# MikroKopter Documentation

## Cyphy Laboratory

---

Inkyu Sa

October 31, 2011

**Contents**

# I. INTRODUCTION

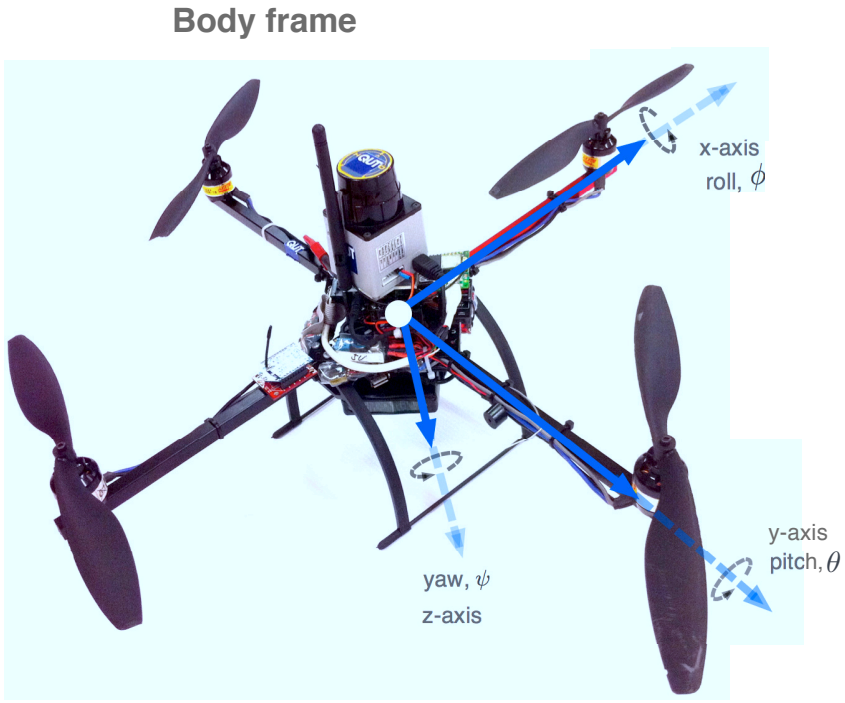
## A. Purpose

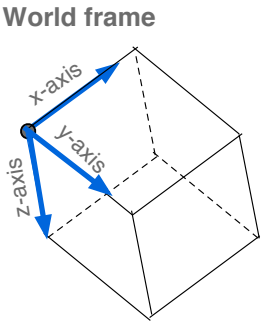
The objectives of this document are to create written history while implementing the quadrotor using Mikrokopter (MK). In addition it will be a good opportunity to share this knowledge with other researchers who are interested in using the MK platform.

## B. Coordinate systems

The MK doesn't specify a scientific coordinate system. It only has pitch, roll and yaw angles. The red bar is the front and the opposite is the rear. Pitch (Nick) angle results in forwards and backwards movements. In contrast, roll angle results in left and right movements. When we push all the way up the pitch stick (right stick on mode2), "StickNick" value (516) is transmitted to the FlightControl(FC). This implies "moving forward with the fastest speed". It is the maximum value of pitch angle. When we pull all the way down the pitch stick, "StickNick" value (-592) is transmitted to the FC. This implies "moving backward with the fastest speed". On the other hand, when pushed all the way right, the roll stick, "StickRoll" value (-516) is transmitted to the FC. When pushed all the way left, the roll stick, "StickRoll" value (512) is transmitted to the FC. Commonly the forward direction is the x-axis of robots and the right direction is the positive y-axis. Pitch angle implies rotation around the y-axis and the roll angle implies rotation around x-axis according to the right-hand rule. Positive pitch angle causes the vehicle to move backwards- the opposite of MK notation. Positive roll angle causes it to move right-again the opposite of MK notation.


FIG. 1: The MK coordinate frame




**Note**

All coordinate frames is ruled by right hand.

FIG. 2: The defined coordinate frame

### C. Position control

In order to control the quadrotor, a traditional PD controller is implemented. The inputs for the controller are desired position and velocity and the outputs are pitch and roll commands. The laser range finder provides x,y positions and a yaw angle every 100ms and these PD controllers generate outputs at the same rate. The unit of the position and angle are represented in SI units, such as metres and radian.

$$ {}^{W}\theta^{*} = K_p({}^{W}x^{*} - {}^{W}x) + K_d\frac{d}{dt}({}^{W}x^{*} - {}^{W}x) \tag{1} $$

$$ {}^{W}\phi^{*} = K_p({}^{W}y^{*} - {}^{W}y) + K_d\frac{d}{dt}({}^{W}y^{*} - {}^{W}y) \tag{2} $$

${}^{W}\theta^{*}$ notes the pitch input in world frame and ${}^{W}x^{*}$ is the reference input which the quadrotor should follow. ${}^{W}x$ is the current position of the quadrotor in the world frame and this data is provided from the laser scan matcher. The desired output to the MK must be in the range $\pm$ 128. Inputs are divided by the scale factor(120).

Then command angles are with respect to the world frame. The world and body frames are related by the yaw angle transformation.

$$ {}^{B}\boldsymbol{R}_{W}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}, \tag{3} $$

where $\psi$ is the angle of the world frame with respect to the body frame.

$$ \begin{bmatrix} {}^{B}\theta^{*} \\ {}^{B}\phi^{*} \end{bmatrix} = \boldsymbol{R}(\psi) \cdot \begin{bmatrix} {}^{W}\theta^{*} \\ {}^{W}\phi^{*} \end{bmatrix} \tag{4} $$

Finally, the commands in the body frame are

$$ {}^{B}\theta^{*} = \cos\psi \cdot {}^{W}\theta^{*} - \sin\psi \cdot {}^{W}\phi^{*} \tag{5} $$

$$ {}^{B}\phi^{*} = \sin\psi \cdot {}^{W}\theta^{*} + \cos\psi \cdot {}^{W}\phi^{*} \tag{6} $$

| Pitch | −90° | −60° | −50° | −45° | −15° | 0° | 15° | 45° | 50° | 60° | 90° |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1st | 602 | 522 | 458 | 428 | 152 | 0 | -122 | -431 | -457 | -531 | -617 |
| 2nd | 602 | 525 | 468 | 432 | 154 | 0 | -123 | -433 | -462 | -532 | -618 |
| 3rd | 602 | 523 | 470 | 422 | 153 | 0 | -122 | -432 | -463 | -532 | -617 |
| Mean | 602 | 523.3 | 465.3 | 427.3 | 153 | 0 | -122.3 | -432 | -460.6 | -531.6 | -617.3 |

TABLE I: MK AccNick Acceleration versus pitch true angle measurement ruled by right hand.

### D. MK acceleration scale factor measurement

MK doesn't provide either SI units or documents explaning MK units. These data are unscaled measurements. Hence we try to figure out what the actual units are. This data is measured by an external accelerometer and MikroKopter Tool V1.72.

*a.* ***Pitch acceleration*** The contents of TABLE I are the pitch acceleration data from the MK. The unit of the measurement is unknown.

Fig. 3 shows the linear line fitting with the measured data. In the x-direction the equation of motion is

$$\ddot{x} = k \sin\theta \tag{7}$$

Where $\ddot{x}$ refers to the acceleration data provided from MK quadrotor, $k$ is the scale factor and $\theta$ is the pitch angle. Finally, $k = 606.3 \ \ units/g$ can be calculated and the acceleration value also can be obtained in units of $1g$.

*b.* ***Roll acceleration*** The contents of TABLE II are the roll acceleration data from the MK. The unit of measurement is unknown as well.

Fig. 4 shows the linear line fitting with the measured data. According to the equation 7, the $k = 603.44 \ units/g$ can be computed for the roll acceleration.

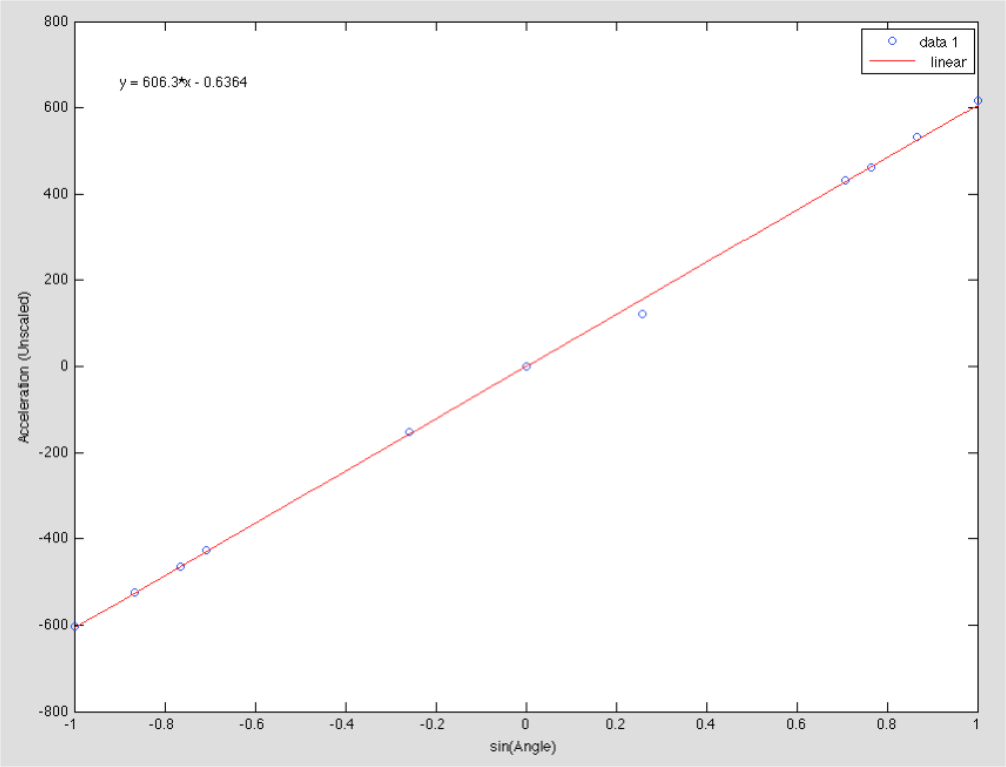



FIG. 3: MK pitch acceleration data versus true pitch angle.

| Roll | −90° | −60° | −50° | −45° | −15° | 0° | 15° | 45° | 50° | 60° | 90° |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1st | 612 | 522 | 458 | 436 | 162 | 0 | -167 | -430 | -462 | -522 | -587 |
| 2nd | 611 | 521 | 456 | 443 | 161 | 0 | -162 | -432 | -463 | -515 | -588 |
| 3rd | 613 | 523 | 457 | 439 | 160 | 0 | -165 | -431 | -463 | -517 | -588 |
| Mean | 612 | 522 | 457 | 439.3 | 161 | 0 | -164.6 | -431 | -462.6 | -518 | -587.6 |

TABLE II: MK roll acceleration versus true roll angle measurement ruled by right hand

*c.* ***Z acceleration*** The drifting problem of raw ACCZ ADC measurement was found when vibration came from four rotors and it was thought that error accumulated when analog data was converted into digital. Hence the preprocessed MK default ACCZ value is exploited to estimate the z-velocity. The MK provides vibration robust filtered data but has an issue that it doesn't know what the unit is. In order to use this acceleration data

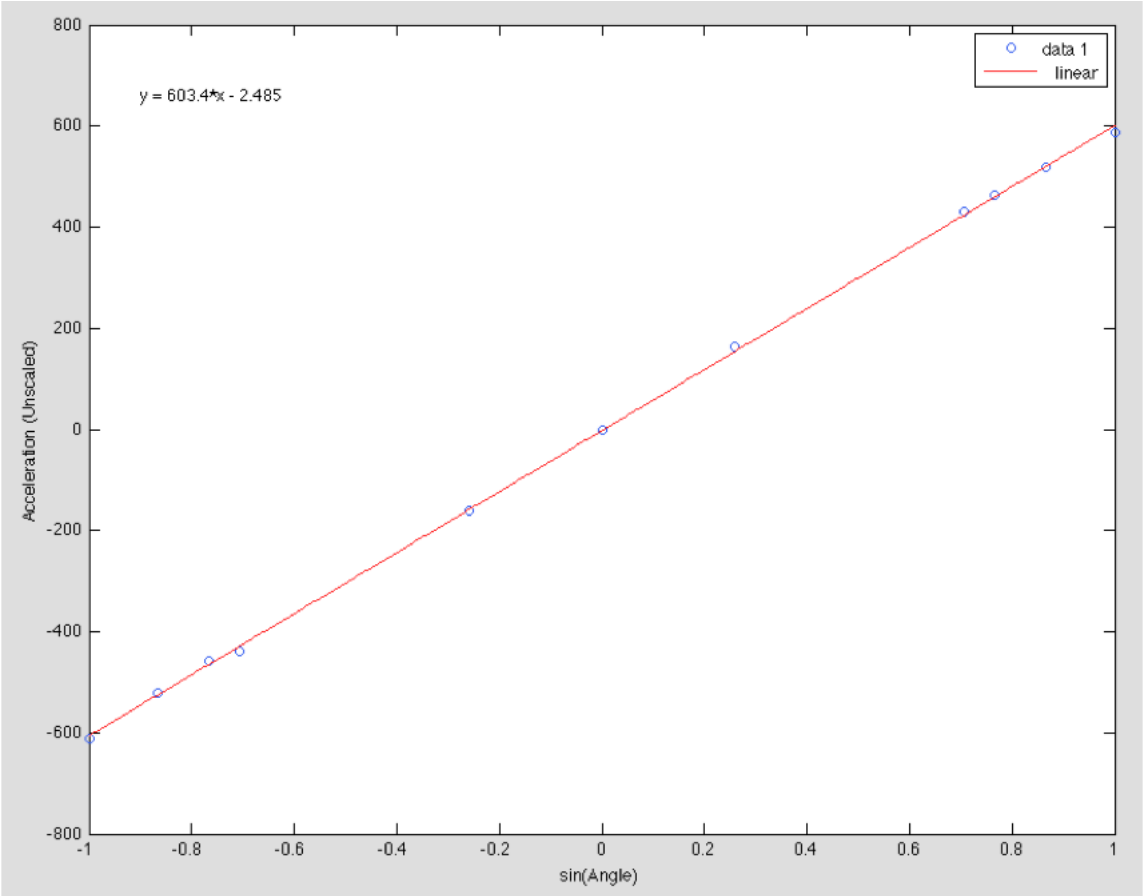



FIG. 4: MK AccRoll acceleration versus true roll angle.

for the complementary filter, the unit of data should be addressed. An experiment was conducted with a XSENS sensor. We put a XSENS on the quadrotor and logged XSENS and MK ACCZ data concurrently at the same sample rate(20 Hz), then matched the two data manually. Figure 5 shows the result.

$$ACCZ_{xsens}(t)[m/s^2] = \frac{ACCZ_{miko}(t)}{15.5} + 9.8 \tag{8}$$

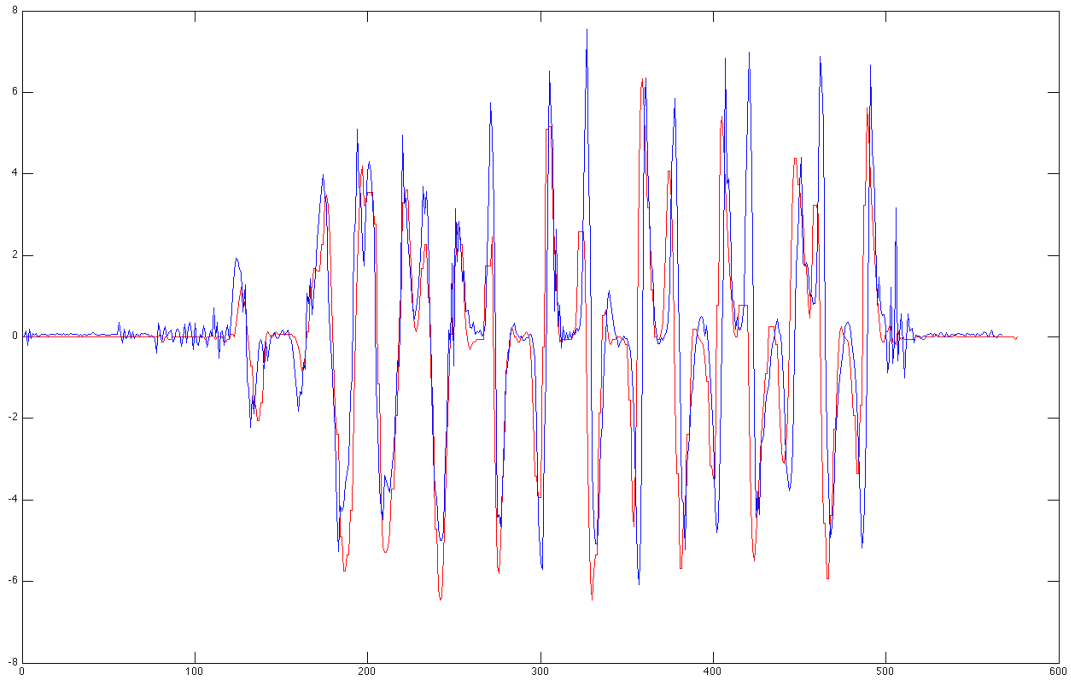



FIG. 5:

### E. SI units and log message protocol

#### 1. *SI units*

All units are used in the system following the SI units. All angle measurements are represented in **Radian** and **Metre** as a unit of all distance and angle measurements and **Second** is a unit of time. $\mathbf{m/s^2}$ is a unit of all acceleration.

#### 2. *log message protocol*

There are two log messages. One is logging attitude of the quadrotor such as pitch angle and acceleration along x axis. Another measures data using the laser range finder like position, velocity and estimation of velocity. The reason why we have two log messages is because of different sampling rates of the laser range finder and the MK quadrotor. Position data based on the laser range finder comes into every $100ms$ and attitude of the quadrotor

| Order | Notation | Description | Unit |
|---|---|---|---|
| 1 | g_W_Vel.x | World velocity along x axis. | $m/s$ |
| 2 | g_B_Vel.y | Body velocity along x axis. | $m/s$ |
| 3 | g_B_esti_Vel.x | Body estimated velocity along x axis. | $m/s$ |
| 4 | g_W_Vel.y | World velocity along y axis. | $m/s$ |
| 5 | g_B_Vel.y | Body velocity along y axis. | $m/s$ |
| 6 | g_B_esti_Vel.y | Body estimated velocity along y axis. | $m/s$ |
| 7 | g_CurrentPosition.x | current x position. | $m$ |
| 8 | g_CurrentPosition.y | current y position. | $m$ |
| 9 | g_CurrentPosition.theta | current angle. | radian |
| 10 | index | Counter increasing +1 every 100ms | number |

TABLE III: $100ms$ loop log message format

is measured every $50ms$.

*a.* ***100ms sampling time*** TABLE III shows the $100ms$ sampling time log message protocol.

| Order | Notation | Description | Unit |
|---|---|---|---|
| 1 | AnglePitch | Pitch angle | radian |
| 2 | AngleRoll | Roll angle | radian |
| 3 | AngleYaw | Yaw angle | radian |
| 4 | AccX | X acceleration with respect to the body frame | $m/s^2$ |
| 5 | AccY | Y acceleration with respect to the body frame | $m/s^2$ |
| 6 | AccZ | Z acceleration with respect to the body frame | $m/s^2$ |
| 7 | ControlPitch | Pitch angle commanded to the quadrotor (-128 $\sim$ 128) | unknown |
| 8 | ControlRoll | Roll angle commanded to the quadrotor (-128 $\sim$ 128) | unknown |
| 9 | ControlYaw | Yaw angle commanded to the quadrotor (-128 $\sim$ 128) | unknown |
| 10 | StickPitch | Transmitter pitch stick value (-128 $\sim$ 128) | unknown |
| 11 | StickRoll | Transmitter roll stick value (-128 $\sim$ 128) | unknown |
| 12 | StickYaw | Transmitter yaw stick value (-128 $\sim$ 128) | unknown |
| 13 | StickThrust | Transmitter thrust stick value (0 $\sim$ 255) | unknown |
| 14 | index | Counter increasing +1 every 50ms | number |

TABLE IV: $50ms$ loop log message format

*b.* ***50ms sampling time*** TABLE IV shows the $50ms$ sampling time log message protocol.

### F. Complex dynamics of the MK quadrotor

In this section, dynamics of the MK quadrotor are presented.

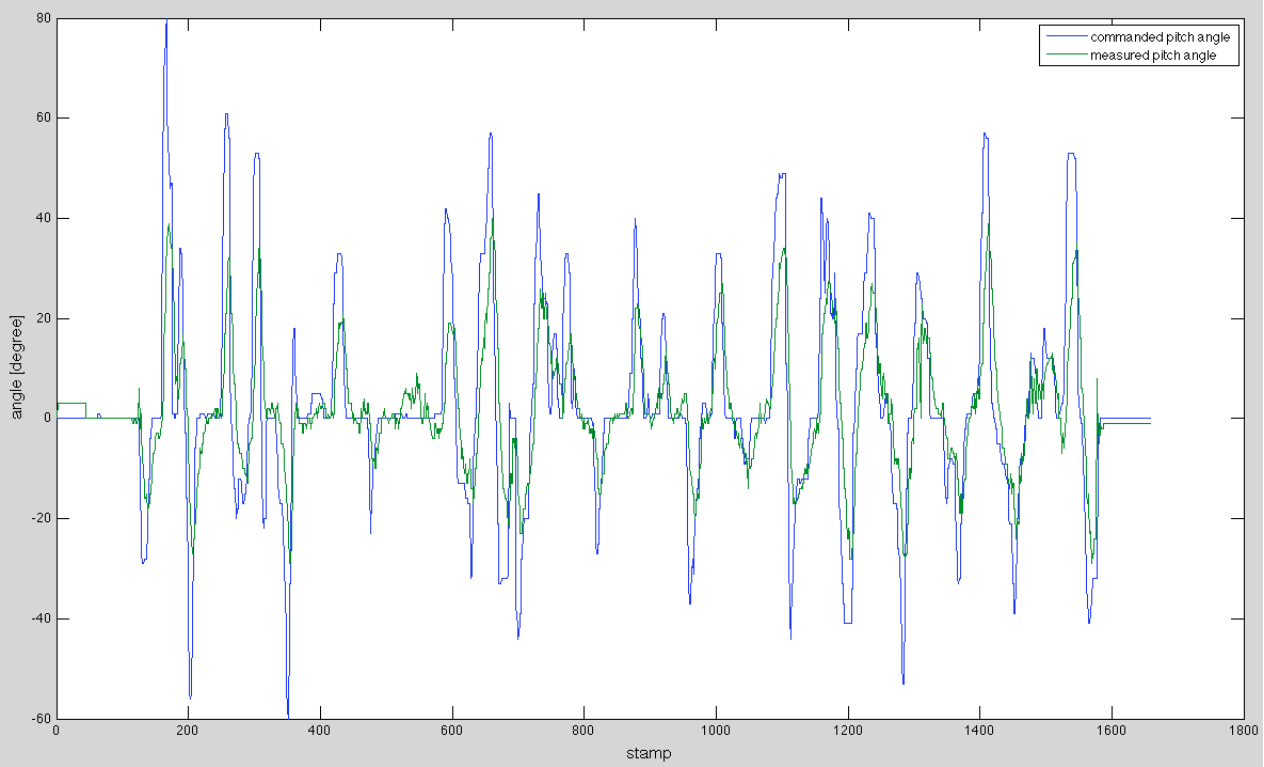



FIG. 6: Measured angle versus commanded angle

The lag can be resolved by looking at the vehicle velocity, in the vehicle frame, and correlating that with the pitch angle. Taking the cross correlation, Fig. 8 shows a lag of approximately 4 time steps which implies that the MK responses are $200ms$ later.

### G. System latency

In order to find out the system latency between the Gumstix and the ground station, the network time protocol (NTP) server is exploited. NTP allows the setting up of two systems with less than $1ms$ time error. Fig. 9 shows synchronised systems and delay. After synchronisation, time duration was measured using ROS time stamp function. This function provides $\mu s$ accuracy and it is sufficient to measure the time period.

Fig. 10 shows $100ms$ time delay of the laser range finder. Fig. 11 also shows $5ms$ delay of the ICP scan matcher. The WIFI latency can be determined by subtraction of

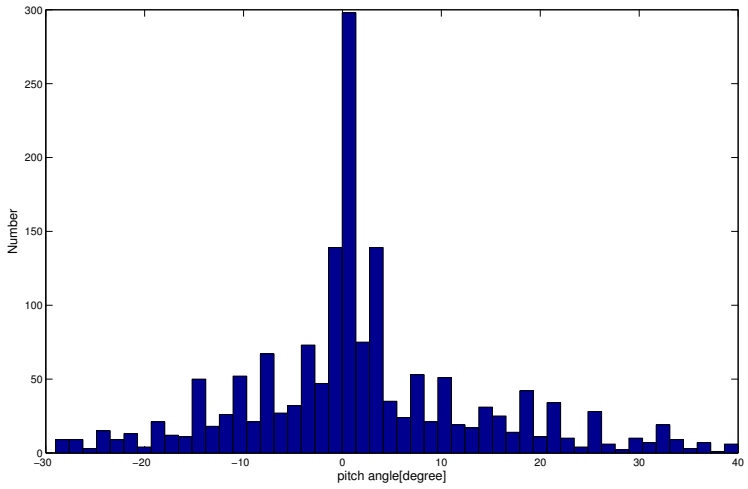



FIG. 7: Pitch angle histogram

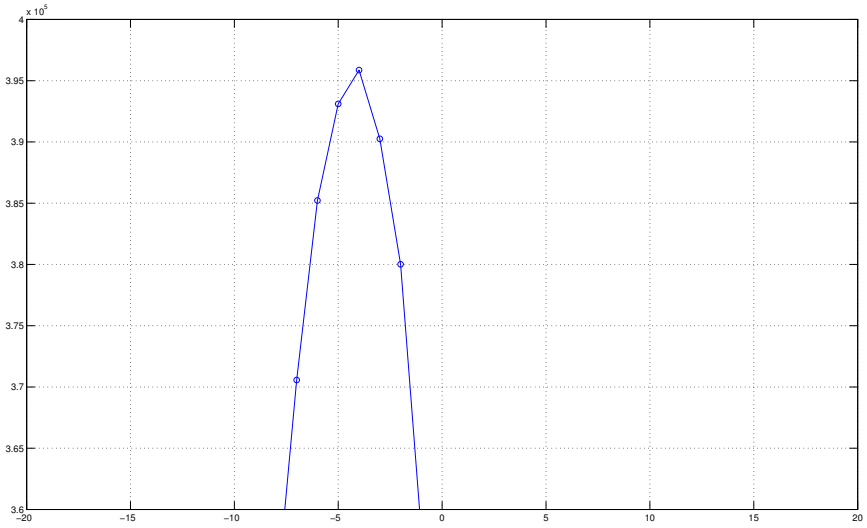



FIG. 8: Cross correlation

$Current\ Time - Received\ Time$. The WIFI latency is in the order of $7ms$.

In summary, figure 12 shows the total system latency in the system. As shown in the Fig. 8, the MK has about $200ms$ natural response time and WIFI, laser range finder and scan matcher have $167ms$ delay.

**Desktop Time**

```
enddl22@BEE-PG-46634:/opt/ros/cturtle/stacks/ccny-ros-pkg/scan_tools/canonical_s
can_matcher$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*LOCAL(0)        .LOCL.           5 l   57   64  377    0.000    0.000   0.001
enddl22@BEE-PG-46634:/opt/ros/cturtle/stacks/ccny-ros-pkg/scan_tools/canonical_s
can_matcher$
```

**Gumstix Time, 1.4ms delay and -1.3ms offset.**

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*BEE-PG-46634    LOCAL(0)         6 u   15   64   37    1.400   -1.368   5.944
 224.0.1.1       .MCST.          16 u    -   64    0    0.000    0.000   0.031
enddl22@overo:~$
```

FIG. 9: Time synchronization using NTP server

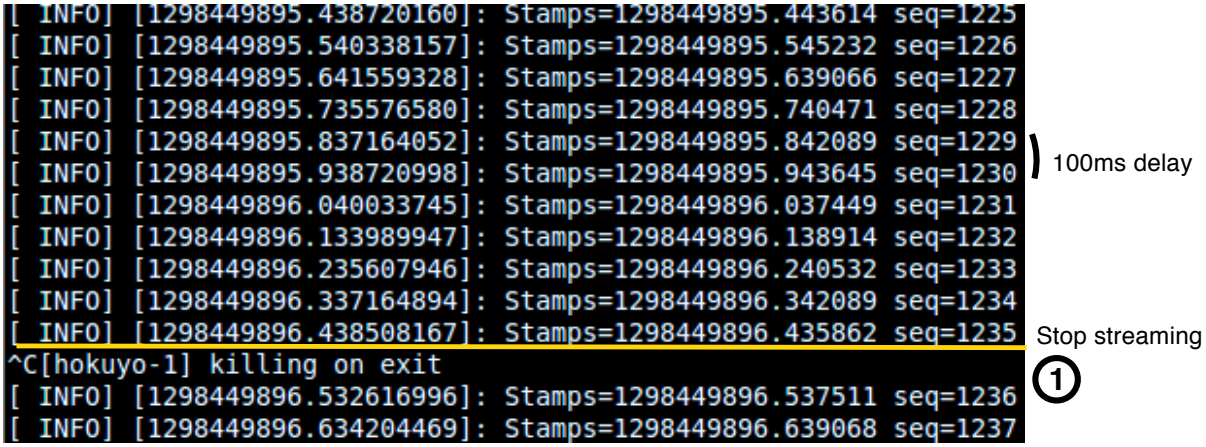



FIG. 10: Laser time delay

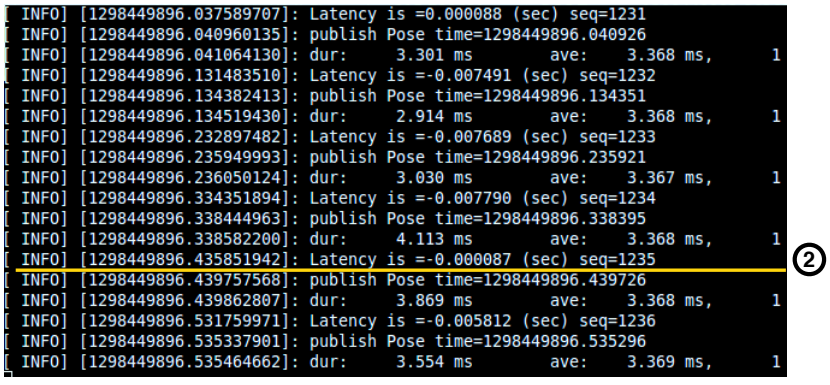



FIG. 11: ICP time delay and wifi latency

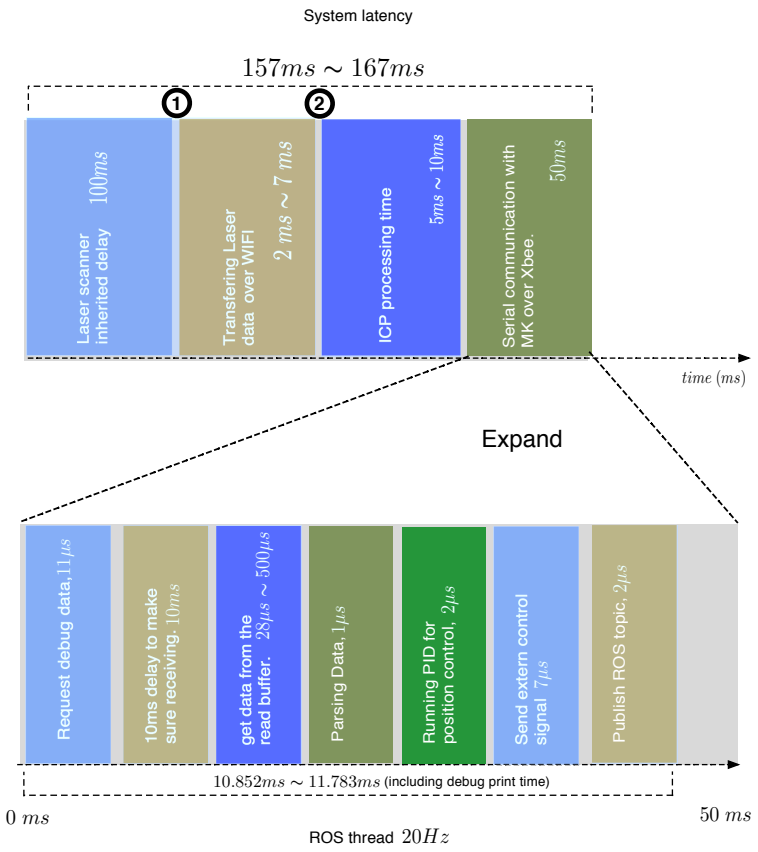



FIG. 12: The total system latency

## H. Velocity estimation

It is necessary to use a velocity estimator because of the complex dynamics and latency of the system. A complementary filter is exploited and this requires accelerations of each axis in order to estimate velocities.

### 1. *X axis acceleration*

the velocity in body frame. The acceleration along the x axis is

$$a_x = {}^B\ddot{x}\cos\theta - g\sin\theta \tag{9}$$

$${}^B\ddot{x} = \frac{a_x + g\sin\theta}{\cos\theta} \tag{10}$$

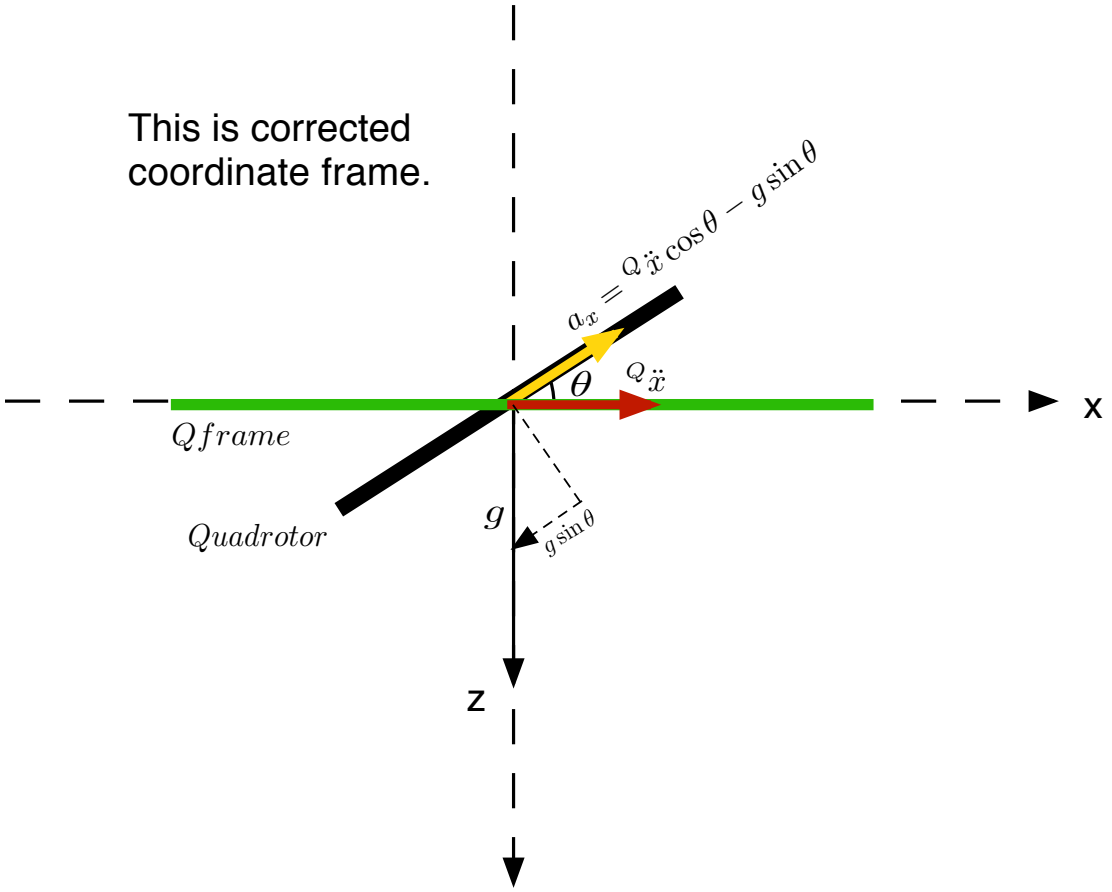



FIG. 13: X axis acceleration

### *2. Y axis acceleration*

The acceleration along the y axis is

$$a_y = {}^{B}\ddot{y}\cos\phi + g\sin\phi \tag{11}$$

$$ {}^{B}\ddot{y} = \frac{a_y - g\sin\phi}{\cos\phi} \tag{12}$$

${}^{w}x$, ${}^{w}y$, ${}^{w}v_x$ and ${}^{w}v_y$ refer to the position and velocity in world frame respectively. $\psi$ is measured orientation in world frame. ${}^{b}v_x$ and ${}^{b}v_y$.

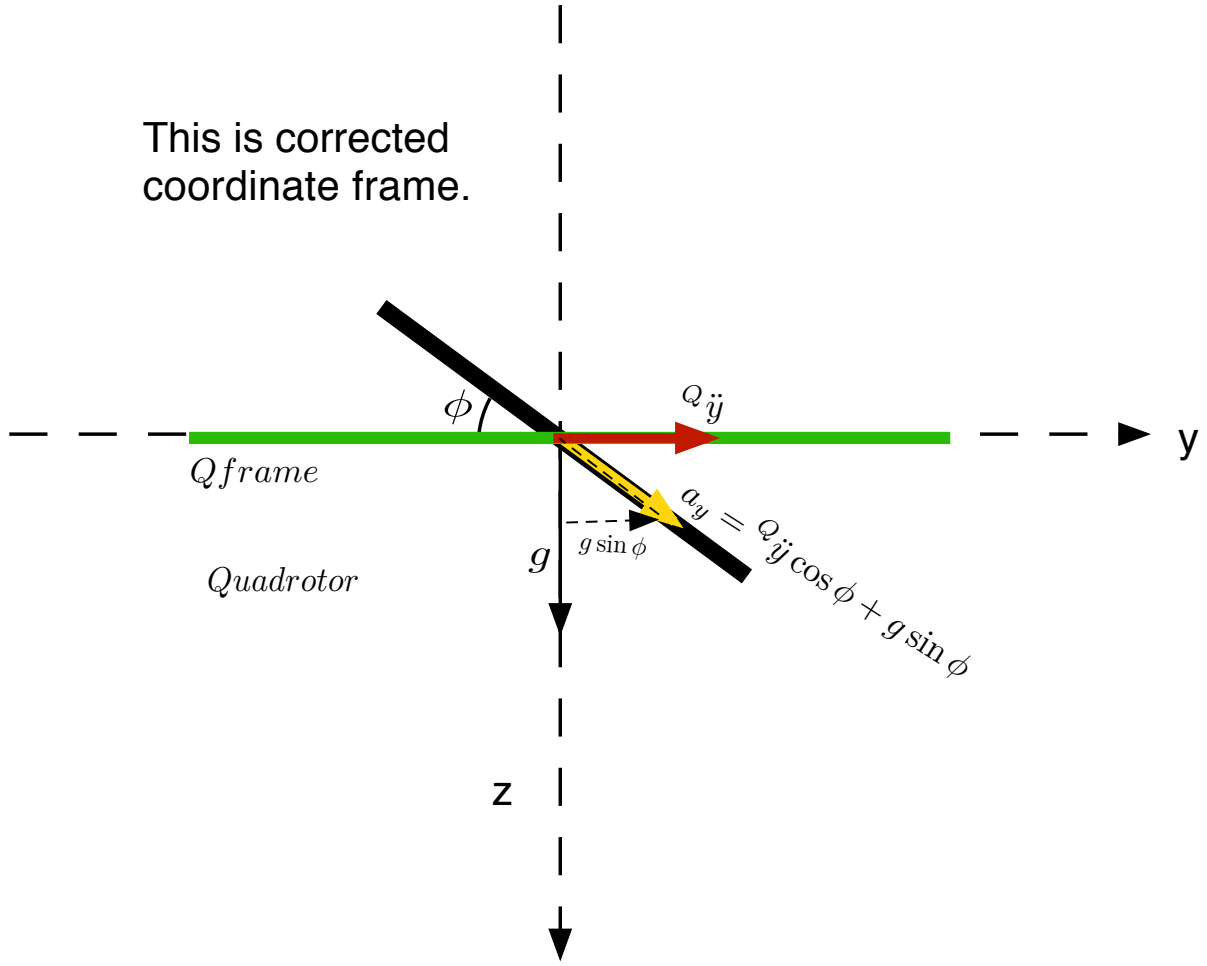


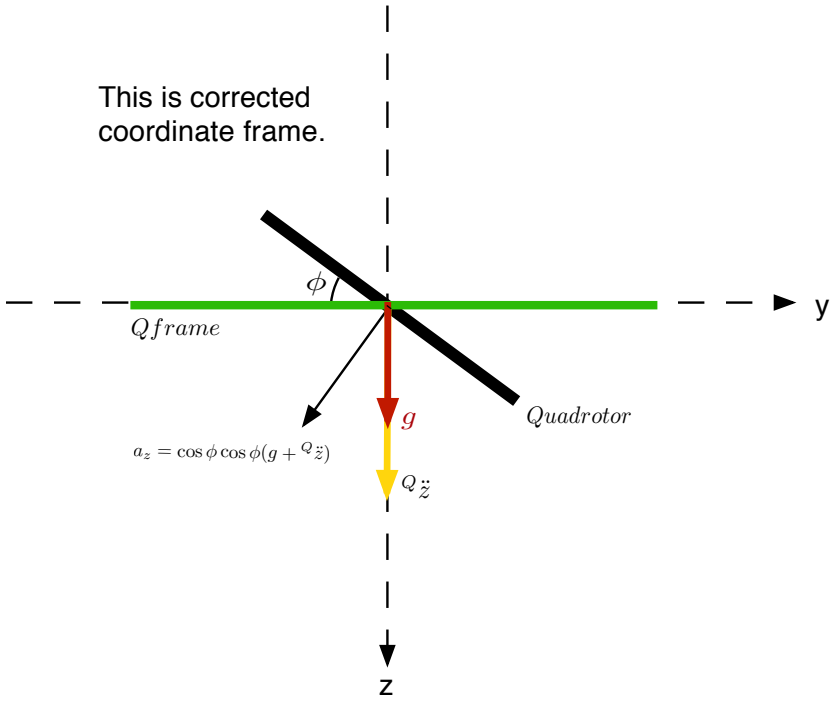

FIG. 14: Y axis acceleration

### *3. Z axis acceleration*

The acceleration along the z axis is

$$a_z = \cos\phi\cos\phi(g + {}^{Q}\ddot{z}) \tag{13}$$

$${}^{Q}\ddot{z} = \frac{a_z}{\cos\phi\cos\theta} - g \tag{14}$$


This is corrected
coordinate frame.
$\phi$
$Qframe$
y
$g$
$Quadrotor$
$a_z = \cos\phi\cos\phi(g + {}^{Q}\ddot{z})$
${}^{Q}\ddot{z}$
z

### a. *A complementary filter*

$$\hat{v}_x(t+1) = \hat{v}_x(t) + T\ (\ k\ (\ {}^{B}v_x - \hat{v}_x(t)) + {}^{Q}\ddot{x}) \tag{15}$$

$$\hat{v}_y(t+1) = \hat{v}_y(t) + T\ (\ k\ (\ {}^{B}v_y - \hat{v}_y(t)) + {}^{Q}\ddot{y}) \tag{16}$$

$$\hat{v}_z(t+1) = \hat{v}_z(t) + T\ (\ k\ (\ {}^{B}v_z - \hat{v}_z(t)) + {}^{Q}\ddot{z}) \tag{17}$$

where $T$ is the sample interval, and $k$ is the filter gain.

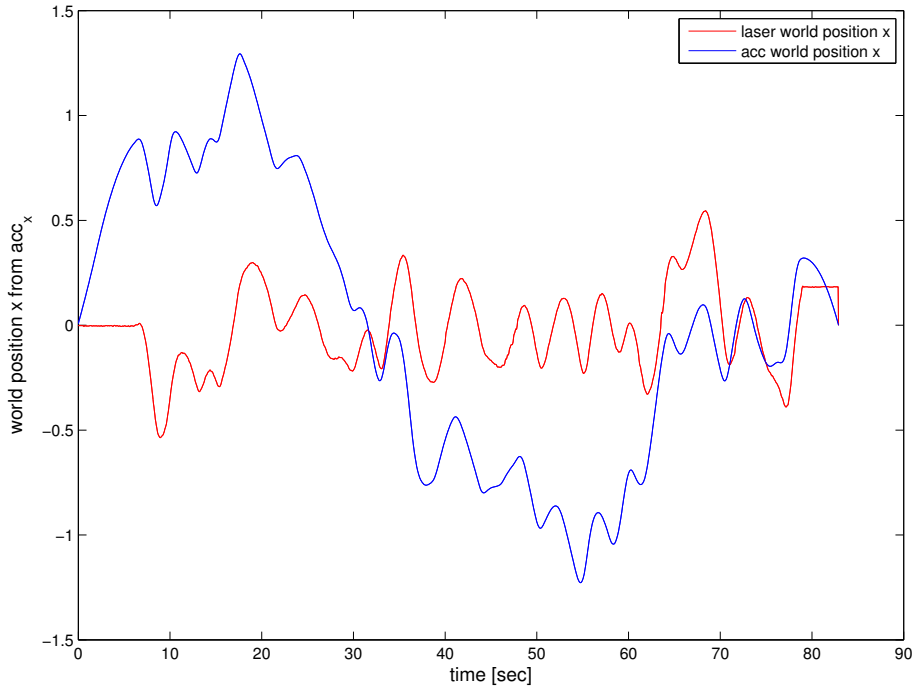



FIG. 16: X position in world frame for laser-based localizer

Fig. 16 is the one sample predicted position using Equation 18. We assume that the quadrotor is hovering and the velocity of it is almost constant.

$$\hat{x}(t+1) = x(t) + v(t) \cdot T \tag{18}$$

$$= x(t) + \frac{x(t) - x(t-1)}{T} \cdot T \tag{19}$$

$$= 2x(t) - x(t-1) \tag{20}$$

Fig. 17 shows the velocity in body frame calculated derivative of the x position and Equation 4.

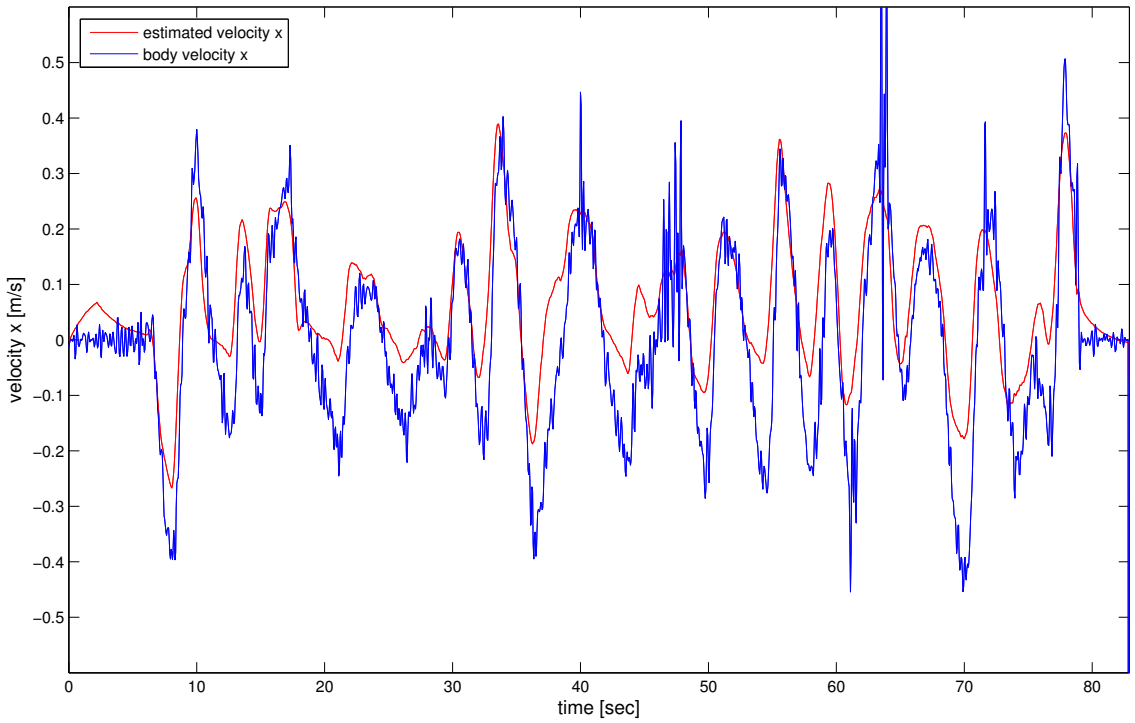



FIG. 17: x velocity versus estimated x velocity in body frame

Fig. 17 shows the estimated velocity in body frame using Equation 15 and 16.

Fig. 18 plots the command data which generated by Equation 21.

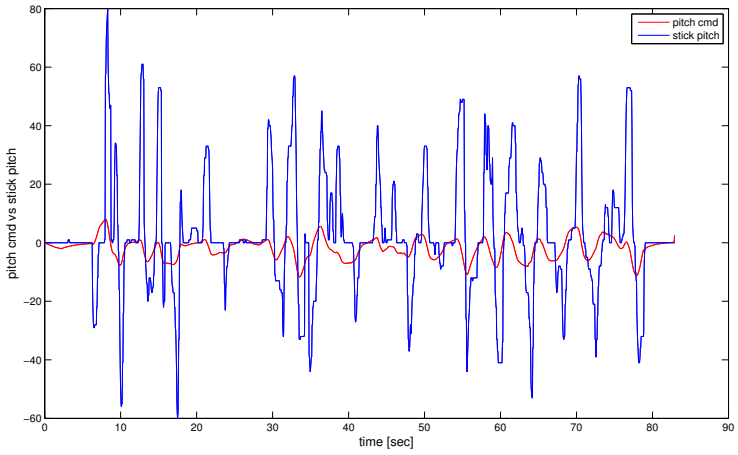



FIG. 18: Pitch command generated with P velocity controller

$$^{W}\theta^{*}(t) = K_p(^{W}v_x^{*}(t) - {}^{W}\hat{v}_x(t)) + K_d\frac{d}{dt}(^{W}v_x^{*}(t) - {}^{W}\hat{v}_x(t)) \tag{21}$$

$$^{W}\phi^{*}(t) = K_p(^{W}v_y^{*}(t) - {}^{W}\hat{v}_y(t)) + K_d\frac{d}{dt}(^{W}v_y^{*}(t) - {}^{W}\hat{v}_y(t)) \tag{22}$$

$$^{W}\tau^{*}(t) = K_{p_{gas}}(^{W}v_z^{*}(t) - {}^{W}\hat{v}_z(t)) + K_{d_{gas}}\frac{d}{dt}(^{W}v_z^{*}(t) - {}^{W}\hat{v}_z(t)) \tag{23}$$

In the Equation 21 we assume that $^{W}v_x^{*}(t)$ and $^{W}v_y^{*}(t)$ are zero to hovering.

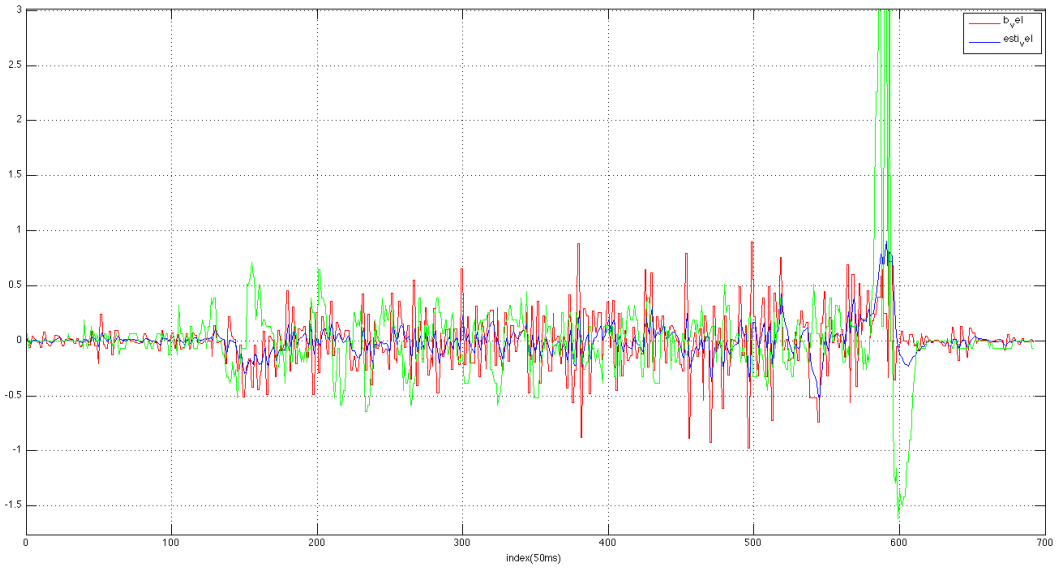



FIG. 19: This plot shows estimated velocity along z axis. Green, Red and Blue lines stand for ACCZ from MK, body velocity and estimated velocity. ACCZ is sampled at 20 Hzand estimated velocity is computed at the same rate. The body velocity is calculated based on laser range finder height data and has a sample rate of 10 Hz. Note that body velocity is smoothed.

## I. ROS implementation

It is easily achieved to integrate systems with ROS. All messages come into the ROS core and are redistributed to nodes through the TCP/IP protocol. This feature allows implementation of the software packages as modules, ROS stacks or packages. Fig. 20 shows simple implementation based on the ROS. In this figure, dark gray boxes denotes the ROS nodes which act as an individual process. The canonical scan matcher receives laser range data and produces 2D positions using the ICP algorithm. It publishes */pose2D* topic every $100ms$ and the data structures are shown in TABLE V and TABLE VI. A Miko Serial node functions are as obtaining IMU data from the flight control board while a quadrotor is flying and publishing */mikoImu* topic every $50ms$. These two topics can be recorded to create a log file and be played using rosbag package. Finally, a complementary Filter node

| Name | geometry_msgs/Pose2D | | |
|---|---|---|---|
| Description | This expresses a position and orientation on a 2D manifold. | | |
| **Data type** | **definition** | **Unit** | **Description** |
| float64 | x | $m$ | x position in world frame |
| float64 | y | $m$ | y position in world frame |
| float64 | theta | $radian$ | theta angle in world frame |

TABLE V: /Pose2D topic data type

computes body and estimated velocity given position and IMU data with Equation 15 and 16. With this complementary filter, the position control of the quadrotor can be achieved. The demonstration footage can be found on the Youtube.

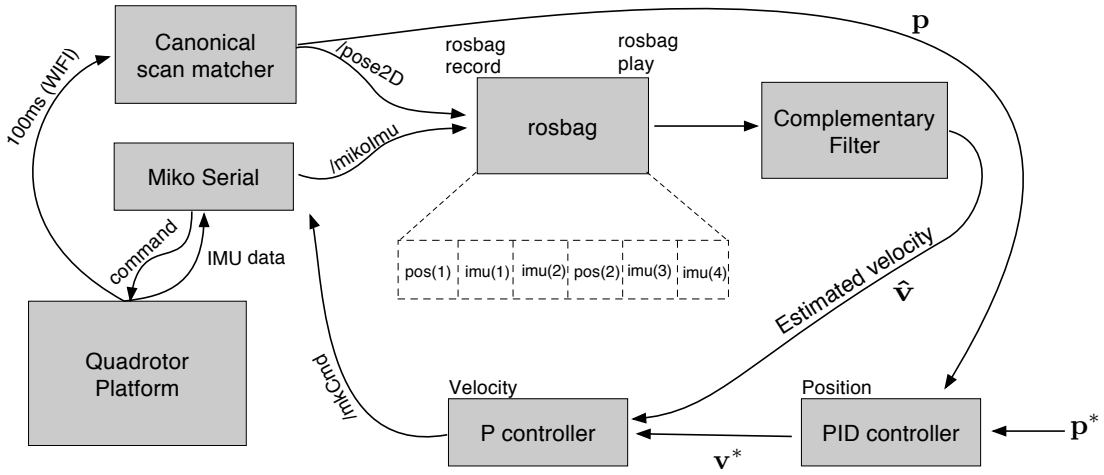



FIG. 20: ROS complementary filter implementation

### J. Yaw control

Yaw control of the quadrotor is relatively easier than pitch, roll and height control. Gyro data is drifting overtime which implies the vehicle is going to change its heading. Therefore

| Name | sensor_msgs/mikoImu | | |
|---|---|---|---|
| Description | This expresses a attitude of a quadrotor. | | |
| **Data type** | **definition** | **Unit** | **Description** |
| float64 | pitchAngle | *radian* | pitch angle in body frame |
| float64 | rollAngle | *radian* | roll angle in body frame |
| float64 | yawAngle | *radian* | yaw angle in body frame |
| float64 | AccX | $m/s^2$ | Acceleration along x axis in body frame |
| float64 | AccY | $m/s^2$ | Acceleration along Y axis in body frame |
| float64 | AccZ | $m/s^2$ | Acceleration along Z axis in body frame |

TABLE VI: /mikoImu topic data type

| Name | sensor_msgs/mikoCmd | | |
|---|---|---|---|
| Description | This expresses command input to the quadrotor. | | |
| **Data type** | **definition** | **Unit** | **Description** |
| int32 | pitch | | pitch cmd -128 $\sim$ 128 |
| int32 | roll | | roll cmd -128 $\sim$ 128 |
| int32 | yaw | | yaw cmd -128 $\sim$ 128 |
| int32 | throttle | | throttle cmd 0 $\sim$ 512 |

TABLE VII: /mikoCmd topic data type

we need to use the absolute measurement data which is less changed as time passes. Laser range data can provide the yaw angle with minimum error as its variance is smaller than a gyro sensor. A traditional PID controller is used to hold the position. The equation 24 shows this controller and figure 21 presents the output and yaw angle. In this figure, the pilot gives the artificial angle using the transmitter and the controller generates the output to keep the initial angle "0".

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{24}$$

where $e(t) = \psi^* - \psi$

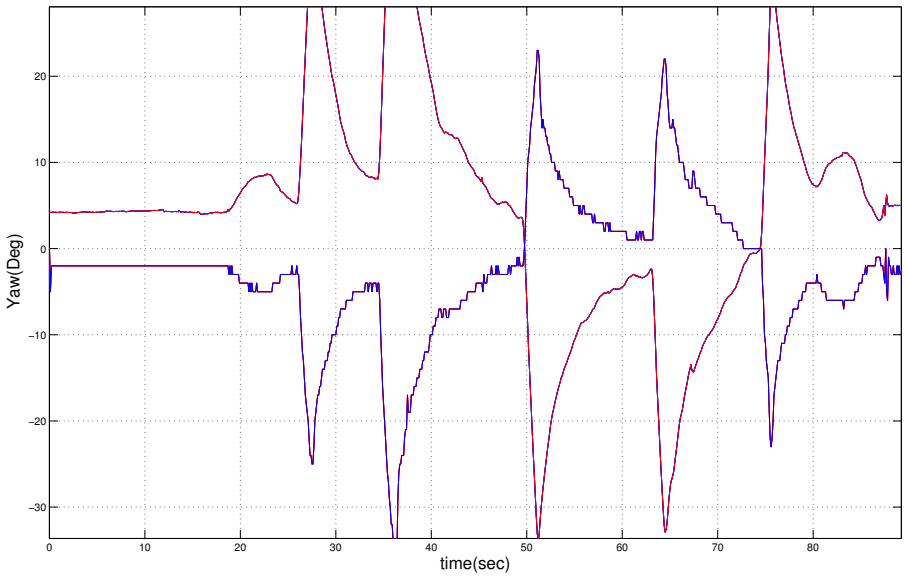



FIG. 21: Yaw angle and yaw command output using PID controller.

### K. Altitude control

In order to implement the height controller, it is a compulsory requirement to make sure safety. Once the vehicle gets a wrong command such as putting negative sign on the *unsigned char* variable, it flies to the sky at a full speed. In fact, this can be a very frightening moment. To avoid this kind of accident, the MK limits the FC code to prevent the user from controlling the gas value if the control cmd value is bigger than the stick gas. To achieve the altitude control, the FC code requires modification by enabling the external control with the saturation function that limits the range from 0 to 130. This number is the threshold of the gas value which lets the vehicle fly with the current payload and is obtained empirically.

### 1. PID altitude control

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{25}$$

where $e(t) = z^* - z$ and $z^*$ denotes the desired position along z axis and $z$ is the current position. In addition, the height measurement is coupled with $\theta$ and $\phi$. The projected estimation of the height can be determined as the following equation.

$$\hat{z} = \cos\theta \cos\phi \; z \tag{26}$$

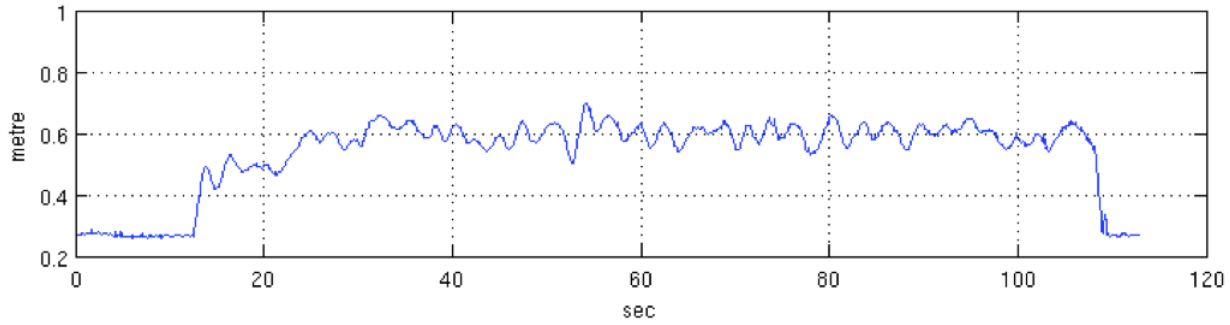



FIG. 22: Height measurement data using PID control with a goal at 0.6 metre. The input for the PID controller is the height measurement from laser hat and output is throttle control command. The MK uses **unsigned char** data type to control quadrotor range from ±127. Note that average height measurement is within ± 5cm. Rarely does maxim error goes up up to ± 10cm owing to ground effect and aerodynamics, but the quadrotor quickly converges into the goal.

## L. Flight time calculation

Flight time is essential to the MAV research area. Many research platforms can only fly for up to 10 minutes or less. However, the MK platform is able to fly up to 41 minutes unofficially. The video demonstration can be found on vimeo. Detailed information can be

found in the article of MikroKopter wiki page. In order to estimate the flight time given payload, we need to know total weight of a vehicle. In our case, the quadrotor weight is around ≈ 1,040g including frames, laser range finder, Gumstix, XBee, laser hat and battery. Therefore 260g is the thrust per motor.

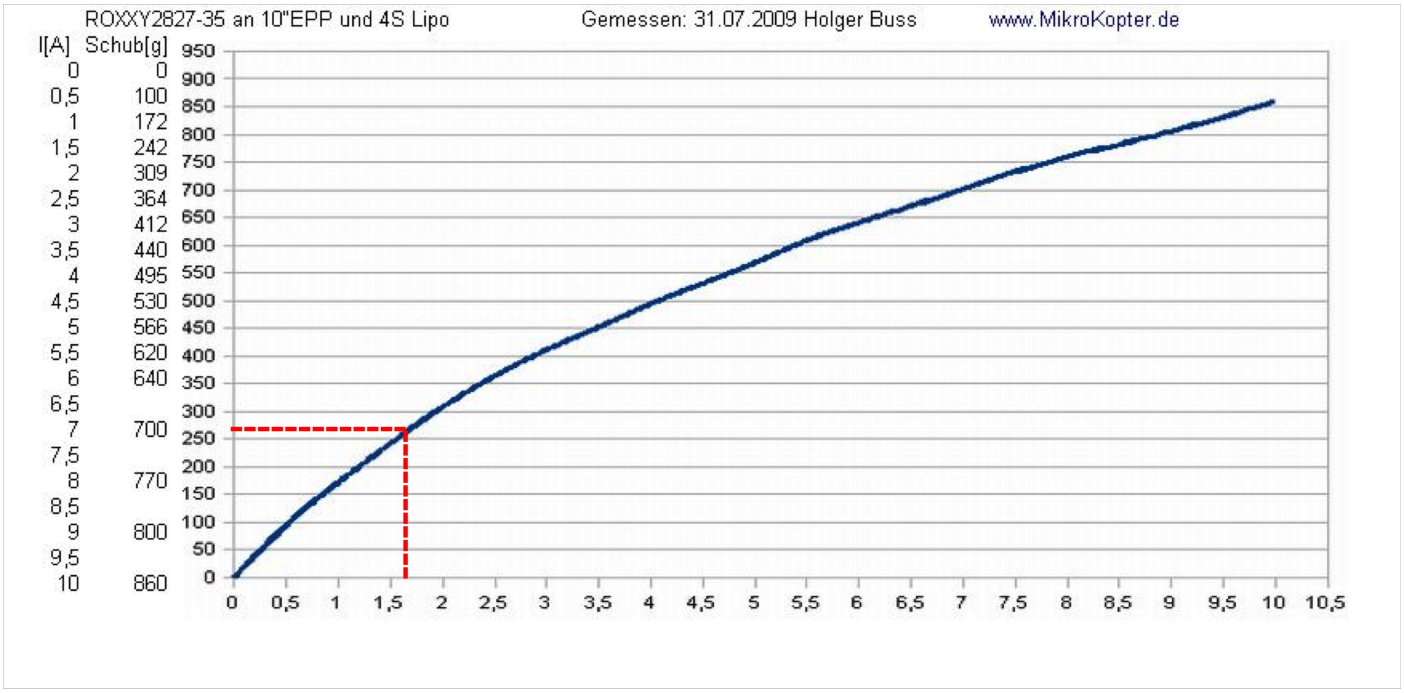



FIG. 23: The Ampere Versus payload characteristic of ROXXY2827-35 motor.

From figure 23, we could notice that a motor consumes around ≈ 1.7A. This implies that there is a total current of $6.8A = 1.7 \times 4$.

$$t = \frac{B}{C} \tag{27}$$

Where t is the total flight time in units of hour, B and C are battery capacity and Total current respectively. A Lipo battery, 2.2Ah, 4 cells, was used and Figure 27 shows 0.323 hour = 19.4 minutes flight time. This flight time is an ideal value that specifies the absolute limit. These factors should be taken account.

- The thrust characteristic was measured with a 100% loaded LiPo. The thrust decreases slightly when the voltage drops in flight.

- Turbulence
- Losses due to regulations
- Other losses (e.g. in cables, etc.)

For these reasons, a scale factor, 0.9, was multiplied. This parameter is chosen empirically and should be changed as the battery efficiency goes down.
In conclusion, the estimated total flight time is ***17.4 minutes*** and this value also can be found in Figure 24. This graph is validated by comparison of our previous work as described in Figure 25

Moreover, extra power consumption electric pars are used such as a laser range finder, a Gumstix, a XBee module and a USB HUB. These parts also consume energy and reduce the total flight time. The power consumption of the Gumstix is order of 2.5W $\sim$ 5W and the laser range finder spends 2.5W. Hence, the extra power consumption could be said to be less than 0.5A. Finally, from the calculation our quadrotor can fly up to ***16.28 minutes***, $0.9 \times \frac{2.2}{6.8+0.5}$ .

### M. Battery discharge dynamics

We logged Lipo batteries, (3cell, 2200mAh), to compensate the trust offset. Interestingly, each battery showed a different discharge curve. (see Figure 26).

### N. Platform price comparison.

The advantage of the MK is a competitive price. As described in Table VIII, it is a lot cheaper than a similar level of research platform. Although MK Basicset is needed to put all parts together before flight, there is detailed documents on the MK web. In addition, it is feasible to modify the platform for user's demands. The pelican from Asecnding Technology is ***6.41*** times more expensive than MK quadrotor.

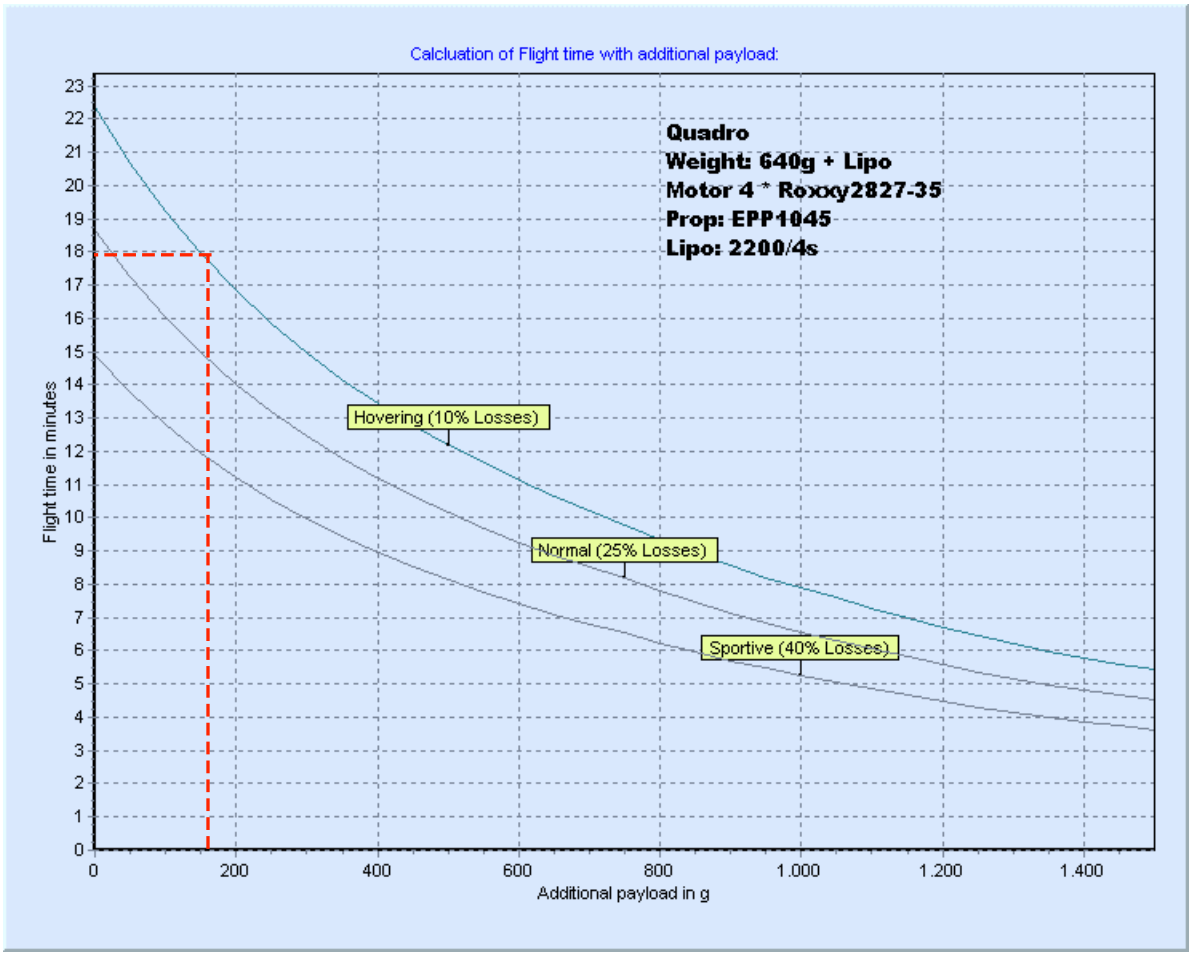



FIG. 24: Flight time Versus payload characteristic graph. Total weight of the vehicle is 1,040g and this graph shows the flight time with 640g+238g (battery) payload. Therefore, the payload is 162g. In addition, the hovering graph is appropriate for the indoor application.

## O. Precision comparison

In this section, a variety of accuracy and precisions of state estimations are presented when a quadrotor performs hovering or following given trajectory. As MAVs have fast dynamics, filtering techniques such as complementary filter and Kalman Filter are often used to estimate the state of the vehicle. Table IX shows a summary of precision from 5 different research groups.[2][5][1][3][4]

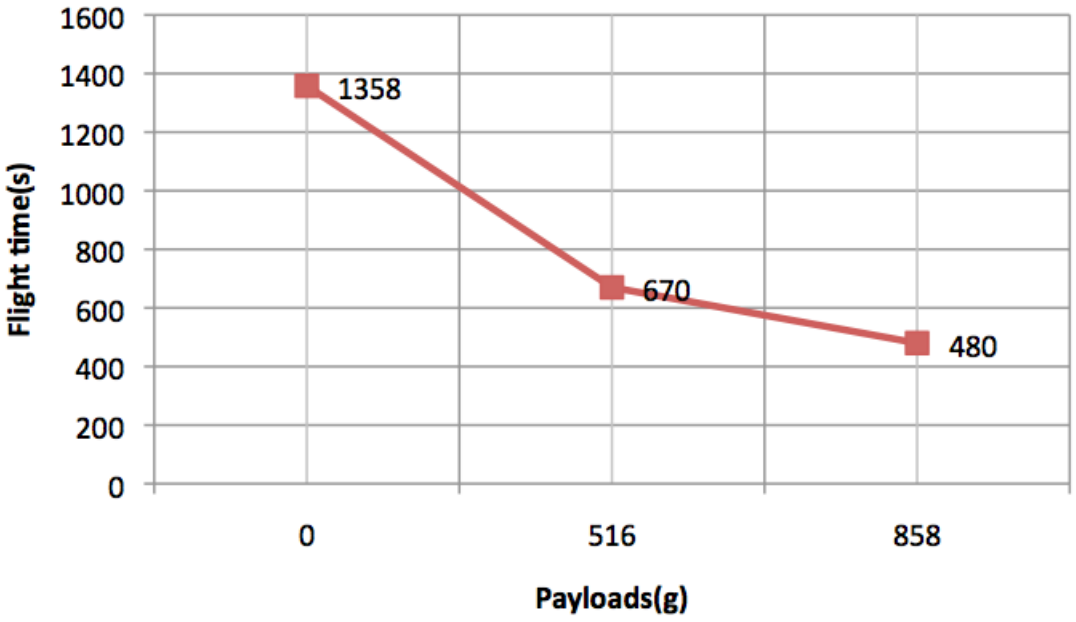



FIG. 25: Payload Versus Flight time table. This table was measured by us.

| | **Mikrokopter** | **Price(USD)** | **Asecnding Tech** | **Price(USD)** |
|---|---|---|---|---|
| platform | MK Basicset | 987.46 | Pelican | 7435.09 |
| Transmitter | 6CH | 169.99 | 7CH | 499.49 |
| Receiver | 7 CH | 79.99 | | |
| Total | | 1237.44 | | 7934.58 |
| Ratio | | | | 6.41 |

TABLE VIII: This table shows prices comparison between the MK Quadrotor set and the similar level platform of Ascending technology, Pelican. These prices come from MikroKopter , HobbyKing and Ascending Tech.

### P. Ground truth and estimation.

In this section, we compare ground truth from the VICON system to our dynamic state estimation. We estimate position $\hat{x}, \hat{y}, \hat{z}$ and velocity $\hat{V_x}, \hat{V_y}$. in addition, The MK estimates pitch($\theta$), roll($\phi$) and yaw($\psi$) angles.

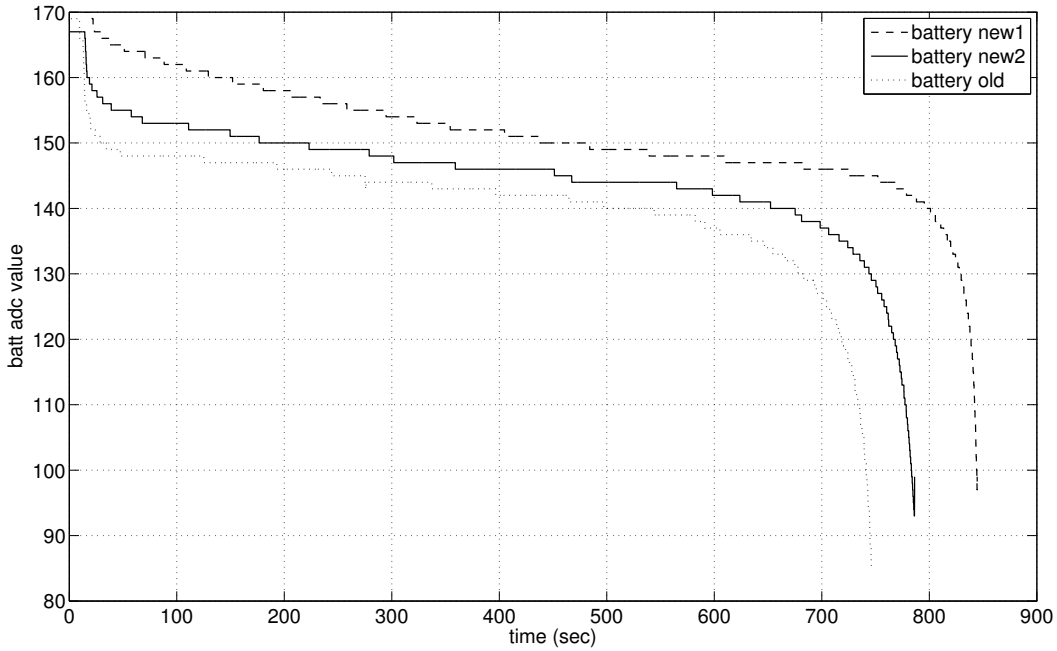



FIG. 26: Lipo battery discharge curves. X axis denotes time and y axis is a battery ADC reading value.

### *1. VICON coordinate system and experiment setup*

Figure 28 shows VICON coordinate and experiment configuration. Since the maximum range of the laser range finder attached to the quadrotor is 4m, we need to put structures within the 4m range. VICON system tracks infrared beam reflected by each makers and provides position and orientation of the makers. Figure 29 shows the place where makers on the quadrotor are attached. By computing error of Euclidean distance between two markers,Equation 28 VICON system shows ± 5mm position error. We could find the distance error increased after taking off because of frame vibration.

$$e = e_{initial} - \sqrt{(x_{maker1} - x_{maker2})^2 + (y_{maker1} - y_{maker2})^2 + (z_{maker1} - z_{maker2})^2} \qquad (28)$$

where $e_{initial}$ denotes the very first distance between markers.

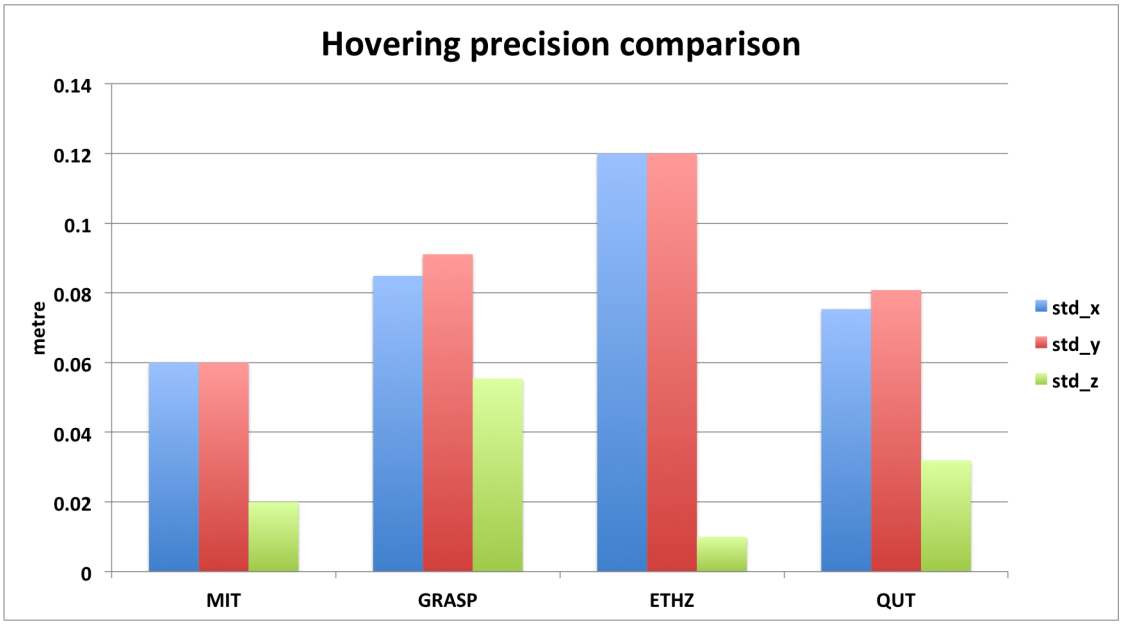



FIG. 27: Precision comparison of state-of-the-art platforms.

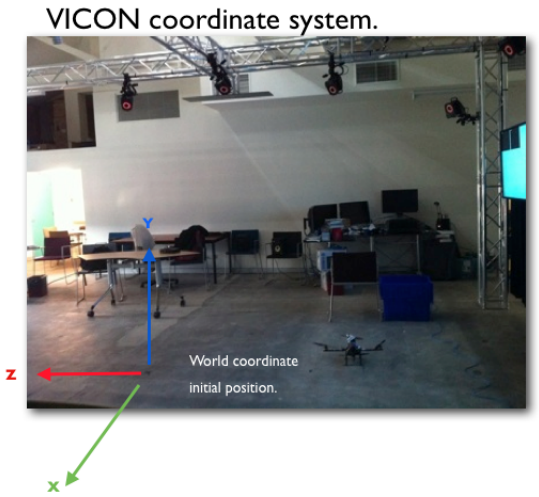



FIG. 28: This photo shows the VICON system coordinate and setup. This system is equipped with 14 cameras to track markers and provides position, normal and orthogonal. A "x" mark on the ground is the initial position of VICON system. The coordinate transformation is required from VICON coordinate to MK coordinate as described in 2.

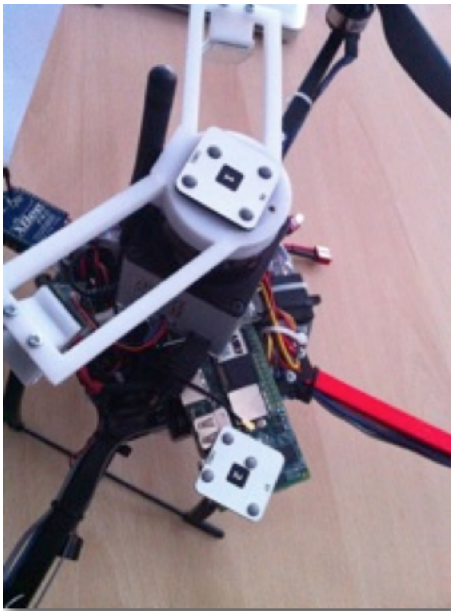


FIG. 29: Two makers are attached in order to calculate position error of VICON system.

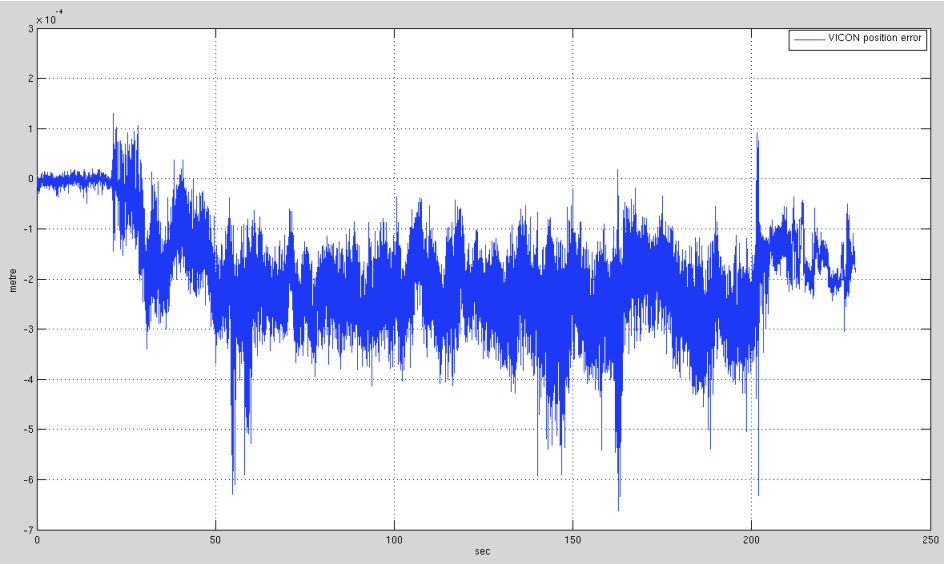



FIG. 30: Position error between two markers. Note that the quadrotor starts taking off and increasing position error due to frame vibration around 22sec. Mean is 0.0002 metre.

TABLE IX:

| **University** | **MIT(RANGE)**[2] | **GRASP Lab.**[5] | **ETHZ(ASL)**[1] |
|---|---|---|---|
| Year | 2010 | 2011 | 2011 |
| Precision | Hovering | Hovering | Hovering |
| (metre) | RMS_X= 0.06 | Std_X= 0.0849 | RMS_X=0.12 |
| (metre) | RMS_Y= 0.06 | Std_Y= 0.0911 | RMS_Y=0.12 |
| (metre) | RMS_Z= 0.02 | Std_Z= 0.0554 | RMS_Z=0.01 |
| | Trajectory following | Trajectory following | |
| (metre) | RMS_X=0.07 | Std_X=0.0247 | |
| (metre) | RMS_Y=0.07 | Std_Y=0.0323 | |
| (metre) | | Std_Z=0.007 | |
| Platform | Pelican | Pelican | Pelican |
| Sensor | Laser 40Hz | Laser 40Hz | 1000Hz IMU |
| | | | 10Hz mono Vision |
| | | | Fusion |
| **University** | **Freiburg**[3] | **Oxford(Chris Kemp)** [4] | **QUT** |
| Year | 2009 | 2006 | 2011 |
| Precision | | | Hovering |
| (metre) | | | Std_X=0.075368 |
| (metre) | | | Std_Y=0.080849 |
| (metre) | Error_Z=±0.1 | Error_Z=±0.1 | Std_Z=0.031913 |

*a. Position estimation* Figure 31 and Figure 32 show ground truth and position, velocity estimation.

## II. REFERENCES

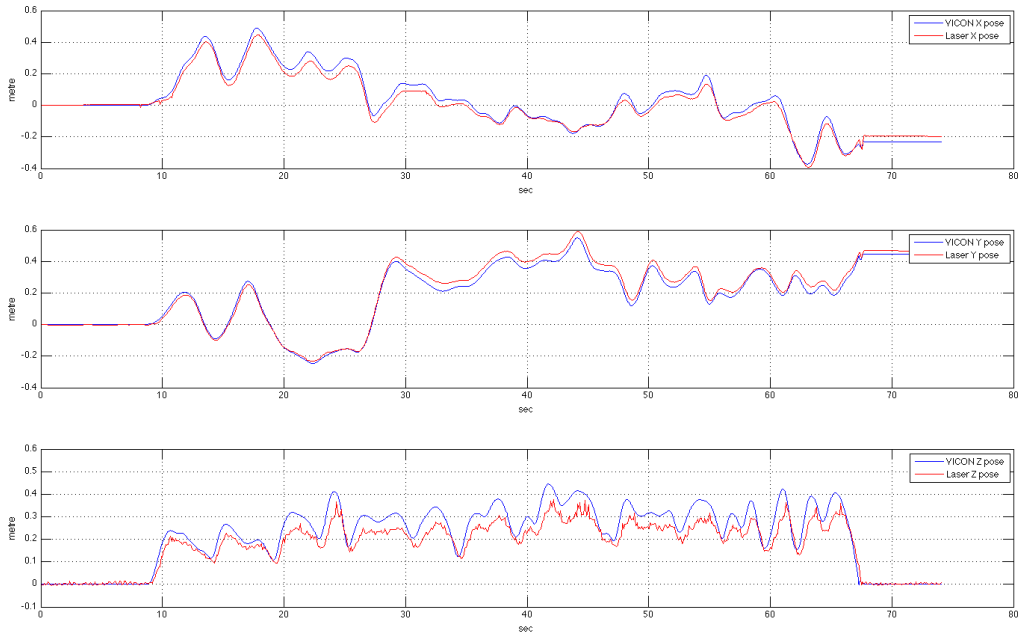



FIG. 31: Blue is the ground truth and red is the laser position estimation. The x-axis is time in second, the y-axis is position in unit of metre.

---

[1] Markus Achtelik, Michael Achtelik, Stephan Weiss, and Roland Siegwar. Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments. In ICRA2011, 2011.

[2] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. RANGE - robust autonomous navigation in GPS-denied environments. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 1096 –1097. MIT, may 2010.

[3] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pages 2878 –2883, may 2009.

[4] Christopher Kemp. Visual Control of a Miniature Quad-Rotor Helicopter. PhD thesis, Univer-

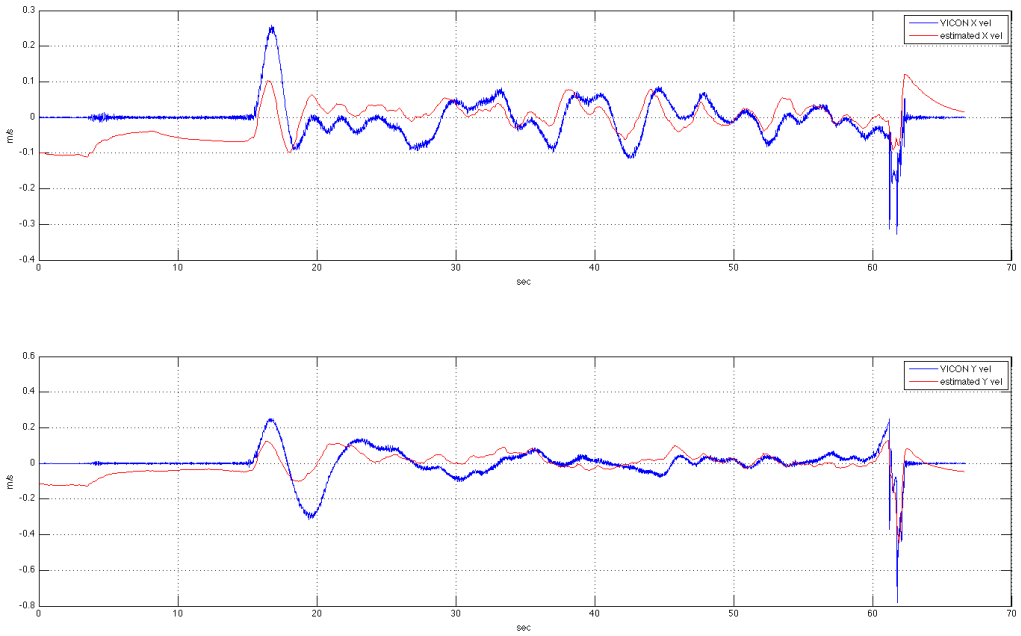



FIG. 32: Blue is the ground truth and red is the velocity estimation from complementary filter. The x-axis is time in second, the y-axis is velocity in unit of $m/s$.

sity of Cambridge, 2006.

[5] Shaojie Shen, Nathan Michael, and Vijay Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. In 2011 IEEE International Conference on Robotics and Automation, 2011.

## Appendix A: Matlab implementation

To implement a complementary filter, a simulation was conducted with MATLAB. The data that was used for simulation was obtained by manual pilot. All data was converted into SI units, metres and radians. The follow is the MATLAB code.

```
clear all;
data=importdata('data_Manual_Coordinate',',',1);

samp_time=0.05; % second/50ms sampling time
W_x=data.data(:,10)/1000;  % unit=metre X position
W_y=data.data(:,11)/1000;  % unit=metre Y position
```

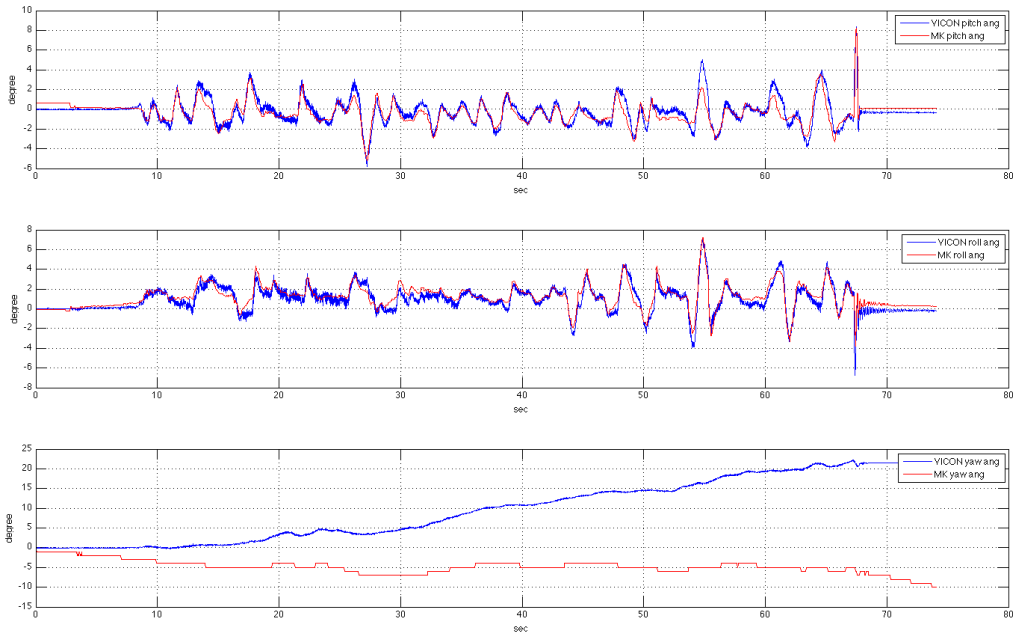



FIG. 33: Blue is ground truth and red is angle estimation by accumulating angular rates of gyro sensor. The x-axis is time in second, the y-axis is the angle in unit of degree. This data comes from the MK.

```
time=data.data(:,15)*samp_time; % unit=second
numData=max(data.data(:,15))+1;  % +1 because index starts from 0
Yaw=data.data(:,12)*pi/180; % radian
ACC_X=data.data(:,4); % unit=1g = 9.8m/s^2
ACC_Y=data.data(:,5); % unit=1g = 9.8m/s^2
MK_PITCH=data.data(:,1)/10*pi/180; % radian
MK_ROLL=data.data(:,2)/10*pi/180; % radian
stick_pitch=data.data(:,13);
stick_roll=data.data(:,14);
g=9.81; % unit 1g
%k=0.25; % filter gain
k=0.5; % filter gain
desired_v_x=0; % desired velocity along X axis unit=m/s
desired_v_y=0; % desired velocity along X axis unit=m/s
Kp=30;
Kd=30;



%Down sampling the x position data because it oversampled.
%The sampling rate of the laser scan matcher is 100ms and the log data is
```

```
%sampled at 50ms rate.

half_siz = numData/2;
W_x_100ms= zeros(half_siz,1);
W_y_100ms= zeros(half_siz,1);
for i=1 : half_siz
    W_x_100ms(i) = W_x(i*2-1);
    W_y_100ms(i) = W_y(i*2-1);
end

%Do interpolation to create smooth graph.
W_x_=interp(W_x_100ms,2);
W_y_=interp(W_y_100ms,2);

%Array size align with numData!
%It's not fancy but works.
W_x_mine= zeros(numData,1);
W_y_mine= zeros(numData,1);
for i=1 : numData-1
    W_x_mine(i) = W_x_(i);
    W_y_mine(i) = W_y_(i);
end


%W_x_mine(1659)=W_x_(1658);
%W_y_mine(1659)=W_y_(1658);


%W_v_x=diff(W_x)./diff(time)


%==========================================
% Step1. Calculate world frame velocity using laser position data
%==========================================
W_x_v = zeros(numData, 1);
W_y_v = zeros(numData, 1);
%for n=2:numData
    W_x_v=diff(W_x_mine)/samp_time;
    W_y_v=diff(W_y_mine)/samp_time;
 %   W_x_v(n)=(W_x_mine(n)-W_x_mine(n-1))/samp_time;
 %   W_y_v(n)=(W_y_mine(n)-W_y_mine(n-1))/samp_time;
    %W_y_v(n)=(W_y(n+1)-W_y(n))/samp_time;
%end

%==========================================
% Step2. Covert world frame velocity to the body frame usnig rotation
% matrix
%%==========================================

b_x_v = zeros(numData, 1);
b_y_v = zeros(numData, 1);
for n=1:numData-1
    b_x_v(n)=cos(Yaw(n))*W_x_v(n)-sin(Yaw(n))*W_y_v(n);
    b_y_v(n)=sin(Yaw(n))*W_x_v(n)+cos(Yaw(n))*W_y_v(n);
end

%==========================================
% Step3. velocity complementary filter
%%==========================================
```

```
89  b_v_hat_x = zeros(numData, 1);
    b_v_hat_y = zeros(numData, 1);
91  ddotX = zeros(numData, 1);
    ddotY = zeros(numData, 1);
93  b_v_hat_x(1)=0;
    b_v_hat_y(1)=0;
95
    for n=1:numData-1
97      ddotX(n) = -(ACC_X(n)-g*sin(MK_PITCH(n)))/cos(MK_PITCH(n));
        ddotY(n) = -(ACC_Y(n)-g*sin(MK_ROLL(n)))/cos(MK_ROLL(n));
99      b_v_hat_x(n+1) = b_v_hat_x(n) + samp_time*(k*(b_x_v(n)-b_v_hat_x(n))+ddotX(n));
        b_v_hat_y(n+1) = b_v_hat_y(n) + samp_time*(k*(b_y_v(n)-b_v_hat_y(n))+ddotY(n));
101 end


103
    %=================================================
105 % Step5. Velocity P controller
    %%================================================
107 pitch_cmd = zeros(numData, 1);
    roll_cmd = zeros(numData, 1);
109 error_pitch = zeros(numData, 1);
    error_roll = zeros(numData, 1);
111 post_error_pitch = zeros(numData, 1);
    post_error_roll = zeros(numData, 1);
113
    error_pitch(1)=0;
115 error_roll(1)=0;

117 for n=2:numData
        error_pitch(n)=desired_v_x - b_v_hat_x(n);
119     error_roll(n)=desired_v_y - b_v_hat_y(n);

121     %pitch_cmd(n) = Kp*error_pitch(n)+Kd*(error_pitch(n)-error_pitch(n-1));
        pitch_cmd(n) = Kp*error_pitch(n);
123     %roll_cmd(n) = Kp*error_roll(n)+Kd*((error_roll(n)-error_roll(n-1))/samp_time);
        roll_cmd(n) = Kp*error_roll(n);
125 end

127 %=================================================
    % Step6. Acceleration integration
129 %%================================================

131 %for n=1:numData-1
    %    vel_int_x(n)=samp_time/2*(cumsum(ACC_X(n+1))+cumsum(ACC_X(n)))*9.8;
133 %end

135 vel_int_x = cumsum(detrend(ddotX)*samp_time);

137 pos_int_x = cumsum(detrend(vel_int_x)*samp_time);

139




141




143
    %=================================================
145 % Step4. Plotting
    %%================================================
```

```
147
    %Create a multiline label for the x-axis using a multiline cell array:
149 %xlabel({'first line';'second line'})
    %Create a bold label for the y-axis that contains a single quote:
151 %
    %ylabel('George''s Popularity','fontsize',12,'fontweight','b')
153


155 %=========================================
    % Fig1. laser world position vs acc world position
157 %%=========================================
    figure(1);
159 plot(time,W_x_mine,'r');
    xlabel('time [sec]', 'fontsize',12),ylabel('world position x [m]', 'fontsize',12);
161 hold on;

163 plot(time,pos_int_x,'b');
    xlabel('time [sec]', 'fontsize',12),ylabel('world position x from acc_x', 'fontsize',12);
165
    legend('laser world position x','acc world position x');
167 %figure(2);
    %plot(time,W_x_v,'b');
169


171 %figure(2);
    %plot(time,b_x_v,'m-');
173 %xlabel('time [sec]', 'fontsize',12),ylabel('body velocity x [m/s]', 'fontsize',12);

175 %figure(4);
    %plot(time,b_y_v);
177
    %=========================================
179 % Fig2. body velocity vs CF estimated velocity
    %%=========================================
181
    figure(2);
183 plot(time,b_v_hat_x,'r');
    xlabel('time [sec]', 'fontsize',12),ylabel('velocity x [m/s]', 'fontsize',12);
185
    hold on;
187 plot(time,b_x_v,'b');

189 legend('estimated velocity x','body velocity x');

191 axis([0 time(end) -0.6 0.6]);
    %hold on;
193 %plot(time,vel_int_x,'g');

195 %=========================================
    % Fig3. pitch input vs stick pitch
197 %%=========================================

199 figure(3);
    plot(time,pitch_cmd,'r');
201 xlabel('time [sec]', 'fontsize',12),ylabel('pitch cmd vs stick pitch', 'fontsize',12);
    hold on;
203 plot(time,stick_pitch,'b');

205 legend('pitch cmd','stick pitch');
```

```
%figure(7);
%plot(time,roll_cmd,'r');
%xlabel('time [sec]','fontsize',12),ylabel('roll cmd vs stick roll','fontsize',12);
%hold on;
%plot(time,stick_roll,'b');

%figure(8);
%plot(time,b_v_hat_y,'r');
%xlabel('time [sec]','fontsize',12),ylabel('velocity y [m/s]','fontsize',12);

%hold on;
%plot(time,b_y_v,'b');

%legend('estimated velocity y','body velocity y');
```