense visual data can be processed in real-time by employing powerful state-of-the-art CPUs und GPUs for workstations. However, the operational range is limited to the particular transmission medium, since there has to be a permanent connection between the UAV and the ground station to ensure stable flight. In contrast, *on-board* solutions process all algorithms for autonomous localization and flight on the UAV itself, hence they are independent of any ground station or reliable wireless transmission. However, weight, size and power consumption of the additional hardware components have to be taken into account at the design of the UAV.

2.1 Off-Board Outside-In Localization

The approach in [2] uses active, light-emitting markers that are mounted on the UAV. It demonstrates a low-cost, transportable solution using a notebook and two off-the-shelf web cameras as ground station. By tracking the UAV with both cameras, its current 3D position is continuously estimated; thereby, steering controls for an autonomous flight are generated and are sent via wireless connection to the UAV. A similar concept is demonstrated in [15]. Instead of tracking active markers, the CAD model of the UAV is used for generating visual contour features. Since the 3D position estimation is calculated on an external ground station, both approaches cannot be considered as truly autonomous [21]. However, they are helpful for generating ground truth data for evaluating on-board solutions.

2.2 Inside-Out Localization

Various research projects exist for Inside-Out localization of robots. While ground robots usually are localized through odometry estimates by evaluating the relative movement of the wheels, other localization methods need to be considered for flying platforms. Such a localization for aerial robots is not only important for autonomous flight including mapping and navigation, but also plays a major role for correcting lateral drifts. Related work mostly implements *Inside-Out* localization based on the following approaches: (1) GPS, (2) Vision, (3) Sensor or (4) a combination of these techniques. GPS signals provide reliable position estimation in outdoor environments. However, for indoor purposes and GPS-denied areas, other approaches need to be examined.

2.2.1 Off-Board Solutions

In [23] the autors provide a marker-based approach for autonomous hovering over a pre-defined position. Therefore, a monocular camera is mounted on the UAV for observing the ground. Camera and IMU data are sent to an off-board PC where the UAV's position and attitude (pose) are estimated. Based on the estimated pose, a steering command is sent to the UAV via XBee for correcting flight deviations.

2.2.2 On-Board Solutions

In [20], a UAV is demonstrated that is able to map an entire multi-floor building in real-time without the need of any ground station. For 3D position estimation, a laser range finder retrofitted by mirrors is employed. To sim-

plify the position estimation algorithm, they make use of the assumption that every target the laser aims at is a planar surface. The approach of [11] provides an open-source system, which calculates position estimates on a linux-based embedded platform using a laser range finder. During experiments, altitude and yaw control were executed on-board, whereas localization, SLAM and altitude estimation were performend on an off-board portable workstation. Although laser range finders serve very accurate information for position and yaw estimation, they are expensive, heavy and consume a lot of power. Thus, they are not sufficient to meet our project's objective to provide a low-cost as well as low-weight flight platform. [17] provides an open-source, vision based platform, called PIXHAWK. It allows high-speed on-board image processing with two stereo cameras. All components, especially the UAV hardware, are self designed. With four cameras the system weights $1000 - 1200g$ and allows a continuous flight time of approximately $15min$. In [13], the authors describe an on-board SLAM system for unstructured environments with Microsofts's Kinect RGB-D camera. The main task is to generate collision-free trajectories. Within this project, the embedded system PIXHAWK is used as base processing hardware.

Stereo vision loses its effectiveness when extracting features at large distances and with small baselines [1, 21]. Since only smaller baselines are possible on a UAV while requiring to extract features from the camera images also in larger distances, localization based on monocular vision might be a better choice. Furthermore, weight reduction can be achieved as well. [1] demonstrates 3D position estimation by fusing IMU data with a monocular camera and a barometer. The system does not require additional preconditioning of the environment, e.g. attaching visual markers into the scene. Absolute position estimates are obtained by the visual SLAM framework of Klein and Murray [14]. All implementations are based on ROS (Robot Operation System) [22], making the work reusable for the community. [7] presents a position stabilization framework by using ultrasonic sensor nodes, whereas [16] demonstrates a stabilization approach by using an optical flow sensor conventionally used for desktop mice. The approach of [14] is also employed on the work of [21]. This approach presents a SLAM and stabilization solution for quadcopters using monocular vision without the need of pre-conditioning the environment.

All above projects except [17] use the commercial hardware platform from Acsending Technologies [5] as base UAV. This increases costs and reduces flexibility. Furthermore, all of the mentioned approaches [1, 20, 11, 13, 17, 21] use different kinds of embedded on-board systems for data processing. However, these are mostly designed for a single use case and the underlying hardware can hardly be extended nor easily replaced with emerging and more powerful modules. [8] provides an alternative solution. They use a customary mobile device (Nokia N95) as on-board processing hardware for executing vision-based localization algorithms. This is the most comparable approach to our work, however, the localization and mapping does not occur simultaneously but is split into two phases - exploration and localization. Furthermore, a valid GPS-signal is required for localization during the exploration phase.

In this paper, we demonstrate the potential mobile computing on an off-the-shelf smartphone that acts as core on-board device for executing simultaneous localization and mapping tasks in GPS-denied environments. Furthermore, we aim on flexibility, cost efficiency and re-usability of all required components by applying open source hard- and software. Since the required components – camera, sensors, communication technologies, and the processing unit – for autonomous localization, mapping and navigation are integrated in a smartphone, weight and monetary costs are minimized and migration to more powerful hardware with little effort is ensured.

3. METHODOLOGY

The overall setup is depicted in Figure 1. The hardware design consists of three components: (1) the smartphone as on-board core processing unit to enable autonomous flight, (2) the on-board flight attitude control comprising a microcontroller that interfaces the UAV's hardware with the smartphone and (3) an optional portable workstation for monitoring all flight data during run-time.

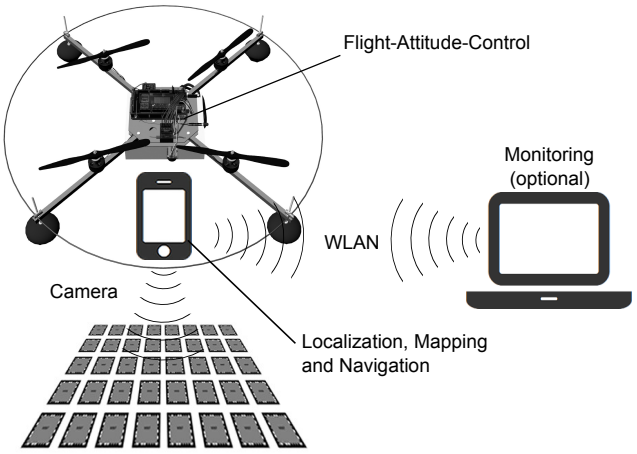

Figure 1: Setup overview.

The smartphone is attached to the bottom of the UAV with the camera pointing down to the ground. By tracking natural features on the ground, the UAV is able to perform simultaneous localization and mapping. Based on the runtime generated map of the environment, the UAV generates navigation commands to explore the unknown area. The monitoring station, as illustrated in Figure 1, is not required for autonomous flight but allows for analyzing all parameters during flight. Therefore, the smartphone communicates via WiFi with the monitoring station. For safety reasons, the UAV is equipped with a receiver for remote control by a human.

3.1 Hardware System

To guarantee a reliable autonomous flight, the UAV hardware has to be able to stabilize itself during every phase of flight. To measure the UAV's attitude and hence be able to generate balancing controls for stabilization, a 9-degree-of-freedom IMU is used. The IMU comprises Accelerometer, Gyroscope and Magnetometer. To get reliable orientation data, all sensor measurements are fused to compensate measuring errors. According to the difference between the current attitude and the attitude needed to balance the UAV, motor-commands are generated for correction.

For sensor fusion and the UAV attitude correction, we use the open source project Arduino Mega ADK Board [4]. This board acts as the main interface between power supply, motors, sensors and smartphone. To connect all hardware components – battery, motors and sensors – to the Arduino, a shield by the open source project AeroQuad [3] is used. Via USB cable, the smartphone is connected to the Arduino Mega ADK board. In addition to the IMU data, we attached a sonar range finder to the UAV to determine the current flight level (height) to allow for autonomous and secure lift-off, landing and pre-defined flight level. All hardware components of the UAV are depicted in Figure 2.

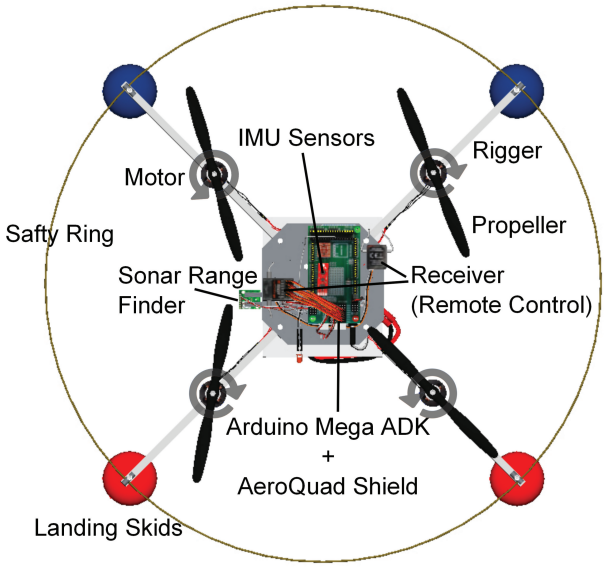

Figure 2: All hardware components of the UAV.

3.1.1 *Frame Design*

The size of a UAV frame construction is specified by the number of motors and the size of the propellers. If more powerful motors and larger propellers are used, more payload can be carried. If too much payload is applied, the UAV's flight stability is negatively affected. To deploy the UAV in an indoor environment, its size must be sufficient to be able to navigate through narrow areas such as doors, hallways and windows. Hence, our frame design is a low-weight solution suitable for indoor flight. Therefore, we designed a UAV with four motors (quadcopter), each equipped with 8" propellers. All four motors were tightly mounted on the same horizontal level. To navigate the quadcopter, the rotation speed of every single motor can be changed individually. To avoid spinning of the UAV around its pitch axis, two opposing motors have to spin in clockwise, the other two in counter-clockwise direction.

3.2 Software Framework

The software system consists of three modules that are executed on the three hardware devices:

- Flight Attitude Control (Arduino Microcontroller [4])
- Autopilot (Android smartphone)
- Optional: Monitoring (PC/Notebook)

Figure 3 shows the communication structure of the modules. The Flight Attitude Control (FAC) is responsible for two very basic UAV abilities, to fly and to balance the UAV during flight. Furthermore, it acts as an interface between UAV hardware (motor, sensors) and the smartphone. The FAC receives data from the IMU and, in case of user intervention as well from the remote control. By evaluating the received data, it generates adequate engine speeds for the motors. We extended the employed AeroQuad shield software with a communication protocol to receive steer commands from the Autopilot (AP) and for sending flight data status updates to the monitoring station.

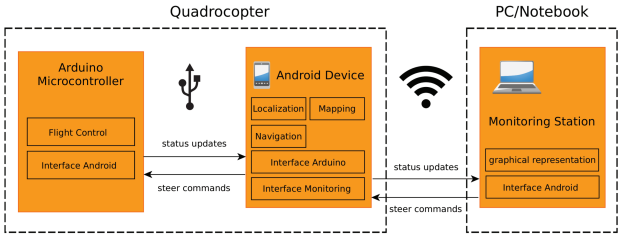

Figure 3: Software modules of the proposed setup.

The monitoring station observes the flight and can be used for UAV parameter configuration before lift-off. Furthermore, it provides visualization of the mapping process that is performed on run-time. Optionally, the autonomous flight can be intervened with manual keyboard commands.

The AP serves as communication center between FAC and monitoring. Furthermore, it is responsible for all three core operations to enable autonomous flight - localization, mapping and navigation that are described in detail in the following sections.

3.2.1 *Localization*

Our approach performs SLAM in a 2D environment. Currently, we use a set of known 2D markers that are attached in random order to the ground. These markers are continuously tracked with the smartphone's camera by using *Vuforia* [19]. Using the calculated view-port coordinates, we perform localization and mapping of the unknown environment by storing the detected markers and their adjacencies. To guarantee correct attitude localization of the UAV as well as linkage between adjacent markers, the current UAV's yaw rotation has to be taken into account. Thus, the yaw rotation φ is extracted from the tracking information and the view-port coordinates $C_{quad}(x, y, z)$ are rotated by φ, resulting in the transformed coordinates $C'_{quad}(x', y', z')$ as described in Equation 1.

$$\begin{aligned} x' &= x\cos\varphi + y\sin\varphi \\ y' &= -x\sin\varphi + y\cos\varphi \\ z' &= z \end{aligned} \tag{1}$$

3.2.2 *Mapping*

During flight, a 2-dimensional map of detected markers within the unknown environment is incrementally built. If a minimum of two markers are tracked at the same time, adjacencies can be established between them. According to C'_{quad}, adjacent markers can be connected in the north, east, south or west, as depicted in Figure 4.

Thereby, an internal graph of the explored 2D area is incrementally created. However, falsely detected markers must be carefully handled to avoid errors in the map. With increasing tracking distance between camera and markers, false marker detection increases as well. Thus, we introduce a quality value for the distance which is encapsulated in the z-coordinate of the currently tracked marker. The higher the distance, the lower the quality of a detected adjacency. With this quality level for new adjacencies, a robust linkage between neighboring markers can be ensured, since detected adjacencies with higher quality overwrite existing ones with lower quality.

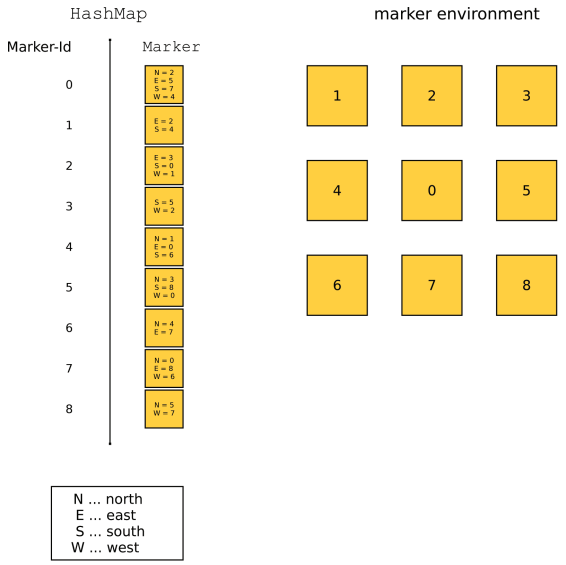

Figure 4: Storing and Linking of Identified Markers.

To build the 2D graph, we employ a combination of a linked list and a key/value hash map data structure. As key we set the marker ID and store the object containing the adjacency information as value. For localization, mapping and navigation, the created graph is translated into a 2-dimensional grid structure to obtain a global view of the 2D marker environment. Therefore, a recursive algorithm analyzes all objects and determines the corresponding 2D grid position based on the stored direction (north, east, south, west).

3.2.3 Exploration

The exploration algorithm provides an efficient mapping process by searching for non-mapped areas within a defined time interval. Therefore, it starts a navigation task (Section 3.2.4) at an already detected and mapped marker. Thus, the unknown area between the UAV's original position and the navigation target are mapped with the described mapping algorithm and all new markers and their adjacencies are added to the global map.

3.2.4 Navigation

As soon as the UAV receives a navigation command - either by the exploration algorithm or by the user via monitoring - it creates a steering command. Based on the global map, this command is calculated by evaluating the distance between the current position P_{curr} and the target position P_{targ}. Next, appropriate steer commands are generated to navigate the quadrocopter into the target direction. Such a push is generated repeatedly in a defined time interval. The strength of the push depends on the distance to the target marker.

3.2.5 Drift Correction

Although the IMU is able to measure deviations from the horizontal UAV position, lateral movement (drift) cannot be detected. Drift occurs since inertial measurements aberration accumulate over time. Furthermore, the current flight level influences the amount of drift since the propellers create air swashes. At low flight level, they can interfere with the UAV. Since our approach requires to robustly detect visual features at smaller distances, low flight level is necessary and thus drift is influenced by air swashes. To robustly detect and compensate drifts, an external static reference point is required [7]. Therefore, we implemented a drift correction algorithm based on visual feature tracking that runs in parallel to the mapping and exploration/navigation task. It evaluates identical markers in two consecutive camera frames to calculate the difference of their viewport coordinates. This information is then sent to the FAC that adjusts the UAV's trims[2].

3.2.6 Edge Detection

To autonomously detect edges of the 2D environment and generate appropriate counter steering commands, we developed the following quality weight based algorithm. If tracking fails for a defined time interval, the UAV's current flight level is evaluated. If the UAV is within an adequate level range that guarantees robust feature detection, it is assumed that the UAV has flown beyond the border of the marker environment. Then, a sufficient counter steering direction is estimated based on the following three informations:

- Polar coordinate p of the last tracked marker
- Global position m of the last tracked marker
- Current calculated drift direction d

Each of the above data is weighted with a quality value $Q_i, i \in \{p, m, d\}$ to describe the estimation quality. To estimate $\varphi \in \{p\}$, we assume a unit circle as polar coordinate system the origin of which lies at the center point of the most recently tracked marker while $Pos_{exit} \in \mathbb{R}^2$ denotes the position of the camera's viewport upon map exit. Then, φ is determined by evaluating the angle of Pos_{exit} in the unit circle. With φ, we can determine the current steering direction d_{steer} of the UAV, based on Equation 2.

$$d_{steer} = \begin{cases} \text{West} & \text{if } \frac{3\pi}{4} < \varphi < \frac{5\pi}{4} \\ \text{South} & \text{if } \frac{5\pi}{4} < \varphi < \frac{7\pi}{4} \\ \text{East} & \text{if } \frac{7\pi}{4} < \varphi < \frac{\pi}{4} \\ \text{North} & \text{if } \frac{\pi}{4} < \varphi < \frac{3\pi}{4} \end{cases} \quad (2)$$

Next, the quality of d_{steer} is calculated based on Equation 3 whereas the highest quality is described with $q(d_{steer}) = 1$. Therefore, the lower and upper bound values of φ, denoted as L and U, are employed.

$$q(d_{steer}) = \frac{L+U}{2} \times \frac{1}{d_{steer}} \quad (3)$$

[2]Trims are used in model flight for initial adjustment of the model aircraft to compensate unequal weight shifts.

To determine the quality of m, we evaluate the distance between the center point of the last tracked marker and the global map. Since this distance depends on the size of the autonomously mapped area, its current size must be taken into account as well to provide robust quality estimates. At the beginning of an autonomous flight, the size of the map will be rather small and hence the resulting quality of m will be low.

For d the quality is described by the strength of the current calculated drift direction. Next, all three parameters are weighted with a value $W_i, i \in \{p, m, d\}$ describing the overall reliability of the data (p, m, d). m is weighted stronger, whereas d and p have less influence. Based on the result with the highest value, a counter steering direction on the x-axis $\vec{Ex}$ as well as on the y-axis $\vec{Ey}$ is calculated. The approach is formally described in Equation 4.

$$\vec{Ex_i} = \begin{pmatrix} DecisionEast \\ DecisionWest \end{pmatrix}, i \in \{p, m, d\} \quad (4)$$

$$\vec{Ey_i} = \begin{pmatrix} DecisionNorth \\ DecisionSouth \end{pmatrix}, i \in \{p, m, d\}$$

$$\text{Steering direction X} = \left\| \sum_{i=0}^{3} \vec{Ex_i} * Q_i * W_i \right\|_{max}$$

$$\text{Steering direction Y} = \left\| \sum_{i=0}^{3} \vec{Ey_i} * Q_i * W_i \right\|_{max}$$

3.2.7 State Cycle of Auto Pilot

The described autonomous flight algorithms are performed by the AP unit depending on the current flight state of the UAV. The overall state diagram is illustrated in Figure 5 while the AP can be in one of the following states:

- Lift-Off
- Exploration
- Navigation
- Edge detection and counter steering
- Land

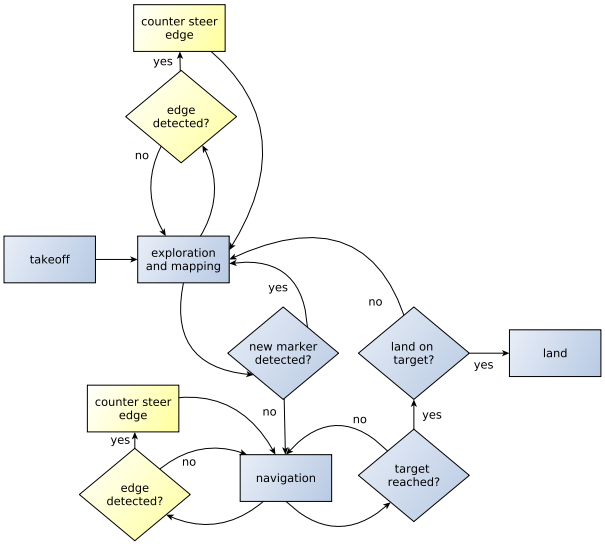

Figure 5: States of the AP unit.

To provide an autonomous flight from lift-off until landing in an unknown environment, the UAV performs the following operations in each state. After *Lift-off*, the quadcopter automatically holds a predefined flight level in which robust target tracking can be ensured. Then, its state changes to *Exploration* and global mapping is performed. If no unknown markers are detected within a defined period of time, the AP unit determines unmapped areas in the global map and defines it as navigation target. The state changes to *Navigation*. Once the navigation target is reached, the status changes back to *Exploration*. If the UAV leaves the 2D environment, the state changes to *Edge Detection* and an appropriate counter steering command is generated. If the AP is pre-configurated to land onto a defined marker, a navigation command is performed. As soon as the target marker is detected, the UAV changes to the final state *Land* and performs an autonomous operation to stop the flight.

4. IMPLEMENTATION

4.1 Hardware

The underlying hardware design of the UAV is inspired by the constructions of [3, 6]. Figure 6 illustrates the current design of the UAV.

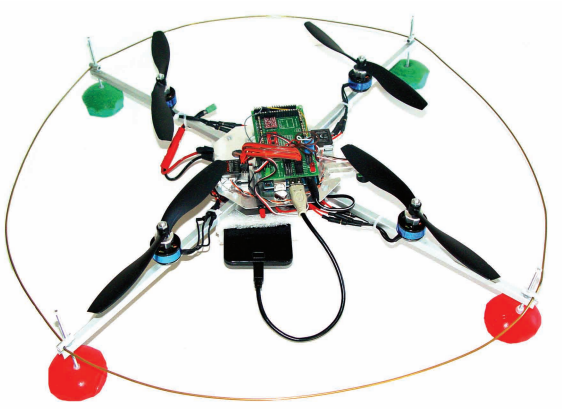

Figure 6: Hardware prototype, smartphone pulled out from mount for visualization.

The entire frame is made of Aluminum. The $580mm$ long, u-profiled riggers are mounted between two $125mm \times 125mm \times 1mm$ sized plates, as illustrated in Figure 2. The Arduino Mega ADK Board with attached AeroQuad Shield and IMU is mounted on the top of the upper plate. The brushless motors can spin with 1088rpm/V[3] and are mounted $160mm$ from to the UAV's center of gravity. The motors are connected via ESCs[4] to the Arduino Board. An ESC is necessary to transform the DC-control signal to a phase shifted AC-signal to drive the brushless motors. At the end of each rigger, a skid is mounted to minimize the bounce during the landing processes. For safety reason, we attached a safety ring around the UAV to protect the complete UAV's hardware in case of collisions. As power supply a $2100mAh$ LiPo[5] battery is mounted on the bottom of the UAV. It supplies

[3]Revolutions Per Minute/Volt
[4]Electronic Speed Controllers
[5]Lithium Polymer

all hardware components with $11.1V$. While the four motors are directly connected via the ESCs to the power source (with $11.1V$), all other components are indirectly supplied with $5V$ over the Arduiono board.

The developed UAV prototype generates thrust to carry $1410g$, while having a net weight of $840g$. With this weight and size, the prototype is able to operate $10-15min$ in the air.

4.2 Software

The autonomous flight processing pipeline of the AP unit was fully implemented with the Android SDK [10] in Java. Run-time critical operations for localization, mapping and navigation were developed in C/C++ using the Android NDK [9] and are interfaced via via JNI[6] with the Java components of the pipeline. For planar marker tracking, we integrated Vuforia [19].

All modules of the FAC are developed with the Arduino IDE. To provide communication between FAC and AP, the Android Open Accessory protocol (AOA) was implemented. Since Android devices have low power output, the Arduino microcontroller acts as USB-Host and powers the bus, whereas the Android smartphone acts as USB-Accessory.

5. EVALUATION

5.1 Test Platform & Environment

We tested our system with a Samsung Galaxy S2 as on-board processing unit, equipped with a Cortex A9 Dual Core processor and an 8 mega pixel camera. As operating system for the tests, Android Version 4.0.4 Ice Cream Sandwich (API level 15) was used. For flight monitoring and parameter analyzing during run-time, the monitoring station was connected over WiFi with the AP unit. In Figure 7, the indoor test environment is depicted.

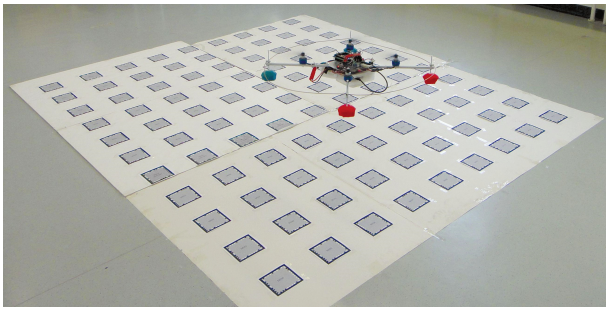

Figure 7: The indoor test environment.

The test area consists of 99 unique identifiable markers where each has a size of $98mm \times 98mm$ and is placed in a distance of $98mm$ to each other. This results in a total size of $1.86m \times 2.25m$ from the test area that was installed in a $4m \times 4m$ room. For the given marker size, the empirically evaluated optimal tracking distances ranged from $100 - 800mm$. With the given test environment, we were able to test flight abilities in very small areas to accurately evaluate localization, navigation drift compensation and landing precision.

[6]Java Native Interface

5.2 Test Cases

To examine all functionalities of the setup, we performed the following tests to evaluate:

1. Flight stability by testing drift correction and holding of flight level.
2. Localization and navigation by examining the exploration of the unknown test environment, the mapping of detected visual markers as well as the quality of edge detection by evaluating the steer command generation and execution to hold the UAV within the test environment.
3. Performance measurements to determine the capabilities of the smartphone as on-board processing unit.

The results of each test case are described in the following section.

5.3 Results

5.3.1 *Flight Stability*

Stable flight behavior is the most important requirement to provide robust (autonomous) flight. Flight stability is negatively influenced by following factors:

1. Declining battery-voltage can cause lower rotational spin of the motors, resulting in inaccurate attitude and flight level.
2. Drifts, as described in Section 3.2.5, must be compensated to prevent uncontrolled lateral movements of the UAV.
3. Increasing heat of the hardware components can result in larger sensor aberrations and hence in inaccurate attitude and flight level.

As described in Section 3.2.5, drifts can be compensated by observing a static point in the scene, e.g. a visual marker and analyzing its position over consecutive camera frames. If the marker translation differs from the current steering direction, a drift correction is initiated. Loss of flight level is detected with the sonar range finder and can be corrected by increasing or decreasing the rotational speed of the motors.

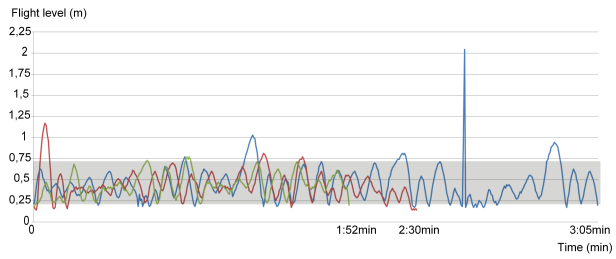

Figure 8: Flight levels of the UAV.

To evaluate the algorithmic drift correction from Section 3.2.5, we placed the UAV at the center of the test environment and performed a lift-off to a flight level of $500mm$, four times with enabled and four times with disabled drift correction. In theory, the UAV is holding its position and hover over the middle of the map if no steering commands

are performed and no drifts occur. Without drift correction, the drift influenced the UAV's position after an average time of $7s$ by $100cm$, resulting in leaving the test environment. With drift-correction, drifts were reliably detected and compensated and the UAV's position could be correctly stabilized up to an average time of $38s$.

Flying at a defined level is important to ensure reliable tracking and thus localization and mapping. With the employed visual markers, robust tracking can be performed between a height of $200 - 700mm$. To constantly hold the UAV within the optimal tracking range, a permanent adaptation of the rotational motor speed related to the measurements of the sonar range finder is necessary.

Figure 8 illustrates the flight level during three test flights with enabled autonomous exploration and navigation. The data indicates that a few times the UAV was below or above the required level but was always able to correct its level to fly in the intended optimal tracking range.

5.3.2 *Exploration & Navigation*

To evaluate the exploration method that comprises localization, mapping and navigation (Section 3.2.3), we measured the time for exploration in correlation to the number of detected and mapped visual markers. Figure 9 illustrates the exploration and mapping process of the unknown test environment during several flights. As depicted, on average 80% of the test environment was successfully explored and mapped within $2min$. These results show the robustness of the applied mapping and exploration algorithm and furthermore reveal the influence of air time on flight stability.

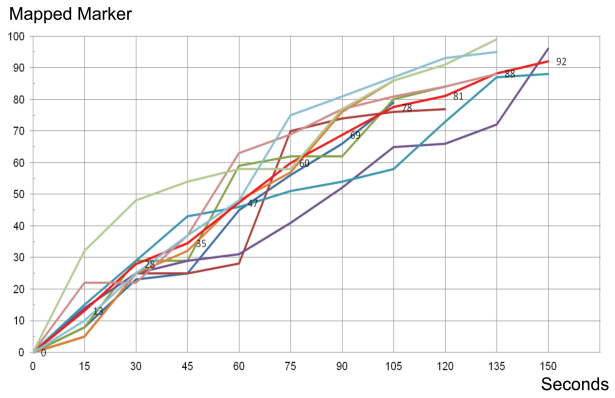

Figure 9: Exploration process.

To test the quality of generated and executed steering commands for autonomous navigation, we measured the time period until the UAV navigates and detects an already mapped position in the test environment. Therefore, the UAV had to perform a navigation task from one corner of the test environment to the diagonal opposite one. Thereby, we ensured a complex navigation task including the largest possible distance in the map as well as steering comprising forward and lateral movements. On average, the target position was reached in $7.8s$.

Furthermore, we determined the accuracy of the landing position if a target marker has been defined. As described in Section 5.3.1, overall flight stability suffers from several factors. Nevertheless, during tests the UAV was able to navigate to a defined target area with a size of $98 \times 98mm$. An average deviation in landing position accuracy of $200mm$ was measured. Deviation was mostly influenced by the current drift of the UAV.

For both test cases, accurate and reliable self-localization as well as edge detection (Section 3.2.6) is crucial. In case of edge detection, a steer-back command needs to be generated for an autonomous flight back into the test environment. The edge detection was tested in 50 runs. Therefore, the UAV was artificially navigated out of the test environment via autonomously or manually generated steer commands or due to drifts. If no visual marker tracking occurred for more than $750ms$, the edge detection algorithm generated a steer command to navigate back into the test environment. The tests revealed that the edge detection works very robustly, in every test case the algorithm detected the correct steer direction to navigate the UAV back to the environment and the corresponding steer command was generated. However, we could identify problems with the generated steer command in 6% of tested flights when a drift diagonal to the current steering direction caused the map exit at one of its corners. Since this drift could not be reliably compensated due to the missing static reference point, the generated counter steering command was leading to an lateral movement only along the edge of the map and did not result in navigating back over the test environment. In cases of drifts opposite to the current steering direction during map exit, the counter steering command outperformed the drift and resulted in successful back-navigation.

5.3.3 *Performance*

We analyzed the performance of the smartphone by evaluating the latency of tracking, mapping, edge detection and navigation command generation. These performance measures are relevant since latency influences the update rate during tracking and hence accuracy of mapping.

All visual markers in a camera frame are identified in $< 1ms$. The integration of a new marker in the global map requires $< 5ms$ while a delay occurs if an edge is detected or a new navigation command for exploration is required. In case of edge detection, tests with 30 samples indicate an average duration of $4ms$ and a maximum duration of $10ms$ to calculate a new steer command. To create a navigation command in exploration mode, the global map is analyzed to identify unknown areas. The more markers are already mapped, the more time the graph analysis requires. An average of $7.5ms$ and a maximum duration of $20ms$ are required to identify unknown target areas and to generate the corresponding steer command.

Summarizing, to process all tasks for autonomous flight – localization, mapping and navigation command generation – a maximum of $25ms$ is required. To compensate for unexpected delays, we added 20% of the real processing time to obtain the expected framerate of the system. Thereby, the final prototype is appointed to repeat the flight pipeline every $30ms$, resulting in an interactive flight framerate of $33fps$[7].

5.4 Discussion

The results of flight stability, exploration and navigation as well as overall system performance show the capabilities of the demonstrated approach to provide autonomous

[7]Frames per Second

flight of a UAV in GPS-denied environments. The employed off-the-shelf smartphone is able to perform all calculations for autonomous flight with a sufficient framerate of 33fps. Thereby, all necessary adoptions to the current flight situation can be performed with interactive frame rates to guarantee accurate localization and mapping as well as collision free flight within a defined environment.

The developed prototype is able to start autonomously from any point within the test environment and hold a defined flight level during exploration and navigation. Borders of the test environment are robustly detected to ensure an autonomous flight within the mappable area. During air time, an internal map is incrementally generated and unknown areas are reliably detected to guarantee the exploration of the entire environment. Furthermore, steer commands and parameters for flight optimization can be manually sent from the monitoring station at any time during flight. In case of a user-defined target, the UAV is able to land autonomously at the specified marker.

With the size of $580mm$ in diameter, our system is able to accurately operate within an area of only $1.86m \times 2.25m$ and navigate to target areas of $200mm \times 200mm$. Therefore, very smooth steering commands and drift corrections are required that the proposed processing pipeline is capable of generating.

Compared to competing approaches, our work does not require embedded hardware platforms. These are expensive and a straightforward hardware extension or replacement with newer models is a tedious task. In contrast, a mobile device such as a smartphone running Android can be easily replaced with emerging, more powerful hardware due to downward compatibility. Hence, the on-board processing unit can be quickly replaced without affecting the overall UAV hardware setup. Furthermore, necessary components for autonomous flight such as camera, processor and communication unit are combined into one single device. Thus, weight can be minimized which was a requirement to design a UAV that is small enough to operate in narrow indoor environments. Furthermore, with this approach, costs could be decreased as well. Commercial UAVs are available from €300 - €10.000. The presented prototype only costs €380 not including the mobile device. However, commercial low-cost UAVs are not able to perform autonomous flight, in most cases the embedded hardware is not modular and extra payload can not be carried or mounted.

5.5 Limitations

Currently, our presented approach is limited to planar visual pattern environments and is not yet capable of performing SLAM based on unconstrained natural visual features extracted from the observed environment. However, emerging mobile devices provide more processing power for real-time feature computation, localization and scene mapping, using monocular SLAM or additional sensing devices. The integration of these external imaging sensors, such as Time-Of-Flight or RGB-D cameras is straightforward with our proposed hardware setup since they can be connected over USB to the mobile device.

6. CONCLUSIONS & FUTURE WORK

In this paper, we demonstrated a low-cost and low-weight UAV that is entirely based on open source hard- and software and uses an off-the-shelf smartphone as its core on-board processing unit to allow for autonomous flight in indoor environments with no GPS coverage. All data processing and steering command generation is performed by a smartphone running Android. Thus, no additional ground hardware is necessary for localization, mapping, navigation as well as lift-off and landing. We demonstrated the autonomous flight capabilities in an unknown 2D environment; no prior knowledge about the planar area is required. The entire setup excluding the smartphone costs €380,- and thus can be compared to commercial low-budget products like the parrot AR drone [18]. The autonomous flight pipeline – localization, mapping, exploring, navigation and edge detection – is executed by the smartphone with $33fps$, providing interactive frame rates for real-time flight operations.

In future work, we will optimize the system's latency by exploiting the GPU processing power for heavy parallelization of emerging mobile device generations. Furthermore, we will integrate depth sensing image devices to allow for dense 3D mapping and localization in unconstrained indoor environments. Thereby, we aim on providing a low-cost platform to fulfill real world tasks like search, rescue and measurements. Especially in areas with poor infrastructure but a high number of mobile devices (i.e. in some developing countries) this is a very promising approach with high potential.

7. REFERENCES

[1] M. Achtelik, S. Weiss, and R. Siegwart. Onboard IMU and Monocular Vision based Control for MAVs in Unknown In- and Outdoor Environments. In *IEEE International Conference on Robotics and Automation*, pages 3056–3063, Shanghai, China, 2011.

[2] M. Achtelik, T. Zhang, K. Kuhnlenz, and M. Buss. Visual Tracking and Control of a Quadcopter using a Stereo Camera System and Inertial Sensors. In *International Conference on Mechatronics and Automation*, pages 2863–2869, 2009.

[3] AeroQuad. The Open Source Multicopter. [Online] http://www.aeroquad.com/, Apr. 2013.

[4] Arduino. [Online] http://www.arduino.cc/, Apr. 2013.

[5] Ascending Technologies GmbH. [Online] http://www.asctec.de/, Feb. 2013.

[6] R. Büchi and P. Dauner. *Fascination Quadrocopter: Basics - Electronics - Flight Experience.* Vth-Fachbuch. Verlag f.Technik/Handwerk, 2010.

[7] J. Eckert, R. German, and F. Dressler. An Indoor Localization Framework for Four-Rotor Flying Robots Using Low-Power Sensor Nodes. *IEEE Transactions on Instrumentation and Measurement*, 60(2):336–344, Feb. 2011.

[8] S. Erhard, K. E. Wenzel, and A. Zell. Flyphone: Visual Self-Localisation Using a Mobile Phone as Onboard Image Processor on a Quadrocopter. *Journal of Intelligent and Robotic Systems*, 57(1-4):451–465, Sept. 2009.

[9] Google Inc. Android NDK. [Software] Revision 8e http://developer.android.com/tools/sdk/ndk/, Jan. 2013.

[10] Google Inc. Android SDK. [Software] Version 4.0.3 http://developer.android.com/sdk/, Jan. 2013.

[11] S. Grzonka, G. Grisetti, and W. Burgard. Towards a Navigation System for Autonomous Indoor Flying. In

IEEE International Conference on Robotics and Automation, ICRA '09, pages 2878–2883, 2009.

[12] J. Hertzberg, K. Lingemann, and A. Nüchter. *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. Springer, 2012.

[13] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. In *International Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA, Aug. 2011.

[14] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[15] S. Klose, M. Achtelik, G. Panin, F. Holzapfel, and A. Knoll. Markerless, Vision-assisted Flight Control of a Quadrocopter. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5712–5717, Oct. 2010.

[16] H. Lim, H. Lee, and H. J. Kim. Onboard Flight Control of a Micro Quadrotor using Single Strapdown Optical Flow Sensor. In *International Conference on Intelligent Robots and Systems*, pages 495–500, 2012.

[17] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A System for Autonomous Flight using Onboard Computer Vision. In *IEEE International Conference on Robotics and Automation*, pages 2992–2997, Shanghai, China, May 2011.

[18] Parrot SA. Parrot ARDrone. [Online] http://ardrone2.parrot.com/, Apr. 2013.

[19] Qualcomm Inc. Vuforia SDK. [Software] Version 1.5.9 https://developer.vuforia.com/resources/sdk/android/, Jan. 2013.

[20] S. Shen, N. Michael, and V. Kumar. Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV. In *IEEE International Conference on Robotics and Automation*, pages 20–25, Shanghai, China, May 2011.

[21] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-SLAM-based Navigation for Autonomous Micro Helicopters in GPS-denied Environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[22] Willow Garage Inc. Robot Operating System. [Online] http://www.willowgarage.com/pages/software/ros-platform/, Feb. 2013.

[23] T. Zhang, Y. Kang, M. Achtelik, K. Kuhnlenz, and M. Buss. Autonomous Hovering of a Vision/IMU guided Quadrotor. In *International Conference on Mechatronics and Automation*, pages 2870–2875, 2009.