Development of Flight Control Software Architecture

Razeen Hussain[1] Divya Shah[2] Dipendra Subedi[3]

***Abstract*— Recent years have seen the use of miniature unmanned aerial vehicles (UAVs) become more popular for search & rescue and delivery missions, but tracking and landing on moving targets is still a major hurdle in their usage. This project presents a modular system comprising of a micro UAV and a mobile platform in which the UAV is capable of performing trajectory tracking of the mobile platform and is able to land on it. The AscTec Pelican quadrotor was used as the UAV and the Husqvarna Automower 330X as the mobile platform. Localization is done through OptiTrack, an eight camera high precision motion capture system which streams the position of the quadrotor and the platform in real time. Control architecture is established for the data-exchange between different modules. Three control laws (PD, PID, and static feedback) were tested for the two-dimensional tracking module. Circular, linear and randomly generated trajectories were used for evaluation purposes. The quadrotor was able to successfully track and land on the mobile platform with small errors allowing for safe indoor flights. The static feedback outperformed the other control schemes in terms of steady state errors, convergence times and simplicity of control.**

I. INTRODUCTION

Aerial robotics is a rapidly growing area in the field of robotics with quadrotors playing a central role in its research activities [1]. A quadrotor consists of four independently controlled motors which govern how it flies. The real potential of a quadrotor lies in its civilian and military applications [2]. Due to its hovering capabilities together with high maneuverability, it can fly at low altitudes without collision with the environment making it a popular choice for various tasks such as aerial photography, delivery drones and search & rescue missions.

One of the key challenges faced in the aerial robotics field is the problem of landing the drone on a remote platform. Although significant work has been done with stationary platforms, the prospect of landing on a mobile platform is greatly underdeveloped. The task can be broken down into two subtasks, namely the tracking and the landing.

Most commercial drones rely on GPS as the main position feedback [3] but their accuracy (approximately 3m) [4] is not up to the mark for high precision demanding applications such as the ones discussed above. Although the use of vision based algorithms for localization is becoming popular but the low payload capacities of the micro unmanned aerial vehicles (μUAV) such as quadrotors limit the size of the onboard computer and ultimately the performance of the algorithm suffers [5]. There exists external position feedback systems which provide high accuracy localization but they limit the autonomy of the drone.

The tracking problem has been addressed by many in recent years with the traditional proportional-integral-derivative (PID) controller being a popular choice among the researchers. Bouabdallah et al. [6] demonstrated its use and successfully landed the quadrotor on a 20cmx20cm area. But his approach is only effective when the tracking object has a smooth trajectory. Wenzel et al. [7] applied a similar approach but the success rate for landing on smaller targets was not good. Another popular approach for target tracking is the use of an extended kalman filter (EKF) like the one used by Prevost et al. [8] but this requires the dynamics of the system to be accurately represented in the observer system.

There are various approaches that can be used to perform the landing. Different approaches based on missile guidance laws have been established by Gautam et al. [9] with pure proportional navigation being most effective for landing a drone.

The objective of this project was to develop a flight control software architecture that handles the data-exchange between the onboard computer and the low-level auto-pilot. Once a suitable architecture has been established, it is expanded to accommodate a high-level control module which handles the mobile platform tracking and the quadrotor landing tasks. Robotic operating system (ROS) has been used as the framework. The experimental setup has been done indoors with an external high precision position feedback. Various control algorithms have been implemented and discussed.

Sec. II describes the complete system used for this work. The proposed control software architecture is explained in the Sec. III. Followed next, is the implementation and the control laws in Sec. IV. The Sec. V describes the experimental results and analysis. The conclusions and the future works are mentioned in the Secs. VI and VII respectively.

II. SYSTEM DESCRIPTION

The complete system used for this work comprises of: the quadrotor, `AscTec Pelican`; the motion capture sys-

[1,2,3] The authors are the students of European Masters on Advanced Robotics (EMARO+) with Department of Informatics, Bioengineering, Robotics & Systems Engineering (DIBRIS) at Università degli Studi di Genova; Via All'Opera Pia, 13 - 16145 Genova - Italy.
[1]razeenhussain@hotmail.com
[2]divyashah.2801@gmail.com
[3]star.dipendras@gmail.com

tem, `OptiTrack` and the mobile platform, `Husqvarna Automower 330X`.

A. AscTec-Pelican

The AscTec Pelican quadrotor, shown in Fig. 1, is built by Ascending Technologies©[1] and can be interfaced with ROS.

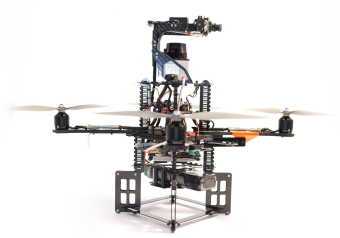

Fig. 1: Asctec Pelican Quadrotor

1) Technical Data: The technical specifications of the AscTec Pelican quadrotor are described in the Table I.

TABLE I: Technical Data

Specification	Description
Onboard computer	Up to 3^{rd} Generation Intel Core i7 Processor
Size	651 x 651 x 188 mm
Max. Total of Weight	1,65 kg
Max. payload	650 g
Flight time including payload	16 min.
Range	4,500 m ASL, 1,000 m AGL
Max. airspeed	16 m/s
Max. climb rate	8 m/s
Max. thrust	36 N
Wireless communication	2,4 GHz XBee link, 10–63 mW, WiFi
Intertial guidance system	AscTec AutoPilot with 1,000 Hz update rate
Flight modes	GPS Mode, Height Mode, Manual Mode

2) Essential Flight System Characteristics:

- The inertial guidance system provides highest precision through advanced sensor components and two ARM7 microprocessors.
- The control unit provides highest flexibility. Latest interfaces simplify the implementation of user's C–code algorithms.
- The Low Level Processor (LLP) ensures a highly stable flight behavior of the flight system. The LLP is the data controller that processes all sensor data and performs the data fusion of all relevant information with an update rate of 1 kHz.
- The High Level Processor (HLP) controls the flight system according to user defined control algorithms.
- Safety Switch function enables to simply switch back into safe mode while testing the control commands and maneuvers, and the proven AscTec AutoPilot takes over.

[1]http://wiki.asctec.de/display/AR/AscTec+Pelican

3) AscTec Mastermind: The AscTec Mastermind is an onboard processor board, which can be carried by the AscTec Pelican. It basically can be used like a ground PC, but it is mounted on the flight system. VGA monitor, USB-keyboard and USB-mouse can be connected for interaction. In comparison to its weight and size, it offers an extremely high processing power, high data rates and a great variety of standard PC interfaces to connect several kind of hardware devices.

The AscTec Mastermind was especially designed to fit mechanically and electronically into the AscTec Pelican. It can be divided into two basic parts, the AscTec Mastermind board and the AscTec Mastermind CPU-option. The AscTec Mastermind Board is identical for each AscTec Mastermind and can hold one CPU-option. It contains all jacks and slots for the diverse interfaces, a display for status information, a power connector for a bridge battery and the power button. A small battery is attached, that powers the system clock. There are several CPU-options available with different performances. Depending on the needs a balance between processor power and weight can be chosen. All are equipped with a specially designed cooler system for the processor.

The AscTec Mastermind comes with a pre–installed Ubuntu Linux on the mSATA SSD. ROS was installed later in the AscTec Mastermind.

4) AscTec AutoPilot: The AscTec AutoPilot is a precise and multi–functional research tool mounted on all AscTec UAVs. This flight control unit contains all necessary sensors to function as an IMU, it also has two onboard ARM7 microprocessors and various communication interfaces. Fig. 2 gives an overview of the AutoPilot.

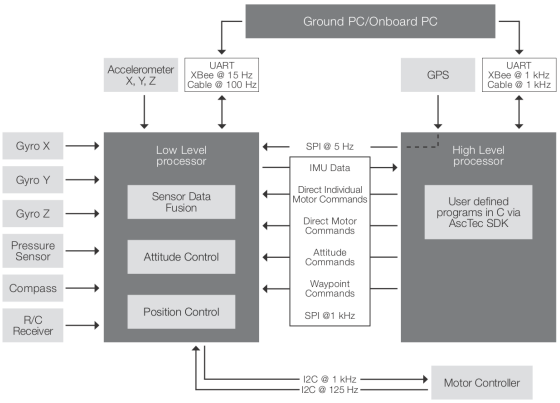

Fig. 2: AscTec AutoPilot Overview

There are several strategies on how the AscTec AutoPilot can be utilized for the research projects. It can directly interpret the R/C data sent by the pilot and the serial data sent from the ground PC. For automatic missions, the user-defined control algorithms can be programmed and directly uploaded to the HLP of the UAV.

If more processing power is required or additional sensors, like cameras or laser scanners need to be connected, an additional onboard processor is installed. Its high performance and fast communication enables the user to handle large amounts of data onboard. An optional WLAN module can be integrated to exchange this data between the additional processor and a ground PC. All gathered data can be used for the control algorithms and the calculated commands can then be sent to the AscTec AutoPilot.

- Low Level Processor (LLP)
  The LLP handles sensor data processing, data fusion as well as the fast and stable attitude control algorithm with an update rate of 1000 Hz. It also executes the position control algorithm, using the onboard magnetometer and GPS module as additional sensor inputs. All AscTec UAVs can be operated by the pilot via a remote control in three different flight modes:
  1) GPS Mode (attitude-, height- and position-control activated)
  2) Height Mode (attitude- and height-control activated)
  3) Manual Mode (attitude-control activated)

  All AscTec UAVs have an emergency mode which is automatically activated if the data link between the UAV and the R/C of the safety pilot is lost (range $>$1 km).
  Sensor data can be retrieved from the LLP via a serial interface using a predefined serial protocol. Furthermore, the serial interface can be used to send attitude commands (pitch angle, roll angle, yaw rate and thrust) or even waypoint commands to the vehicle depending on the flight mode. Due to the size of the data structures, update rates of up to 15 Hz can be accomplished via a wireless serial link (XBee, range $>$1 km). Higher update rates of up to 100 Hz can be achieved using a serial cable connection.

- High Level Processor (HLP)
  The basic concept of the AscTec AutoPilot is to offer easy programmability, and to provide a safety net while testing the control algorithms. The HLP can be allowed to control the flight system by flashing it with the user defined control algorithm. To recover from critical flight situations during the test flights, the pilot can always switch back to the well proven control algorithms on the LLP as a safety backup.
  The two microprocessors communicate with an extremely fast rate of 1000 Hz, hence all sensor data is available to the HLP as well. Control commands can be sent back to the LLP at the same frequency, for example the: rotational speed commands for each individual motor, pitch/roll/yaw/thrust commands, attitude commands or waypoint commands.
  The HLP offers UART, SPI and I2C interfaces as well as simple port I/Os to connect your own devices like additional sensors, servo motors or extension boards. Also custom payloads, like wireless cameras, can be powered by a 5 V or 12 V supply of the AscTec AutoPilot.

B. OptiTrack

OptiTrack is a motion capture system made by NaturalPoint Inc.©[2] It provides premium optical motion capture, and tracking solutions for commercial, industrial, game development and research. The software toolkits are designed for rigid body tracking, full body motion capture and face motion capture. The EMARO+ laboratory at the Università degli Studi di Genova is equipped with eight-camera system capable of tracking quadrotors with high-speed grayscale video at 100 fps. The toolkit used for localization of robots is **Tracking Tools** provided by NaturalPoint Inc.© The position and orientation data of quadrotors streamed from the optitrack cameras can be saved in CSV format or sent over network in real-time.

Motive[3] is a software platform designed to control motion capture systems for various tracking applications. It not only allows the user to calibrate and configure the system, but it also provides interfaces for both capturing and processing of three–dimensional data. The captured data can be both recorded or live-streamed to other pipelines. It obtains three–dimensional information via Reconstruction, which is the process of compiling multiple two–dimensional images of markers to obtain three–dimensional coordinates. Using three–dimensional coordinates from tracked markers, Motive can obtain six Degrees of Freedom (three for position and three for orientation) data for multiple rigid bodies and skeletons, and enable tracking of complex movements in the three–dimensional space.

C. HUSQVARNA AUTOMOWER 330X

HUSQVARNA AUTOMOWER 330X[4] or the Husqvarna Research Platform as shown in the Fig. 3, is ROS enabled durable outdoor robot which has simple ROS interface via provided driver (velocity in, odometry out) and includes Gazebo models for simulated environment.

Fig. 3: Husqvarna Automower 330X

III. CONTROL ARCHITECTURE

The flight control software architecture of the system implemented in this work is shown in the Fig. 4. It consists

[2]http://optitrack.com/
[3]http://optitrack.com/products/motive/
[4]http://www.husqvarna.com/us/products/robotic-lawn-mowers/

of the following modules:

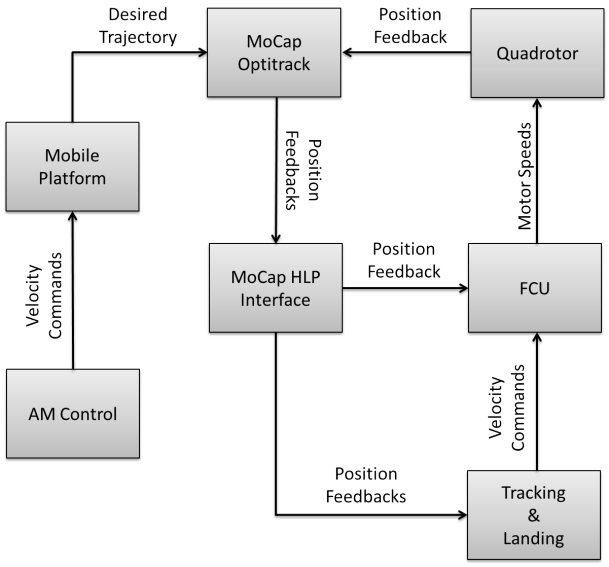

Fig. 4: Flight Control Software Architecture

A. Flight Control Unit

The HLP is flashed with the firmware (asctec_hl_firmware) of *asctec_mav_framework*[5]. The asctec_hl_interface on the mastermind provides ROS interface to communicate with the HLP.

Flight Control Unit (FCU) AscTec Autopilot features a complete Inertial Measurement Unit (IMU) as well as two 32 Bit, 60 MHz ARM-7 micro-controllers used for data fusion and flight control. One of these micro-controllers, the LLP is responsible for the hardware management and IMU sensor data fusion. An attitude and GPS-based position controller is implemented as well on this processor. The LLP is delivered as a black box with defined interfaces to additional components and to the HLP. To operate the quadrotor, only the LLP is necessary. Therefore, the HLP is dedicated for custom code. All relevant and fused IMU data is provided at an update rate of 1 kHz via a high speed serial interface. In particular, this comprises body accelerations, body angular velocities, magnetic compass, height measured by an air pressure sensor and the estimated attitude of the vehicle.

In this work, a position/velocity controller is implemented on the HLP, based on the Motion Capture (MoCap) input from the onboard computer and the inertial data provided by the LLP. On the LLP, the attitude controller is used as the inner loop.

To provide maximum portability of the code and to avoid potential (binary) driver issues, Ubuntu Linux 14.04 is installed on the onboard computer which makes tedious cross compiling unnecessary. Since a couple of different subsystems that need to communicate between each other are running, ROS framework is used as the middleware. This is also used to communicate to the ground station over the WiFi datalink for monitoring and control purposes. The FCU is interfaced via a ROS node communicating over a serial link to the FCU's Higlevel Controller. Software (asctec_hl_firmware) on the HLP is based on a SDK available for the AutoPilot FCU providing all communication routines to the LLP and a basic framework. The HLP communicates with the ROS framework on the onboard computer over a serial datalink and a ROS FCU-node handling the serial communication. This node subscribes to generic ROS pose messages with covariance, in this case from MoCap Framework, and forwards it to the HLP. It also takes velocity commands from the tracking and landing module.

B. MoCap-OptiTrack

The Motion Capture (MoCap)[6] module publishes the 3-D position of the quadrotor and the mobile platform streamed from the optitrack cameras over the network in real-time.

C. MoCap-HLP Interface

This module[7] gets the data from the MoCap and publishes as the FCU/Pose of quadrotor and mobile platform position.

D. AM Control

Automower Control module is used to generate user–trajectories for the mobile platform. Fig. 5 shows the general working schematic.

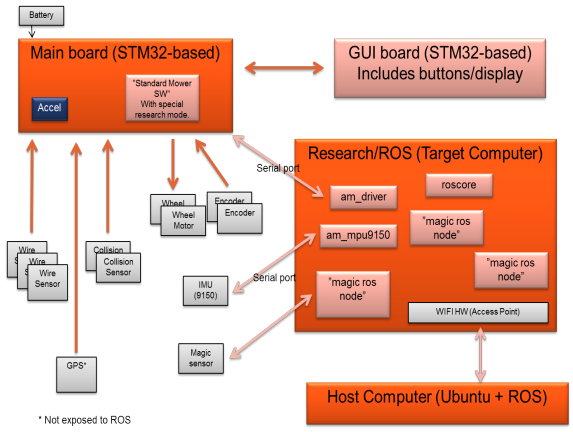

Fig. 5: Husqvarna Automower 330X Working Schematic

E. Tracking & Landing

This module implements a two–dimensional tracking and a landing velocity control. It gets the FCU/Pose of quadrotor and mobile platform position from the MoCap-HLP Interface and calculates the position and velocity error. It gives corresponding velocity commands to the FCU of quadrotor based on the control law. The implementation for the tracking and landing algorithm in this module is explained in the following Sec. IV.

[5]https://github.com/divyashah/QATL/tree/master/asctec_mav_framework

[6]https://github.com/divyashah/QATL/tree/master/mocap_optitrack

[7]https://github.com/divyashah/QATL/tree/master/interface_mocap_hlp

IV. CONTROL LAWS

Based on the control architecture explained in the Sec. III, a common algorithm as shown in Algorithm 1 was implemented to test the performance of different control laws.

Algorithm 1: Psuedo-code for Control Implementation for Tracking & Landing

```
Input: platPos, quadPos, initPos, freq
Output: quadVel
activate Waypoint Server ;
WaypointGoal = initPos ;
rate(freq) ;
while (1) do
    compute errorPos ;
    compute platVel ;
    if quadPos < boundary then
        quadVel = ctrlVel ;
        if quadVel > maxSpeed then
            quadVel = maxSpeed ;
        end
        if errorPos < tolerance then
            Land ;
        else
            quadVel = ctrlVel ;
        end
        if errorPos.z > allowable then
            TakeOff = true ;
        else
            TakeOff = false ;
        end
        sleep(freq) ;
    else
        break ;
    end
end
quadPos = initPos ;
Land ;
```

The Waypoint server from the $asctec\_hl\_interface$ of the $mav\_framework$ is used to send the goal setpoints to the controller. The quadrotor starts from rest on the ground and is first made to ascend to an initial height. Then a velocity control is implemented to track the mobile platform within the capture arena. The positions of the quadrotor and the mobile platform, *quadPos* and *platPos* respectively, inside the arena are provided from the motion capture system. In each iteration the tracking is executed with a given frequency (*freq*) as follows (until it breaks):

- The position errors (*errorPos*) and the mobile platform velocities (*platVel*) along the individual co-ordinate axes are calculated.
- The quadrotor velocities (*quadVel*) are set to the control input velocities.
- If the control input velocities exceed the maximum speed considered (*maxSpeed*) then, the quadrotor velocity is set to the maximum speed itself. This ensures safety by not allowing the quadrotor to fly with speeds greater than permissible.
- If the position error is within the *tolerance* values then, the quadrotor starts descending.
- If the error in z–axis, that is height, exceeds the *allowable* value, quadrotor is made to ascend until it falls back within permissible limits.
- If at any instance the quadrotor reaches the *boundary* limits of the arena while tracking, it is forced to break and return to the initial position (*initPos*) and land. This is also made to ensure the safety during flying.

In this work, three different control laws as presented below, were tested with the above algorithm. The experimental results and performance analysis for each are explained Sec. V.

1) The first control law implemented was based on the classical proportional–derivative control theory.

$$quadVel_{k+1} = K_p{}^* errorPos_k + K_d{}^* errorVel_k \quad (1)$$

where the subscript *k* denotes the instant and the parameters, K_p and K_d are the proportional and derivative gains respectively, that need to be correctly tuned.

2) To the control law stated in (1), an integral term was introduced to obtain the classical Proportional–Integral–Derivative (PID) controller as follows:

$$quadVel_{k+1} = K_p{}^* errorPos_k + K_i{}^* \sum_{i=0}^{k} errorPos_i + K_d{}^* errorVel_k \quad (2)$$

where the parameter K_i is the integral gain which also needs to be tuned correctly in addition to the previous two gains.

3) An alternate control law, based on the modern static feedback control theory was also implemented as follows:

$$quadVel_{k+1} = K_s{}^* errorPos_k + platVel_k \quad (3)$$

In this case, there is only one tuning parameter K_s, that is the static gain and this makes it easier to implement.

V. EXPERIMENTAL RESULTS

The experimental setup was carried out in the EMARO+ lab which has an arena of dimension 4mx4m. Five markers were attached to the Asctec Pelican quadrotor and six markers were attached to the Husqvarna Automower 330X. The placement of the markers was done asymmetrically. The eight camera Optitrack motion capture system as described in Sec. II-B, was used to track the markers and stream the position feedbacks to the processor available on

the quadrotor.

The quadrotor was placed at the origin of the world frame and initially was flown to a height of 1m. The mobile platform, 65cmx50cm in dimension, was given a fixed circular trajectory of radius 0.7m and a linear velocity of 0.2m/s. A 25Hz frequency was used for the position and velocity updates in the tracking module. Using the various control laws described in Sec. IV, the quadrotor was controlled to track the moving target and land on top of it. A tolerance of 5cm was given for the landing module. As a safety feature, the quadrotor motion was restricted within a 3mx3m area and its maximum speed was set at 0.35m/s. Fig. 6 shows the quadrotor in action when it is descending to land on the mobile platform.

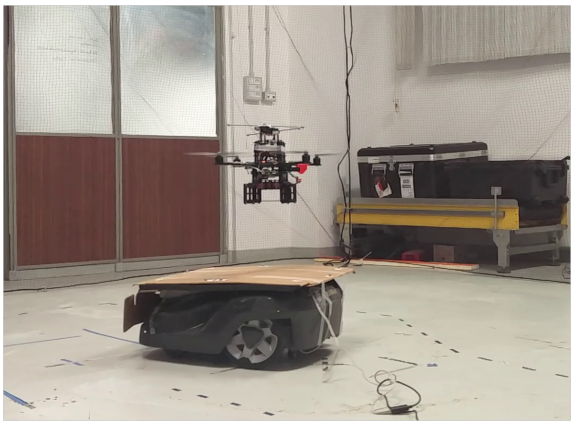

Fig. 6: Quadrotor and Mobile Platform during the Experiment

A. Control Law 1

The classical PD controller as shown in (1) was implemented and the output logs were analyzed. In this case, the gain values were tuned to, $K_p = 3.2$ and $K_d = 2.2$. During flight, it was observed that the quadrotor never landed on the mobile platform, as it can be seen from the Fig. 7. The Figs. 8,9, depicting the position curves, showed an average lag of approximately 9.8cm; this being the steady state error in the system. This error is outside the allowed tolerance limit as seen in Fig. 10, thus the quadrotor never descended to try landing.

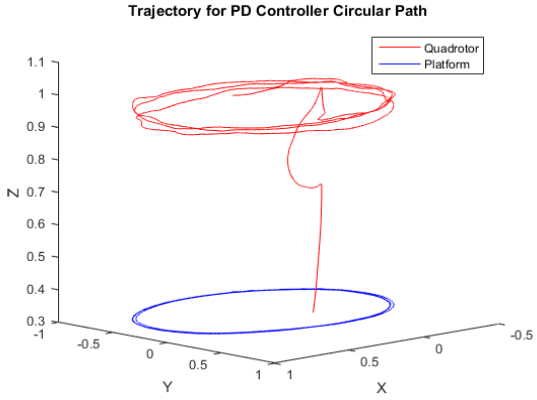

Fig. 7: 3-D Trajectory Plot for PD Control

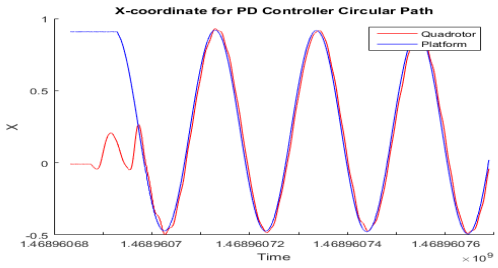

Fig. 8: X-Coordinate Plot for PD Control

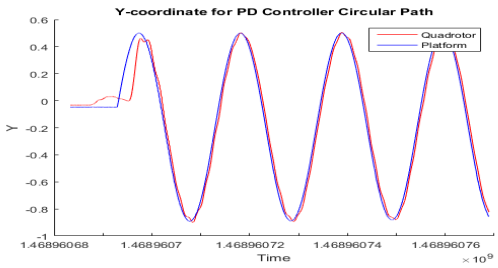

Fig. 9: Y-Coordinate Plot for PD Control

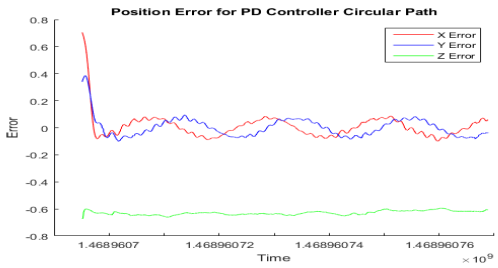

Fig. 10: Position Error Plot for PD Control

B. Control Law 2

A PID controller as shown in (2) was implemented and its gains tuned. The tuning was done using trial and error. The performance was evaluated with the gains tuned to the following values: $K_p = 3.2$, $K_d = 2.2$ and $K_i = 0.04$. It can be observed from the Fig. 11, that the quadrotor, now with the inclusion of the integral term, was able to land. This was due to the fact that the mean steady state error, as shown in the Fig. 14, was 3.8cm which is within the specified tolerance. The x and y co-ordinate graphs are also shown for the same in Figs. 12 and 13.

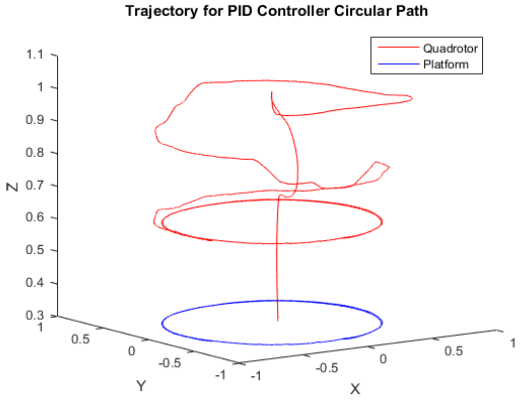

Fig. 11: 3-D Trajectory Plot for PID Control

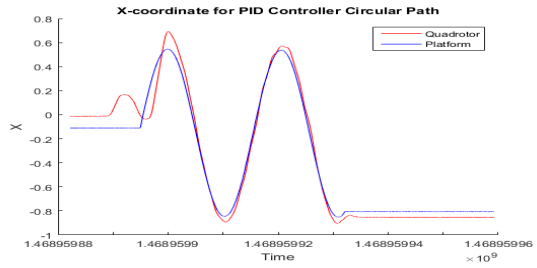

Fig. 12: X-Coordinate Plot for PID Control

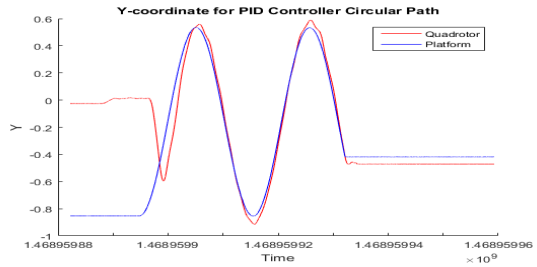

Fig. 13: Y-Coordinate Plot for PID Control

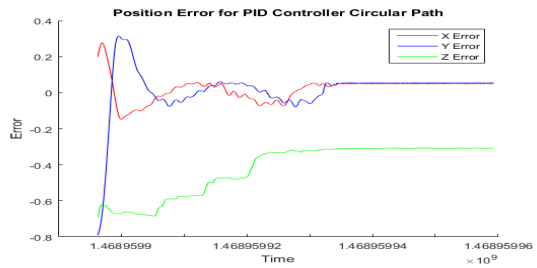

Fig. 14: Position Error Plot for PID Control

C. Control Law 3

The static feedback control scheme, shown in (3) was also tested. There was only one gain to be tuned and was set to $K_s = 1.4$. The output logs depicted in Figs. 15, 16 and 17, showed a faster convergence and an average steady state error as shown in Fig. 18, of 2.6cm. This control scheme was effective only because the tracking was done only for the x and y coordinates. If the inclination of the mobile platform had also been taken into consideration then this control law would have failed and a dynamic feedback controller would have to be implemented.

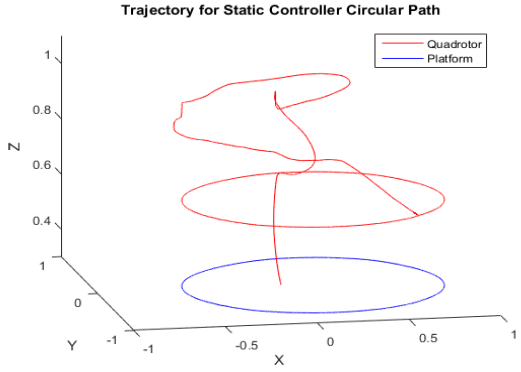

Fig. 15: 3-D Trajectory Plot for Static Control

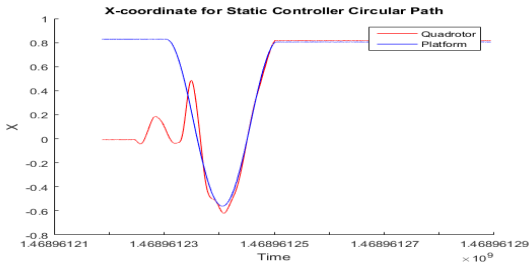

Fig. 16: X-Coordinate Plot for Static Control

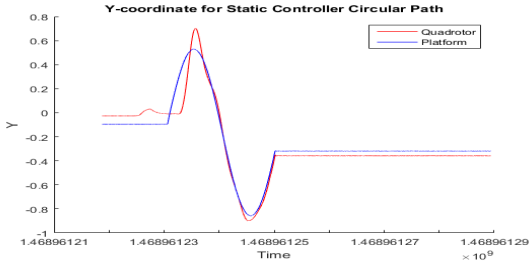

Fig. 17: Y-Coordinate Plot for Static Control

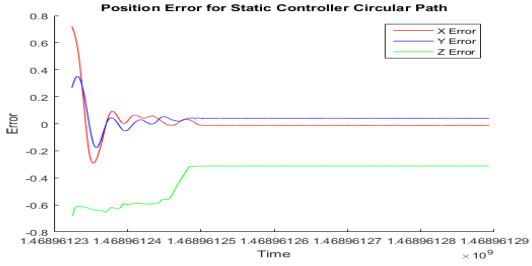

Fig. 18: Position Error Plot for Static Control

D. General Observations

- In all the control laws, it can be seen that the error in height does not go to zero but settles at 0.31m. This is the height of the quadrotor.
- When the gains were high in all the control algorithms, the quadrotor demonstrated an unstable behavior.
- The mobile platform had a slightly vibratory motion and its top surface was uneven. This sometimes caused the quadrotor to displace itself after successfully landing on the target.
- The control laws were also tested with linear and randomly generated trajectories. The results were similar to the ones in the circular trajectory with the static feedback controller being the most effective. Figs. 19, 20 and 21 represent the co-ordinate and error plots with linear trajectory and Figs. 22, 23 and 24 represent the same with randomly generated trajectory.

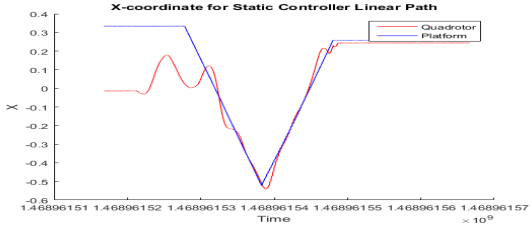

Fig. 19: X-Coord. Plot for Static Control & Linear Trajectory

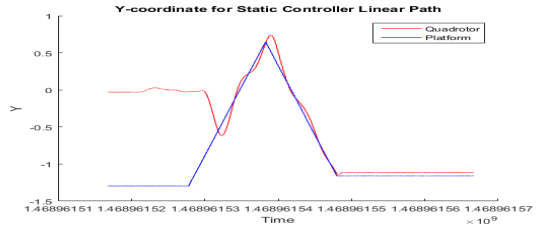

Fig. 20: Y-Coord. Plot for Static Control & Linear Trajectory

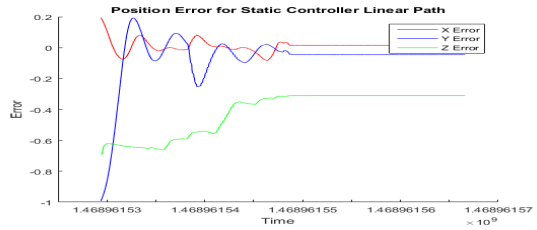

Fig. 21: Position Error Plot Static Control & Linear Trajectory

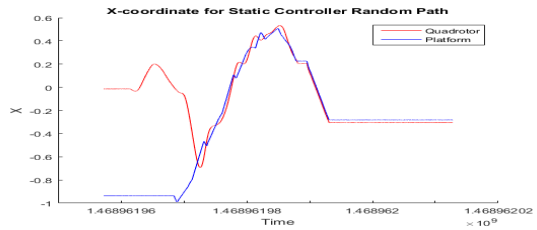

Fig. 22: X-Coord. Plot for Static Control & Random Trajectory

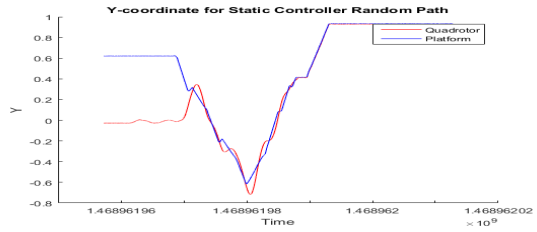

Fig. 23: Y-Coord. Plot for Static Controller with a Random Trajectory

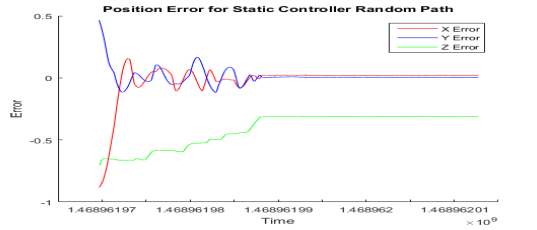

Fig. 24: Position Error Plot for Static Control & Random Trajectory

VI. CONCLUSIONS

In this work, a modular flight control software architecture for a quadrotor has been designed and successfully tested. The FCU module provides the quadrotor with the data exchange between the on-board mastermind and the autopilot. The MoCap HLP Interface module provides a full integration between the OptiTrack MoCap system and the Asctec Pelican quadrotor. The high level Tracking & Landing module provide successful two–dimensional tracking of an object (here, the mobile platform) and eventual landing of the quadrotor. A general algorithm explaining the implementation of the Tracking & Landing module is also provided, with which the different control laws were tested. With easier implementation, acceptable steady state error and faster convergence, the static feedback control law proved to be more effective compared to others. A detailed documentation [8] on how to setup and run the complete system architecture is also provided.

VII. FUTURE WORKS

The tuning of the controller gains in this work, was done using a trial and error approach. For better tuning and ultimately better performance, a different approach such as the PID Autotuning VI, available with the NI LabVIEW software [9], which uses the Ziegler-Nichols method, will be used. Also, the inclination of the landing surface was not taken into account. In future, a dynamic feedback controller will be applied to incorporate the orientation of the platform as well. A vision based localization algorithm will also be implemented to ensure complete autonomy of the quadrotor.

ACKNOWLEDGMENT

The authors thank Andrea Nisticò, Department of Informatics, Bioengineering, Robotics & Systems Engineering (DIBRIS) and Prof. Marco Baglietto, Department of Communication, Computer and System Sciences (DIST) for providing the system and also their support. They also thank the EMARO+ Laboratory at the Università degli Studi di Genova which houses the motion capture system using which the experiments presented in this work were performed.

REFERENCES

[1] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2, p. 4, 2007.

[2] Z. Sarris and S. Atlas, "Survey of uav applications in civil markets," in *IEEE Mediterranean Conference on Control and Automation*, p. 11, 2001.

[3] J. Kim, Y. Jung, D. Lee, and D. H. Shim, "Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pp. 1243–1252, IEEE, 2014.

[4] A. Kleusberg and R. B. Langley, "The limitations of gps," *GPS World*, vol. 1, no. 2, 1990.

[5] S. Majumder and R. Shankar, "Moving object tracking from moving platform," in *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pp. 85–89, IEEE, 2014.

[6] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2451–2456, IEEE, 2004.

[7] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *Journal of intelligent & robotic systems*, vol. 61, no. 1-4, pp. 221–238, 2011.

[8] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle," in *2007 American Control Conference*, pp. 1805–1810, IEEE, 2007.

[9] A. Gautam, P. Sujit, and S. Saripalli, "Application of guidance laws to quadrotor landing," in *Institute of Electrical and Electronics Engineers Inc.*, 2015.

[8] https://github.com/divyashah/QATL

[9] http://www.ni.com/labview/