Ball trajectory estimation using VICON system and Kalman filtering and catching using a quadcopter

Akash Ravindra
Robotics Graduate student
University of Maryland
aravind2@umd.edu

Patrik Dominik Pördi
Robotics Graduate student
University of Maryland
ppordi@umd.edu

Abstract

The following document intends to give insights into our semester project, showcasing each cornerstone that will be accompanied by detailing the related hardware and software components. The main emphases were on developing a state machine that is responsible for detecting a ball and estimating its parabolic trajectory of it using Kalman- filtering, using the estimate a quadcopter is controlled to catch the flying object.

1. Introduction

At the University of Maryland in the course called ENAE788M – Hands-on Autonomous Aerial Robotics, several components of drone missions were covered, such as off-board control, trajectory tracking, state estimation, system identification, and perception were covered. We were encouraged to combine some of the previously mentioned in a semester project. Since the project is used in another class ENPM673 - Perception for Autonomous Robots, our designed project combines the fields of autonomous control and state estimation and highly relies on perception while taking advantage of the given cutting-edge resources.

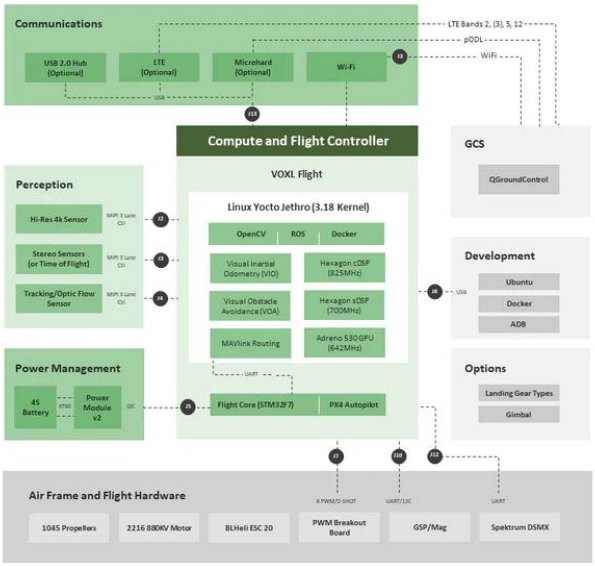
1.1. Inspiration

Our team aimed for a project which is challenging and relies on the efficient cooperation of several sub-components; however, it is fun and remarkable in engineering manners. Special thanks go to Institute for Dynamic Systems and Control (IDSC), ETH Zurich, whose research on "Computationally efficient trajectory generation: quadcopter catching balls" provided us with great inspiration. [1]

2. Project description

2.1. Hardware overview

There were two main hardware components involved in the project. The quadcopter VOXL m500 and the VICON Motion Capture System. Let us begin with the drone since the camera system will be detailed in the next paragraph.



1. Figure: Technical details of VOXL m500

As you can see in the diagram above the quadcopter is equipped with all the different sensors and control units to perform autonomous control using the PX4 autopilot system and MAVROS.

2. Figure: VOXL m500

Since we aimed to perform catching slight modifications were introduced, unnecessary hardware components such as the GPS antenna were dismounted to free up space for the costume designed and printed basket, while some additional reflective markers were also placed on the drone. The used hardware setup is featured below.

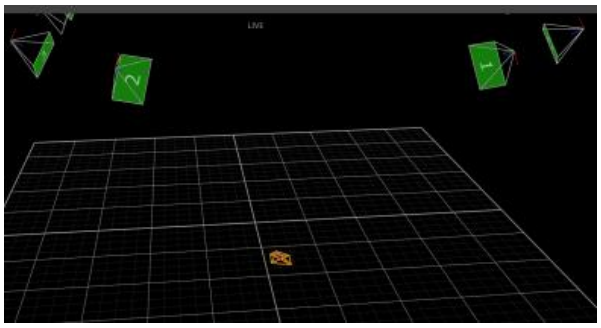
3. Figure: Modified hardware

2.2. VICON Motion Capture System

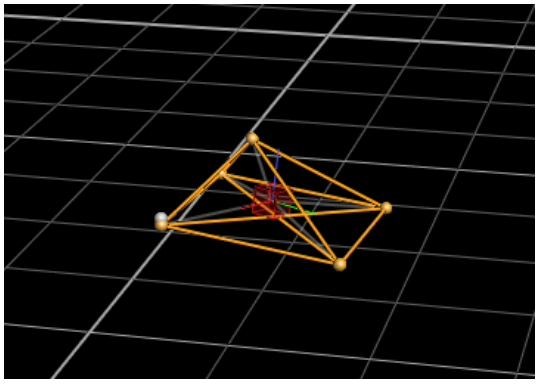
In the Brin Family Aerial Robotics lab, a VICON system is provided, which consists of twelve cameras, the system has to be calibrated from time to time to maintain precise measurements, this can be seen below.

4. Figure: Calibration of VICON

Using the vicon_bridge ROS package we were able to extract information from the VICON. To be able to detect the drone we had to place reflective markers on it and create an object using them in VICON that represents the drone.



5. Figure: VICON



6. Figure: The drone object

The ball also had to be covered with reflective tape to be tracked, at the end of a long iteration we decided to create an object of the ball using several markers to increase the robustness of the tracking and attached velcro to avoid the scenarios when the ball is bouncing out of the basket.

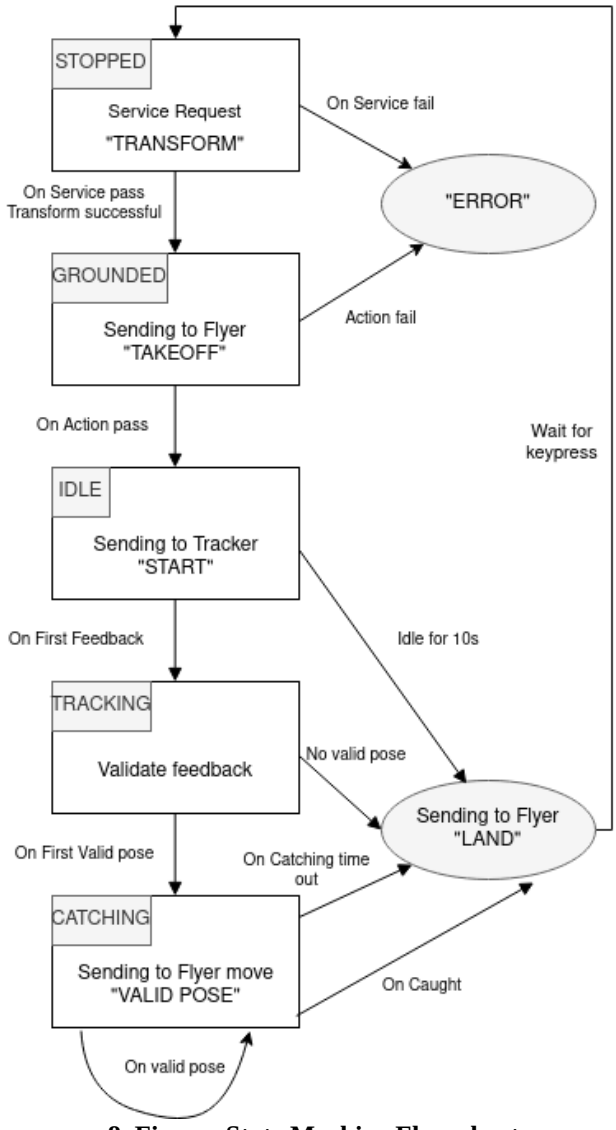
7. Figure: The reflective balls

The ROS message types are:
- ‘vicon_bridge/Markers’
- ‘vicon_bridge/Marker’

The former contains a list of the latter, and the topic where the data is published is:
- ‘/vicon/markers’

2.3. State machine

The states of the implemented system can be seen below. The “ERROR” state prevents the drone from flying, while in-air operations result in “LAND” state either through successful operation or intermediate failure of the outcome. There are five ROS packages used, one is the already discussed vicon_bridge, while there are four custom-made packages, reorient, estimation, flyer, and catcher that will be introduced in the following paragraphs.



8. Figure: State Machine Flow chart

2.4. ROS Nodes

2.4.1 Reorient

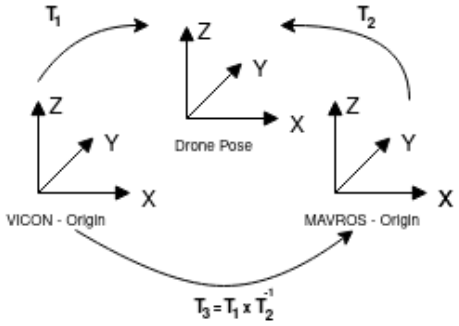
The VICON has its’ own coordinate system were the ball moves, but the drone can only be controlled with respect to its own inertial frame system hence, we need to connect the tf frames. The world frame is arbitrarily defined as X=0, Y=0, Z=0 using a static broadcast in the launch file, while the Drone inertial frame is statically broadcasted on the service call. The topics and the message types are the following.

The drone object from the VICON:

- /vicon/ENAE788m_mavrik/ENAE788m_mavrik
- geometry_msgs/Transform

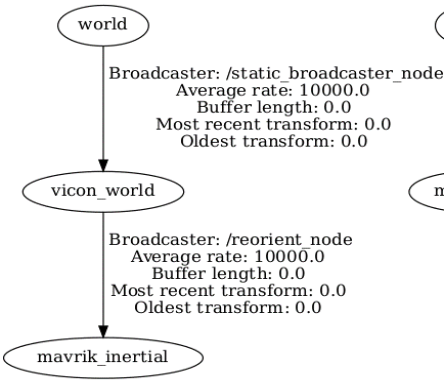
The drone’s position from MAVROS:

- /mavros/local_position/pose
- geometry_msgs/Pose

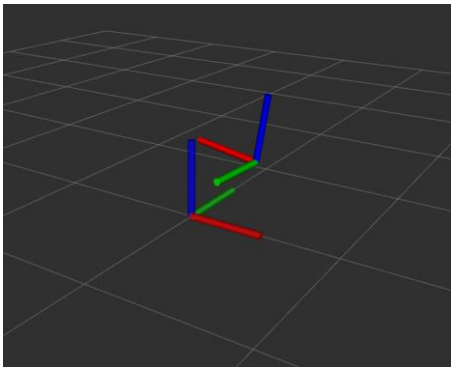


9. Figure: Transformation matrices

The transformations can be seen above while their result is below.



10. Figure: Tf2 frames on service request



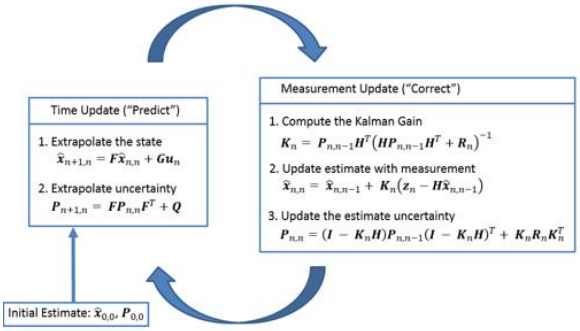
11. Figure: The Vicon and the Drone origin in Rviz

2.4.2 *Estimator*

Using VICON the position of the ball can be tracked. This data is noisy as the tracking is not consistent through the flight of the ball, additionally, a tool is required to predict the path of the ball therefore, a Kalman filter had to be introduced. First, it was implemented in Python, to get familiar with its working principles and tune it, later everything was shifted to C++, since the application has time and computation as its bottleneck.[2][3]

The position of the ball is extracted from the message type:

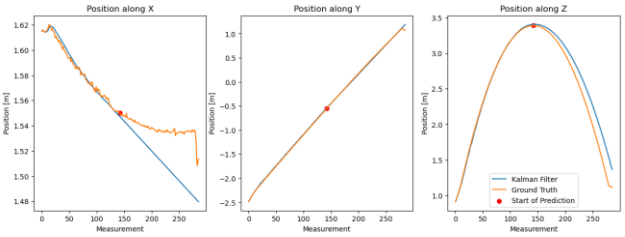
- geometry_msgs/Point.msg



Term	Name
x	State Vector
z	Measurements Vector
F	State Transition Matrix
u	Input Variable
G	Control Matrix
P	Estimate Covariance
Q	Process Noise Covariance
R	Measurement Covariance
w	Process Noise Vector
v	Measurement Noise Vector
H	Observation Matrix
K	Kalman Gain
n	Discrete-Time Index

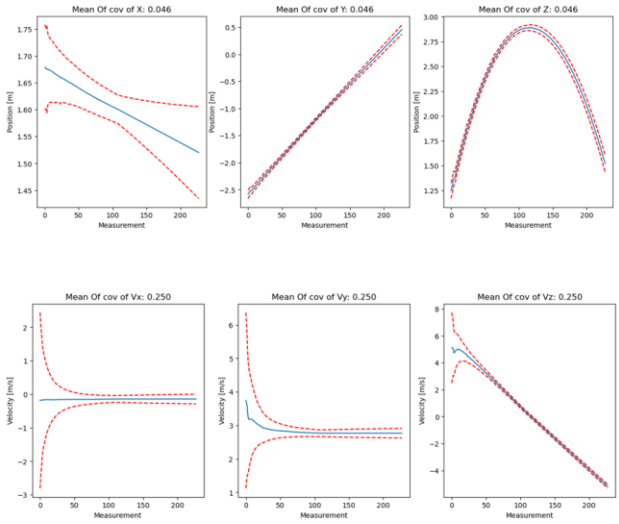
12. Figure: Implemented Kalman filter

In Jupyter notebook previously rosbaged throws were unpacked and simulated to adjust the related matrices and find out how long do we need to rely on the measurement before switching to pure prediction to achieve a precise enough measurement. The resulting estimations of the tuned filter can be seen below, in all three directions, the margin of error is a couple of cm which is sufficient for catching.



13. Figure: Real and estimated parabola

As more measurements are obtained the estimates of the states become more refined, in other words, the estimate homes in on the ground truth eliminating measurement noise. This is indicated by the reduction of the covariance magnitude corresponding to each state.



14. Figure: State estimate covariance over time

The tuned Kalman filter was ported over to C++ and provided to the network as a ROS Action Server. This Action server helps to asynchronously track the ball by subscribing to the aforementioned VICON topics and performing the KF update and corrections steps. It provides tuneable launch parameters like prediction timestep, start prediction height and target height. The feedback provided by the tracker server contains the predicted path and other useful information such as heading remaining flight time, and ball state. The predicted path can then be queried to find suitable poses for intercept.

2.4.3 *Flyer*

The Flyer is a ROS Action Server, in other world this module is responsible for handling the actions that move the drone.

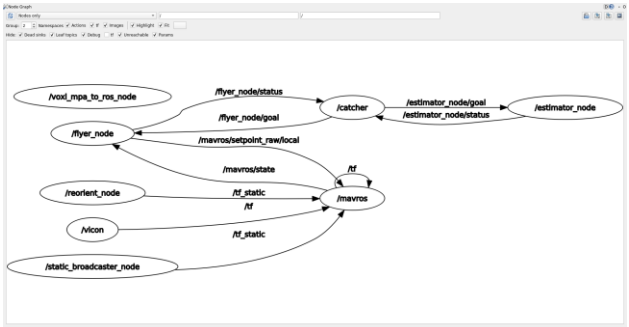
```
enum class FlyerCommand
{
    TAKEOFF=1,
    LAND,
    HOME,
    MOVE,
    SET_MODE,
    ARM,
    DISARM
};
#endif
```

15. Figure: Actions performed by the drone

The defined actions are cross related and happen sequentially, for instance, the "SET_MODE" to offboard, "ARM", TAKEOFF, "MOVE" to "HOME". Position control is used to move the drone with the on board QVIO as feedback. Each pose target is specified in the world frame which is then transformed to the drone inertial frame using the TF tree. This action server ensures that the drone reaches the required setpoint within a specified duration. It then continues to hold the position by publishing at a constant rate such that the OFFBOARD state is not exited. In addition to motion commands it provides an easy interface to trigger take-off and landing routines asynchronously.

2.4.4 Catcher

The catcher node is responsible for implementing the State Machine, in other words combining the previously mentioned packages to perform catching. The node consists of the tracker and flyer action client and the transform server client. In addition to the clients, it contains a process tick timer whose callback handles the execution of the state machine. The tick callback performs the actions corresponding to each state while the state transition conditions as implemented in the client feedback callbacks. Due to the asynchronous nature of transition-triggering events, the whole node was adapted to be run in a multithreaded fashion with critical variables being locked to ensure thread safety.



16. Figure: The State machine performed by the catcher node

2.5. Safety measurements

Since the drone operates fully autonomously several safety measurements were introduced. Two bounding boxes were defined: one for the VICON and one for the drone. The former is responsible for making sure that the tracking happens inside of the box, in other words, if the predicted parabola has the goal position outside of the area it won't be sent to the drone, while the drone cannot leave its bounding box, which is relative to its starting position. Moreover, on VICON several parameters were adjusted, for example, the number of cameras required for tracking, threshold, etc., to make sure that nothing else will be tracked as the ball. Additionally, there are timeout timers implemented in each state to ensure that the experiment terminates safely by moving to the "LAND" state. Due to the autonomous nature of the project, the execution is closely monitored, and the drone can be stopped using a kill switch.

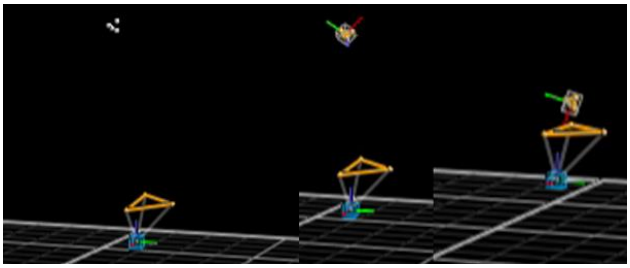
3. Conclusion

3.1. Challenges

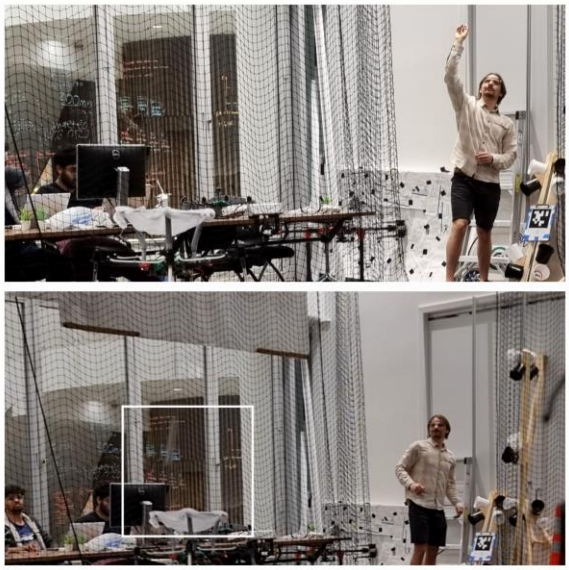
The project involved many different engineering aspects and integrating them to operate effectively in such a time and precision critical task was challenging. Since we did not use the best balls the VICON could not track it continuously, as a result, our data was noisy, and the estimation was not always reliable and fast enough. Another concern is that a throw takes approximately a couple of seconds, and getting the estimate and moving the drone there in such a small time slot was an interesting optimization problem. We tried to increase the time windows by increasing the distance between the thrower and the drone and speeding up our algorithm, by introducing multi-threading and adjusting the PID values in MAVROS.

3.2. Results

When the throw was close to the drone, we were able to perform successful catches that demonstrated that our implementation works, which can be seen below. When the throw is further from the drone, it reaches the position by a delay, therefore the catch is not successful however the system performs as expected.



17. Figure: Successful throw from VICON



18. Figure: Successful throw

3.3. Future development proposals

There are a couple of things that can be introduced in the future. The throwing area should increase to allow higher throws. A proper ball that is fully reflective could also correspond to a better estimate. In the meantime, the implemented Kalman filter could be also developed using more research in the area and it would be beneficial to introduce an Extended Kalman filter for real-world scenarios. Currently, a position controller is used for moving the drone, for better performance a Velocity-based trajectory planner not only would fasten up the fly but also would allow us to implement a smooth catching that does not allow the ball to bounce out.

4. Resources

The related codes can be found below:
https://github.com/Akash-Ravindra/Project_Catch

The related videos can be found below:
https://drive.google.com/drive/folders/15eWMsKni29h0M7qM7sx7zN-zz1yGpZ3c?usp=sharing

References

[1] https://www.youtube.com/watch?v=oMy5y-eQVeE

[2] Becker (www.kalmanfilter.net), Alex. 'Online Kalman Filter Tutorial'. Accessed 18 May 2023. https://www.kalmanfilter.net/.

[3] Zarchan, Paul Musoff, Howard. (2015). Fundamentals of Kalman Filtering - A Practical Approach (4th Edition) - Progress in Astronautics and Aeronautics, Volume 246 - 9. Cannon-Launched Projectile Tracking Problem. American Institute of Aeronautics and Astronautics/Aerospace Press (AIAA). Retrieved from https://app.knovel.com/hotlink/pdf/id:kt010RJ5D1/fundamentals-kalman-filtering/cannon-launched-projectile