Technical documentation for AR.Drone project

Inkyu, Hu and Van

Cyphy Lab.
Queensland University of Technology

January 14, 2013

Contents

1 Introduction

This document addresses the procedure how to develop an autonomous fly system based on off-the-shelf AR.Drone.

2 System Overview

2.1 AR.Drone platform

2.2 Initial ARDrone coordinates

This document describes our ardrone coordinate systems: a IMU and a camera.

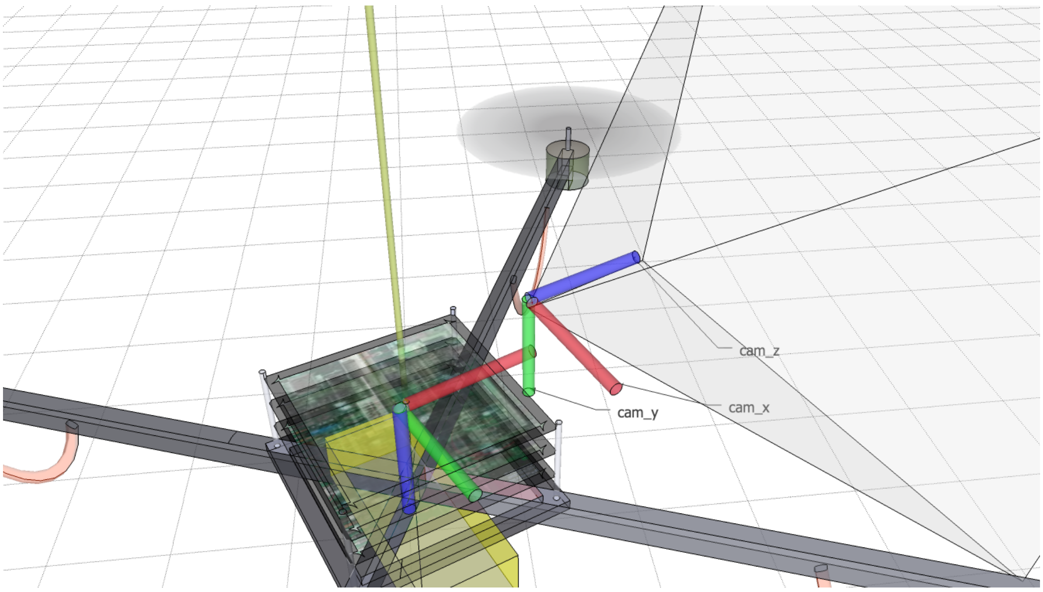


Figure 1: Close look. IMU and camera coordinate definitions. Red, green and blue denote x,y,z respectively.

2.3 Navdata

navdata is a fundamental data structure which contains Euler angles, altitude, accelerations, linear velocity[1]. This data can be obtained up to 100 Hz.

[1]ROS Header message definition. `http://www.ros.org/doc/api/roslib/html/msg/Header.html`

Table 1: navdata structure definition with SI unit.

navdata	**type**	**Description**	**Unit**
batteryPercent	float32	0 to 100	%
rotX	float32	left/right tilt	$^\circ$
rotY	float32	forward/backward tilt	$^\circ$
rotZ	float32	orientation,yaw	$^\circ$
altd	float32	estimated altitude	m
vx	float32	linear x velocity	m/s
vy	float32	linear y velocity	m/s
vz	float32	linear z velocity	m/s
accx	float32	body x acceleration	m/s^2
accy	float32	body y acceleration	m/s^2
accz	float32	body z acceleration	m/s^2
gyrox	float32	angle rate about x axis	$^\circ/s$
gyroy	float32	angle rate about y axis	$^\circ/s$
gyroz	float32	angle rate about z axis	$^\circ/s$
tm	float32	Time stamp from ardrone	sec
header	Header	ROS header[1]	

2.4 ROS topic name

For ROS, we have to define topic names and namespace. Figure2 shows brief description of ROS topic messaging map. More detail also can be found in the repository named "`ardrone_navi/src/ours`".

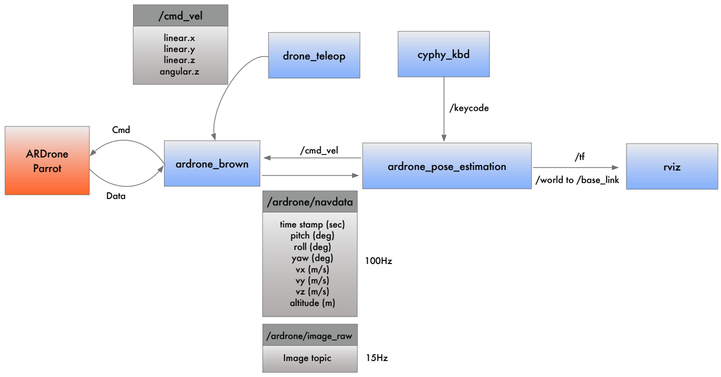


Figure 2: Blue boxes denote ROS nodes the orange box is a ardrone platform. Gray boxes present both ROS topic name and data structure. Generally a prefix "/" stands for ROS topic.

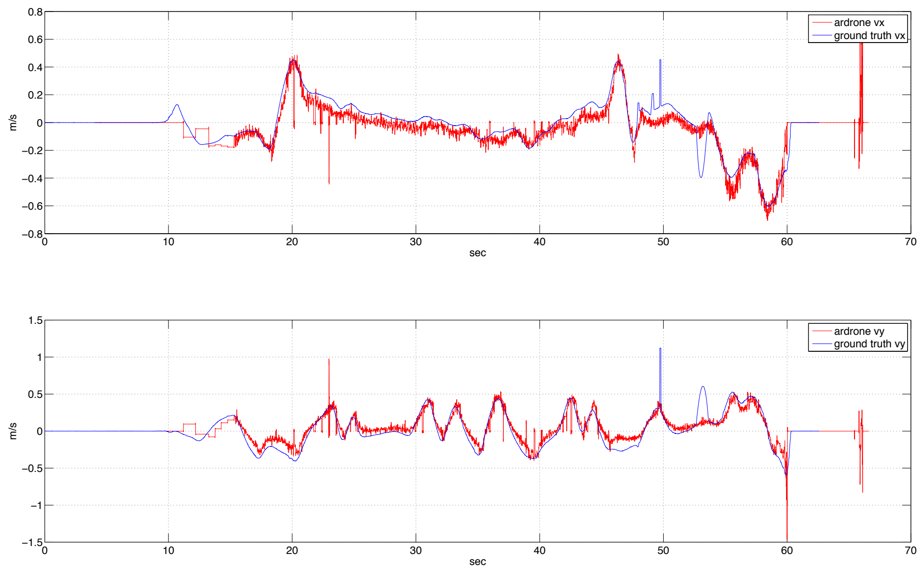


Figure 3: Blue denotes ground truth velocity obtained from VICON and red is AR.Drone estimated lateral x and y velocity respectively.

3 VICON comparison

In this section ground truthed comparison is presented. We compared lateral body velocity estimation from AR.Drone to velocity obtained from sub-millimetre accuracy VICON system. In addition roll-pitch-yaw angle estimation comparison is presented.

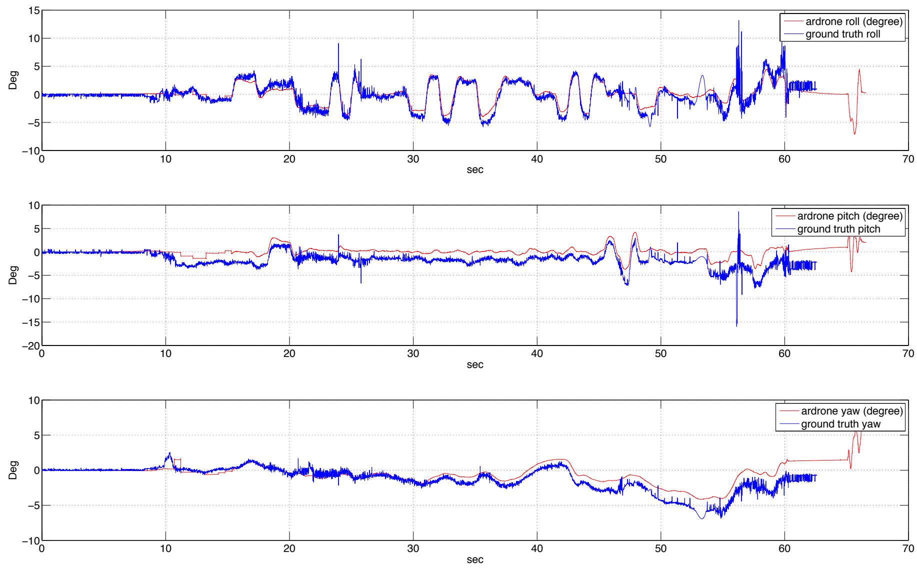


Figure 4: Blue denotes ground truth angles obtained from VICON and red is AR.Drone estimated angles, roll, pitch and yaw respectively.

4 Software building instruction

In this section, how to build ardrone_brown, ardrone_pose_estimation and ethzasl_ptam packages will be described step by step. Before getting started, we define the repository sync local folder as:

```
/home/enddl22/Workspace/ardrine_side_project
```

and ROS working space path is

```
/home/enddl22/Workspace/fuerte
```

Note that this path should be different for each machine.

4.1 Building ardrone_autonomy package

Compared to previous ardrone_brown package, ardrone_autonomy package extended the driver compatible with AR.Drone 2.0 and also increase the image sampling rates.

In order to build ardrone_autonomy package on your local machine, you should checkout the package from our repository located in `https://enddl22@bitbucket.org/enddl22/ardrone_side_project.git`. Once you are under the folder named ardrone_autonomy, the driver should be built with following commands.

```
        ./build_sdk.sh
```

```
        mkdir build
        cd build
        cmake ..
        make
```

Now you should see the ardrone_driver appearing under subfolder **/bin**.

4.2 Running ardrone_autonomy driver

By running this package we are able to access all data of AR.Drone. Open a new console and type

```
roscore
```

Searching SSID "ardrone_015526" from your WiFi searching lists and this SSID should be different for each ardrone. After establishing connection, type

```
ifconfig
```

You should able to see "192.168.1.x" for your WiFi adaptor. Go to ardrone_brown/bin and run ros ardrone driver node.

```
cd /home/enddl22/Workspace/fuerte/ardrone_brown/bin
./ardrone_driver
```

Checking all data is being published using ros topic command.

```
rostopic list -v
```

You should be able to see /ardrone/navdata on the publishing lists.

```
rostopic echo /ardrone/navdata
```

You should be able to see data, rotX, rotY, rotZ, accX etc on the console. Further if you want to plot all data from AR.Drone, type

```
roslaunch ardrone_brown rxplot_data.launch
```

You should be able to see Fig. 5.

4.3 Building ethzasl_ptam package

To avoid corruption the local repository, let's make a working copy to workspace.

```
cp -a ./ardrone_side_project/src/ours/ethzasl_ptam /
    home/enddl22/Workspace/fuerte/
cd /home/enddl22/Workspace/fuerte/ethzasl_ptam
rosmake ethzasl_ptam
```

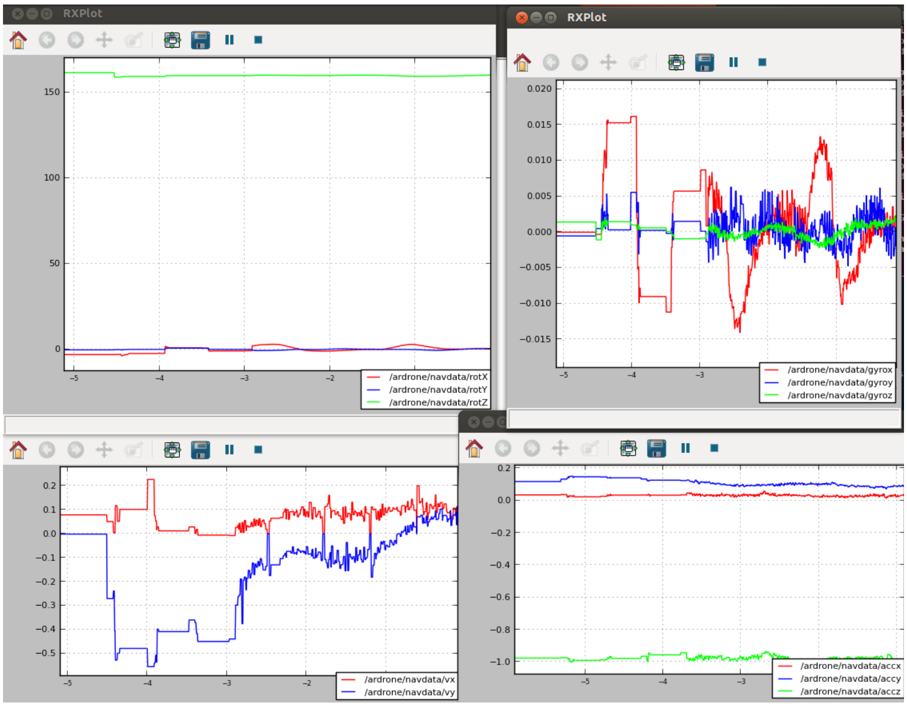


Figure 5: Plotting all data using rxplot tool.

You should be able to see the following log message

```
...
[ rosmake ] Built 25 packages with 0 failures.
[ rosmake ] Summary output to directory
```

Now we are at

```
pwd /home/enddl22/Workspace/fuerte/ethzasl_ptam
```

Let's check all executable binaries are generated correctly.

```
cd ptam/bin
```

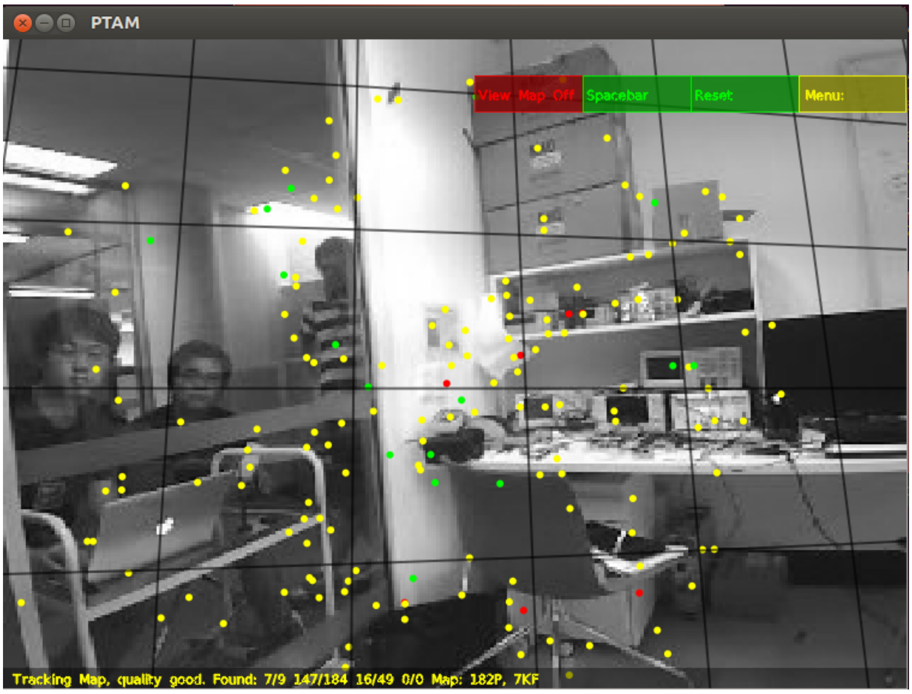
You should be able to see build, bin folders and also can see executable files from bin folder inside.

4.4 Running ethzasl_ptam package

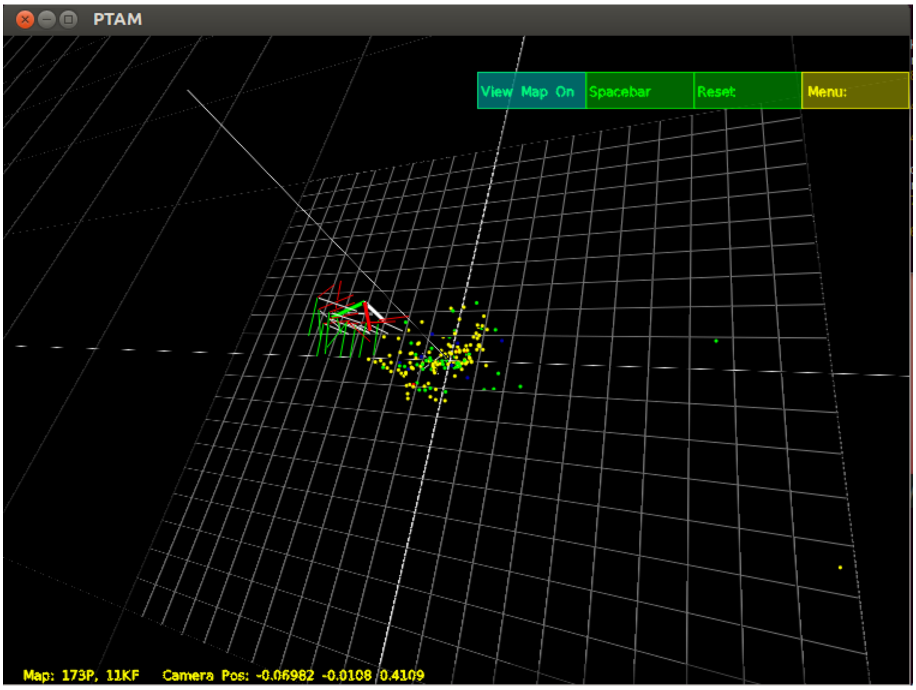
We need open 4 terminals for roscore, ardrone_brown ros ardrone driver, image_proc to covert RGB image into gray scale and ptam. Running the following commands for each terminal window.

```
roscore
ardrone_brone (dir path=ardrone_brown/bin)
ROS_NAMESPACE=ardrone rosrun image_proc image_proc (dir path=image_pipeline/image_proc/bin)
roslaunch ./ptam.launch (dir path ethzasl_ptam/ptam/launch)
```

You should be able to see Fig.6.



(a) Image overlaying window with tracking features and the reference frame.



(b) 3D Map and 6 degree of freedom camera position estimation visualisation view

Figure 6: PTAM

4.5 Building ardrone_pose_estimation package

As mentioned before, let's make a working copy to workspace.

```
cp -a ./ardrone_side_project/src/ours/ardrone_pose_estimation /home/enddl22/Workspace/fuerte/
cd /home/enddl22/Workspace/fuerte/ardrone_pose_estimation
rosmake ardrone_pose_estimation
```

You should be able to see the following log message

```
...
[ rosmake ] Results:
[ rosmake ] Cleaned 20 packages.
[ rosmake ] Built 20 packages with 0 failures.
[ rosmake ] Summary output to directory
```

You should be able to see build, bin folders and also can see executable files from bin folder inside.

4.6 Running ardrone_pose_estimation package

Make sure ardrone_brown driver running and publishing navdata topic by typing

```
rostopic list -v
```

You should be able to see the following message

```
Published topics:
 * /ardrone/navdata [ardrone_brown/Navdata] 1 publisher
 * /ardrone/image_raw [sensor_msgs/Image] 1 publisher
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher
 * /rosout [rosgraph_msgs/Log] 1 publisher
 * /clock [rosgraph_msgs/Clock] 1 publisher
```

Launching the ardrone_pose_estimation node.

```
roslaunch ardrone_pose_estimation ardrone_navi_trj.launch
```

Note that if you are publishing "navdata" through rosbag file then you need to modify source code. Since ardrone_pose_estimation will be started position estimation 4 second later after receiving taking off command. This 4 second was picked experimentally and is the time when AR.Drone goes to hovering state.

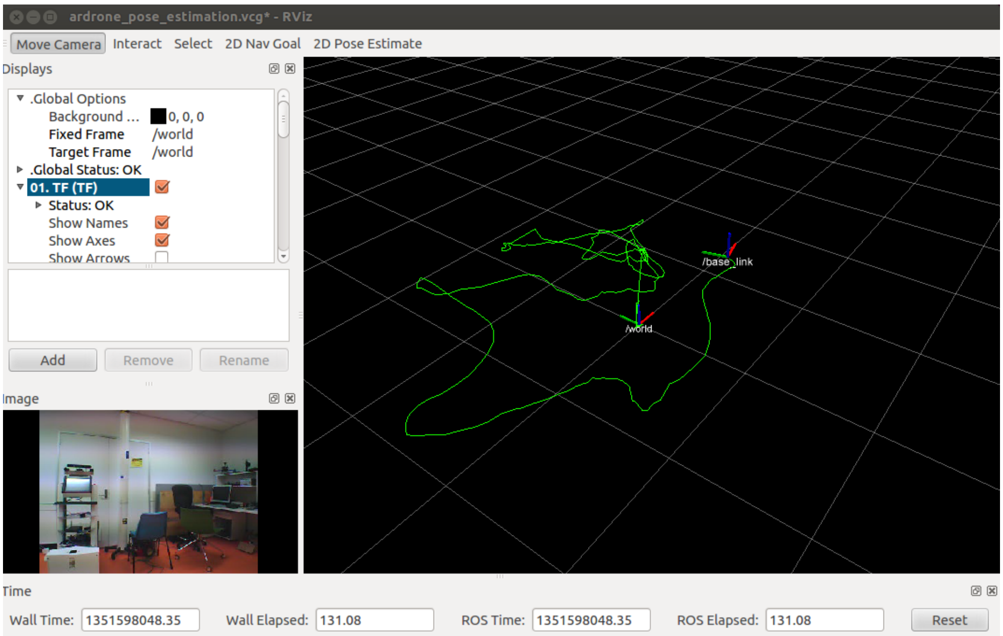


Figure 7: Position estimation visualisation using rviz.

Modify ardrone_pose_estimation/src/ardrone_pose_estimation.cpp file line number 338

```
if( now.toSec() - takeoff_start.toSec() >4 &&
    pick_takeoff_time)
```

to

```
if(1)
```

then rebuild the package

```
rosmake ardrone_pose_estimation
```

Then you should be able to see Fig. 7. Note that rviz is installed properly on your system otherwise need to build rviz by typing

```
rosmake rviz
```

You should be able to see the following message.

```
[rosmake-1] Finished <<< rviz [PASS] [ 413.15 seconds ]
    -- WARNING: 857 compiler warnings
[ rosmake ] Results:
[ rosmake ] Cleaned 33 packages.
[ rosmake ] Built 33 packages with 0 failures.
```

```
[ rosmake ] Summary output to directory
```

5 Kalman Filter for states estimation

In this section, we will describe only states, process model and measurement model. More detail and well-documented Kalman filter can be found from wikipedia.[2]

5.1 States

States are expressed w.r.t world coordinate.

$$\mathbf{X}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T \tag{1}$$

5.2 Process model

Linear constant-velocity process model is

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where δ is sample rate of Kalman Filter, 160 Hz.

5.3 Measurement model

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{X}_k \tag{4}$$

where $\mathbf{z}_k$ is 4×1 position and velocity measurement vector. Position is obtained from PTAM with scale estimation at 15 Hz and velocity comes from AR.Drone onboard at 160Hz. We choose $\mathbf{Q}$ and $\mathbf{R}$ matrix as

$$\begin{aligned} \mathbf{Q} =& \mathrm{diag}\begin{bmatrix} 0.1\,\mathrm{m} & 0.1\,\mathrm{m} & 0.3\,\mathrm{m/s} & 0.3\,\mathrm{m/s} \end{bmatrix}^2 \\ \mathbf{R} =& \mathrm{diag}\begin{bmatrix} 0.1\,\mathrm{m} & 0.1\,\mathrm{m} & 0.05\,\mathrm{m/s} & 0.05\,\mathrm{m/s} \end{bmatrix}^2 \end{aligned}$$

[2] wikipedia Kalman Filter. `http://en.wikipedia.org/wiki/Kalman_filter`

5.4 Prediction

States prediction may be represented

$$\hat{\mathbf{X}}_k^- = \mathbf{F}\hat{\mathbf{X}}_{k-1} \tag{5}$$

Covariance prediction may be represented

$$\hat{\mathbf{P}}_k^- = \mathbf{F}\hat{\mathbf{P}}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{6}$$

$\hat{\mathbf{P}}_k^-$ is predicted covariance and $\hat{\mathbf{P}}_{k-1}$ is estimated covariance at $k-1$.

5.5 Update

Compute Kalman gain.

$$\begin{aligned} \mathbf{S} &= \mathbf{H}\hat{\mathbf{P}}_k^-\mathbf{H}^T + \mathbf{R} \\ \mathbf{K} &= \hat{\mathbf{P}}_k^-\mathbf{H}\mathbf{S}^{-1} \end{aligned}$$

Compute innovation matrix.

$$\mathbf{N} = \mathbf{z} - \mathbf{H}\hat{\mathbf{X}}_k^-$$

States and covariance update

$$\begin{aligned} \hat{\mathbf{X}}_k &= \hat{\mathbf{X}}_k^- + \mathbf{K}\mathbf{N} \\ \hat{\mathbf{P}}_k &= \hat{\mathbf{P}}_k^- - \mathbf{K}\mathbf{H}\hat{\mathbf{P}}_k^- \end{aligned}$$

6 Scale Estimation

6.1 Scale calibration

6.2 Online scale estimation

6.3 Experimental validation

7 Control

7.1 System Identification

For x,y,z,yaw models.

7.2 Controller design

Something about PIDs.

7.3 Performance evaluation

Simulated model output VS real model output.

8 Experimental results

Hovering and way points following.