AR Drone Setup with ROS and Sensor Data Fusion using AR Drone's Accelerometer and Gyroscope

Welcome

Lab 5

Dr. Ahmad Kamal Nasir

Today's Objectives

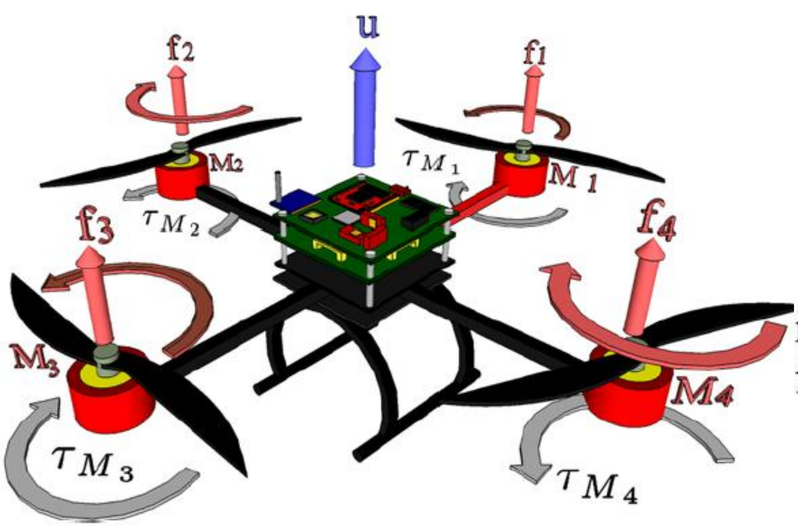
- Introduction to AR-Drone
  - Hardware
  - Communication
- AR-Drone Interface with ROS
  - ROS driver nodes
  - Teleop Keyboard/Joystick
- ROS with Quadrotor Gazebo Model
  - Orientation estimation
  - Setting up pose estimation node based on EKF

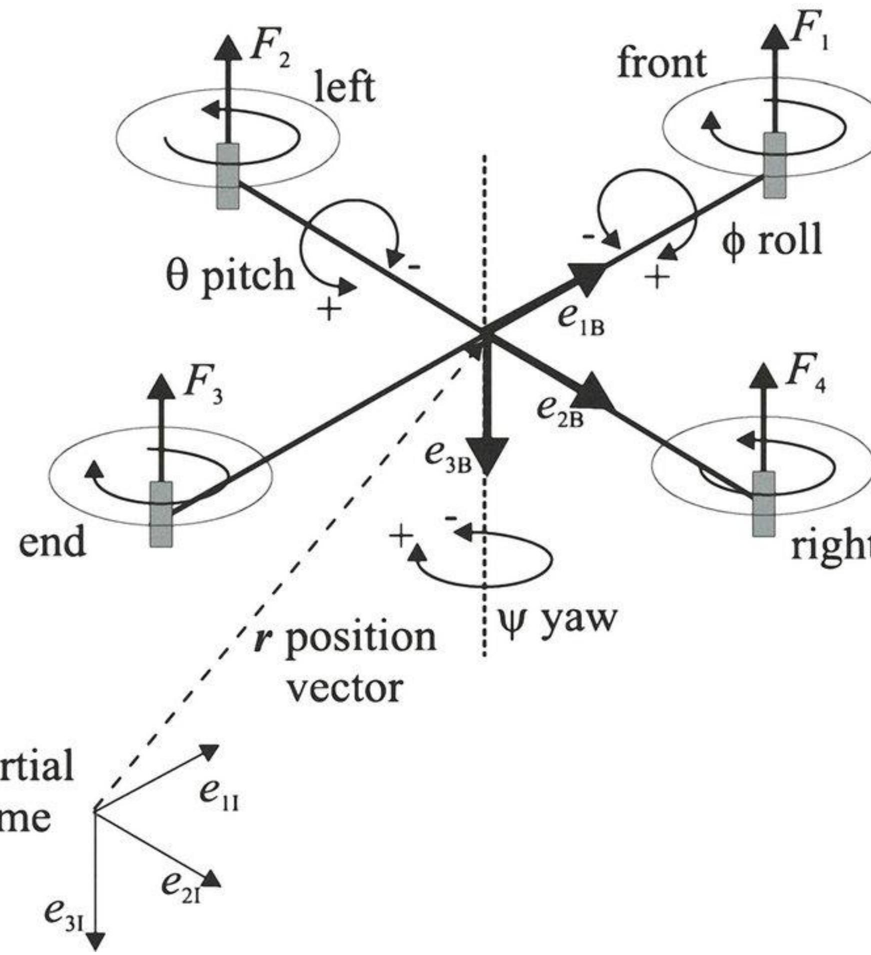
Quad-rotor

- Quad-rotors were introduced 14 years before helicopters but due to control problems were not able to make the way.
- French company parrot SA Introduced $300 device at International Consumer Electric Show in Las Vegas 2010-12
- Expanded poly propylene body, 380grams(outdoor), 420grams(indoor)

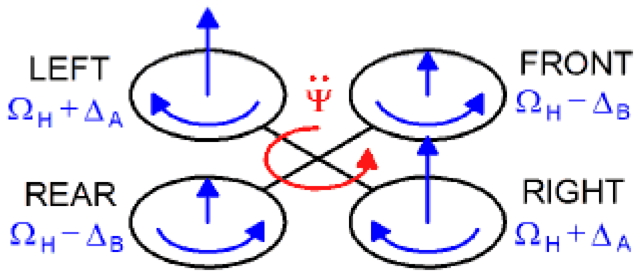
Flight Basics

- Unstable system, therefore, require feedback for stability

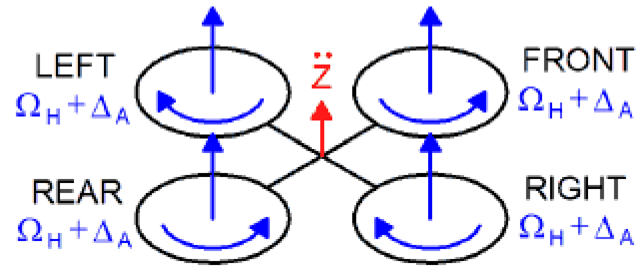




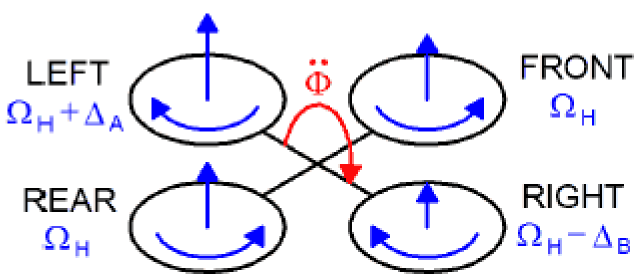
Flight Control



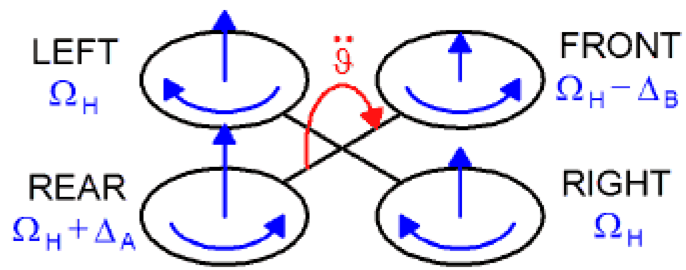
Yaw (Rotate)
CW/CCW



Throttle
Up/Down



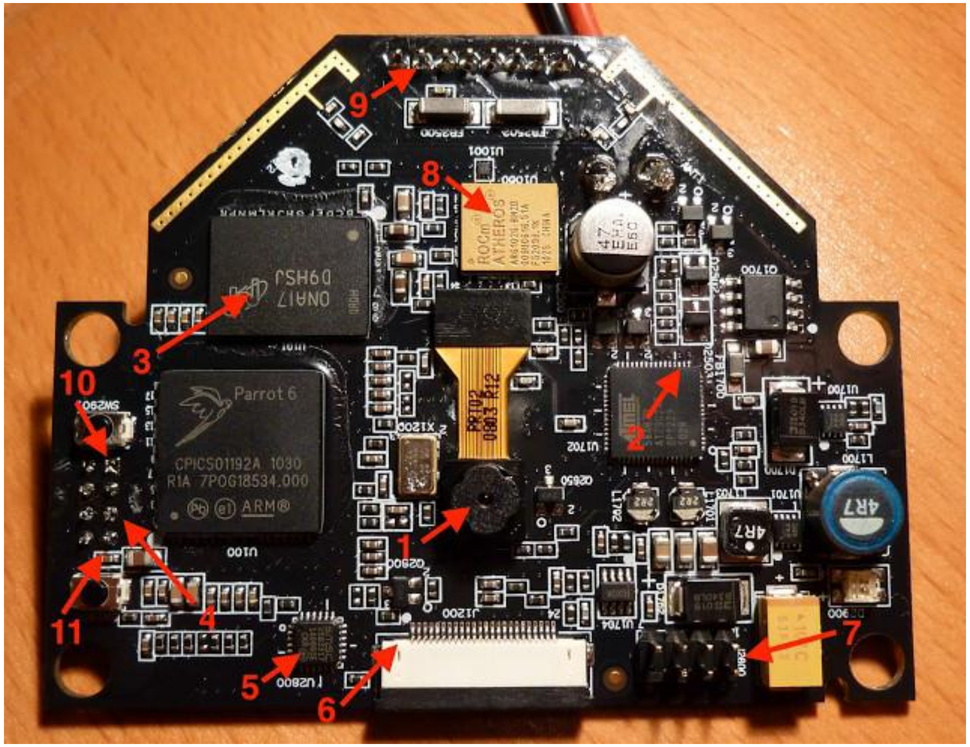
Roll
Left/Right



Pitch
Forward/Backward

AR-Drone Hardware

- **CPU** (ARM Cortex A8,OMAP 3630) @ 1GHz, **GPU** (PowerVR SGX530) @ 800 MHz,1GB **RAM**, 128MB **ROM**
- 2 **Webcams**
- 1 **WiFi**
- 1 **Ultrasonic**
- 1 **Barometer**
- 1 **9DOF IMU**
- 1 USB Port (GPS and LTE Modem)
- 4 brushless motors @ 28500 RPM, 14.5W, 1:8.75 Gear Ratio, with control board (ATMEGA8L)
- Up **to 5m/sec, 13 mins** of continuous flight
- 1000mAh, 11.1V LiPo batteries (Discharge capacity 15C, 80grams) voltage decreases from full charge (12.5 Volts) to low charge (9 Volts)

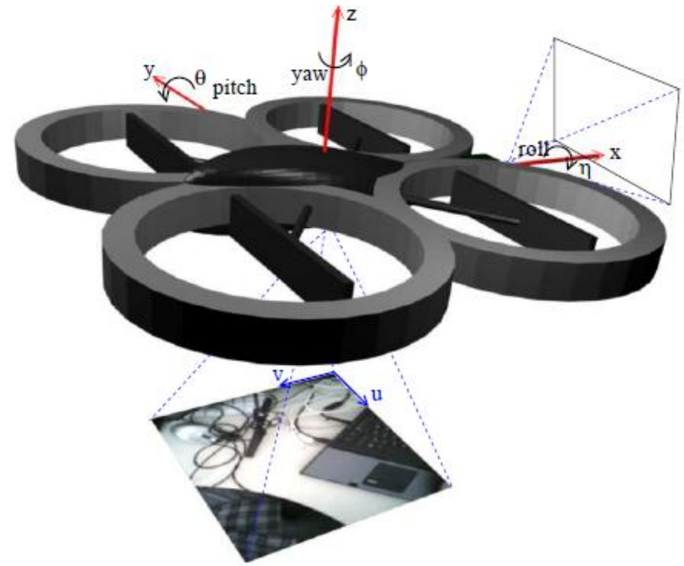


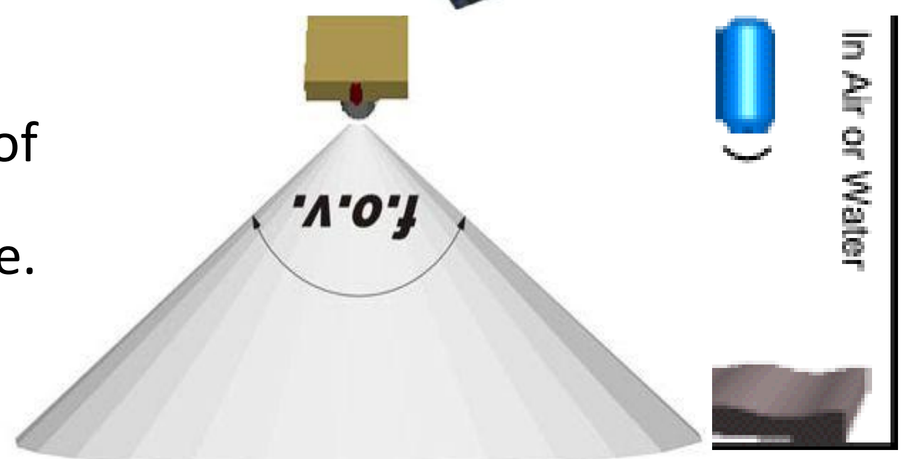
Onboard Processing Power

- Busy Box based GNU/**Linux** distribution with 2.6.27 Kernel
- It is possible to **cross-compile** an application for the ARM processor and run it directly on the AR-Drone control board.
- In this case, one can access the drone cameras and onboard sensors directly without a delay caused by the wireless data transfer. Thus, one can achieve **faster control loops** and experiment with a low level control of the drone.
- The AR-Drone runs Linux on-board. The AR-Drone is running a **telnet** as well as an FTP daemon . The Telnet daemon will allow login as root (no password e.g telnet 192.168.1.1).
- The cameras are exposed as standard video4linux2 devices (/dev/video0 and /dev/video1)
- The navigation board, which handles accelerometer, gyrometer, and sonar sensors, is exposed as a **serial port** (/dev/ttyPA2) according /dev/ttyPA0 for USB serial port and /dev/PA1 for motor controllers
- **DroneGames**, which took place over the weekend in San Francisco, tasked programmers with hacking the UAVs in the most interesting and creative ways possible. { echo "reboot"; sleep 1 } | telnet 192.168.1.1

AR-Drone Vision System

- Vertical camera **63fov @60fps**, 240p for horizontal speed measurements. Front cam **93fov@30fps**,720p.
- Ultrasound sensors has maximum range of **6m**. Barometric sensor ($\pm$10pa)for higher altitudes. It determines the vertical displacement of the vehicle. Vertical scene depth in the image.
- The received video can be from either of the two cameras or a **picture-in-picture** video with one of the camera images superposed on the top left corner of the other one.

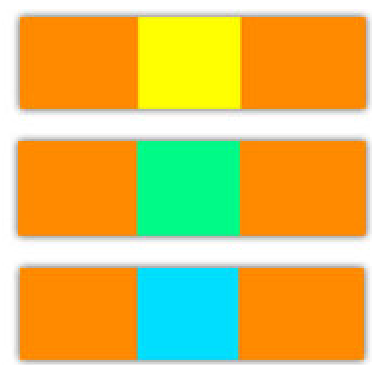




AR-Drone Vision System (Cont.)

- AR-Drone can run a simple analysis of the images from the frontal camera and search for a specially designed tags in the images.

- In the case the tags are detected, the **navdata** contains estimates of their positions.

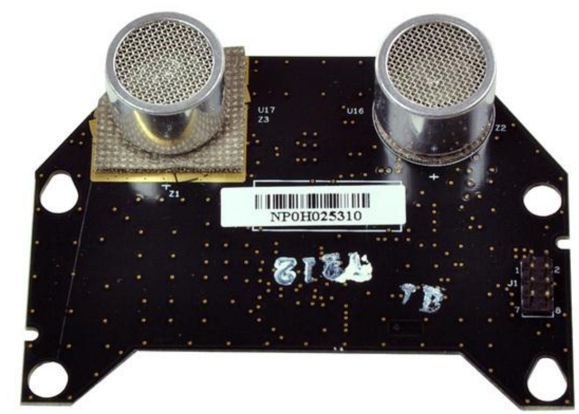


On-board Velocity Sensor - Vision

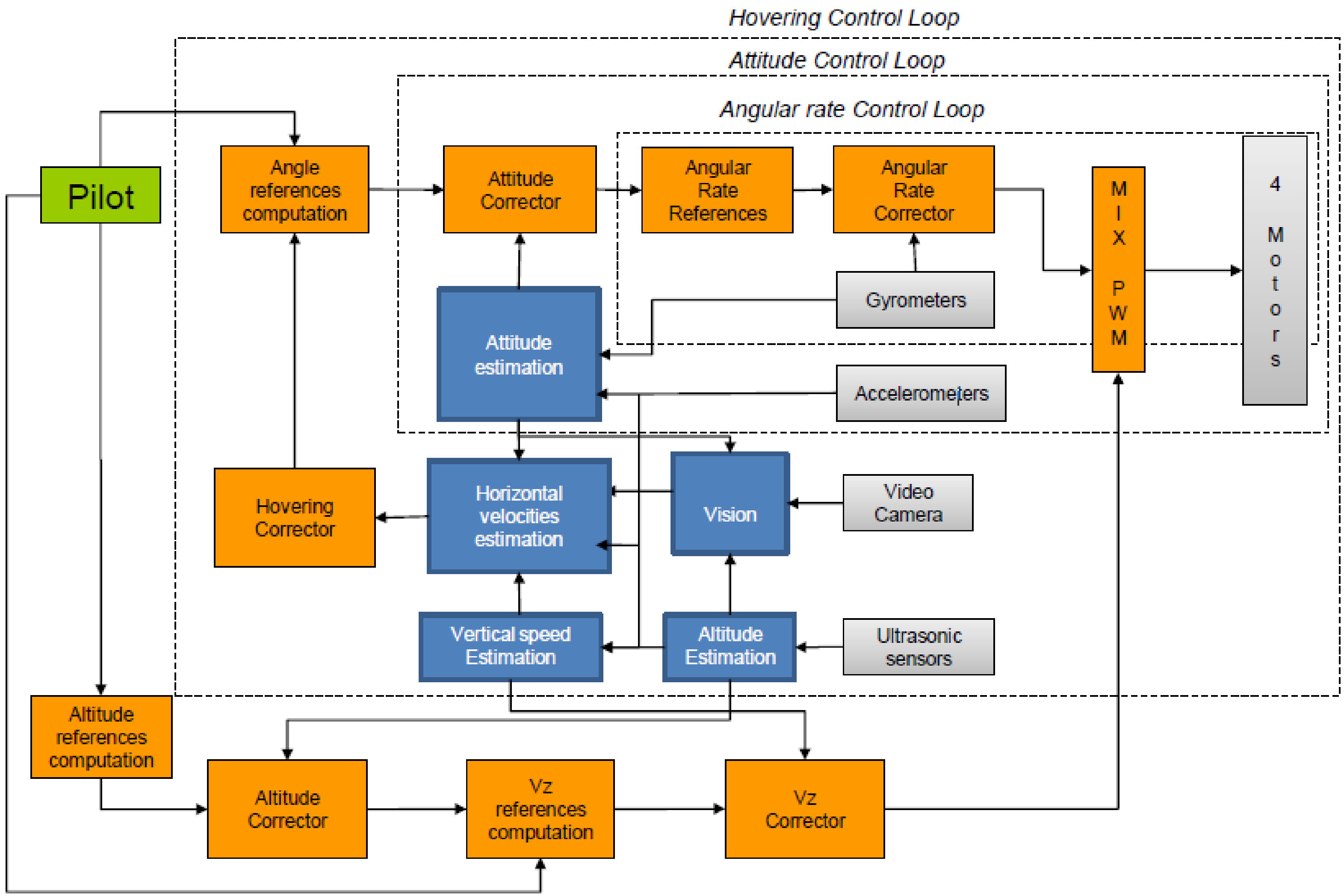
- To achieve a stable hovering and position control, the AR-Drone estimates its horizontal velocity using its vertical camera.
- Two different algorithms are used to estimate the horizontal velocity.
  - One tracks local interest points **(FAST corners)** over different frames and calculates the velocity from the displacement of these points. It provides a more accurate estimate of the velocity and is used when the vehicle's speed is low and there is enough texture in the picture.
  - The second algorithm estimates the horizontal speed by computing the **optical flow** on pyramidal images. It is the default algorithm during flight. It is less precise but more robust since it does not rely on highly textured or high-contrast scenes.
- The AR-Drone uses inertial information from its IMU for estimating the state of the vehicle. It fuses the **IMU** data with information from the **vision algorithms** and an **aerodynamics model** to estimate the velocity of the vehicle

AR-Drone Navigation System

- Navigation board contains **3axis accelerometer**(±50mg), 2axis gyro(2000deg/sec), precise yaw gyro(XB-3500CV, **drift 12deg/min** dynamic, **4deg/min** static). Data is processed at 200Hz.3 Axis magnetometer(±6deg).
- The navigation board uses a **16bits dsPIC24h** micro-controller running at 40 MHz, and serves as an interface with the sensors. These sensors are a 3-axis accelerometers, a 2-axis gyroscope, a 1-axis vertical gyroscope, and a ultrasonic Sensors (**200Hz**). The PIC micro-controller handles the ultrasonic transmitter, **25Hz**



On-board Control Algorithm



Communication Ports

- WIFI network with ESSID:ardrone_xxx. 192.168.1.1. clients request IP from DHCP server.
- AR-Drone sends two types of streams
  - Controlling and configuring the drone is done by sending AT commands on UDP port 5556.
  - navdata, are sent by the drone to its client on UDP port 5554. 15HZ in demo mode and 200Hz in full(debug)
- A video stream is sent by the AR-Drone to the client device on port 5555
- A fourth communication channel, called control port, can be established on TCP port 5559 to transfer critical data

Port	Explanation
21 (TCP)	FTP Server which serves video and image files recorded by the drone
23 (TCP)	Telnet Server offering a root shell
5551 (TCP)	FTP access to the update folder for the purpose of firmware updates
5553 (TCP)	VIDEO: The H264-720p frames of the camera are available here if the phone application is recording
5554 (UDP)	NAVDATA: Current telemetry data (status, speed, rotor speed) is sent to the client here (15 cmds/s demomode, 200 cmds/s full/debug mode).
5555 (TCP)	VIDEO: The video stream of the drone is available to clients here
5556 (UDP)	ATCMD: The drone is controlled in the form of AT commands. These control commands are sent periodically to the drone (30 cmds/s).
5559 (TCP)	CONTROL port: Some critical data, such as configurations are transferred here.

Communication Protocol

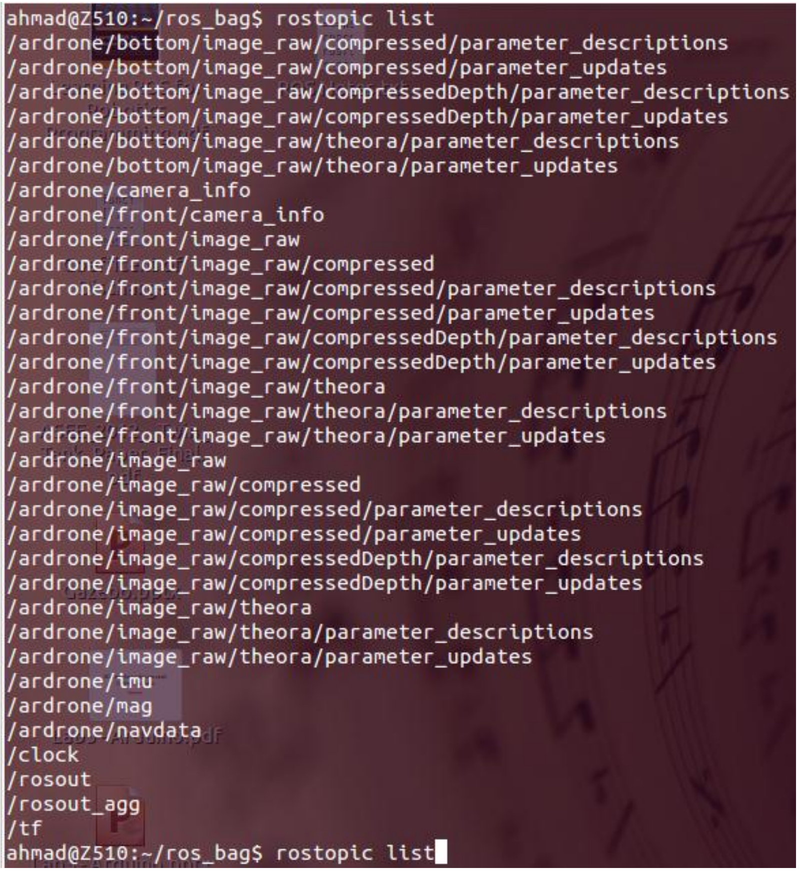
- AT commands are text strings sent to the drone to control its actions.
- AT*PCMD=<sequence>,<enable>,<pitch>,<roll>,<gaz>,<yaw>

AT command	Arguments[1]	Description
AT*REF	input	Takeoff/Landing/Emergency stop command
AT*PCMD	flag, roll, pitch, gaz, yaw	Move the drone
AT*PCMD_MAG	flag, roll, pitch, gaz, yaw, psi, psi accuracy	Move the drone (with Absolute Control support)
AT*FTRIM	-	Sets the reference for the horizontal plane (must be on ground)
AT*CONFIG	key, value	Configuration of the AR.Drone 2.0
AT*CONFIG_IDS	session, user, application ids	Identifiers for AT*CONFIG commands
AT*COMWDG	-	Reset the communication watchdog
AT*CALIB	device number	Ask the drone to calibrate the magnetometer (must be flying)

Android GUI

AR-Drone with ROS

- Install ardrone_autonomy packages found at
  - **sudo apt-get install ros-indigo-ardrone_autonomy**
- Use the following command to launch the quadrotor ROS driver, make sure wireless connection between AR-Drone and Computer is already established
  - **rosrun ardrone_autonomy ardrone_driver _realtime_navdata:=False _navdata_demo:=0**



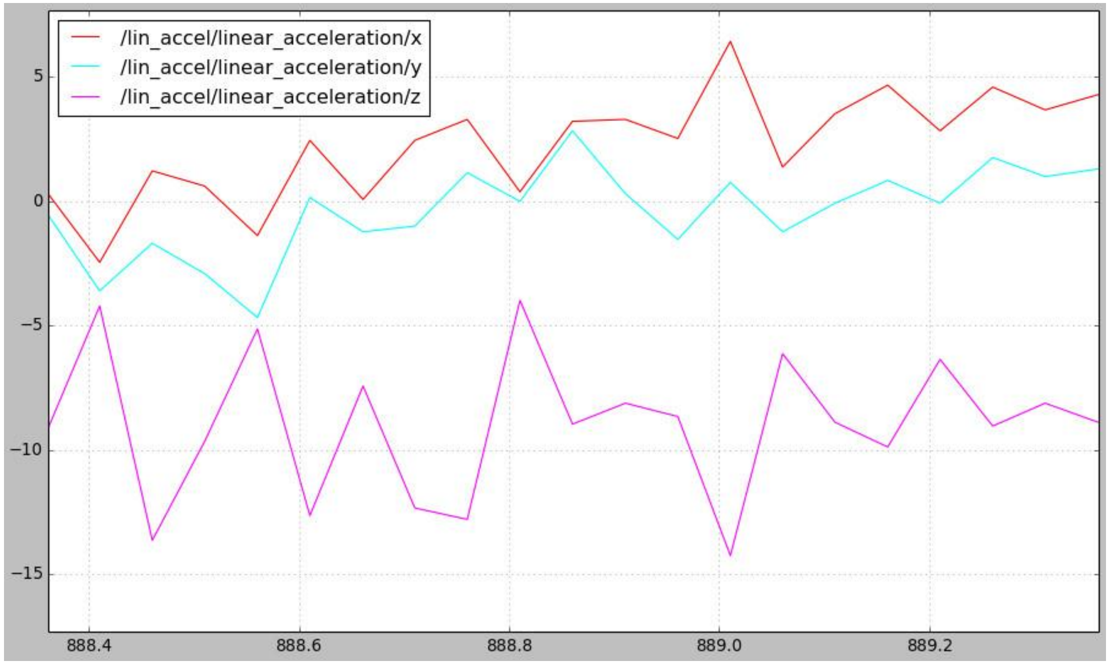
/navdata ROS Topic

- Use ardrone/navdata topic to acquire sensor information such as orientation, linear and angular velocity

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]	

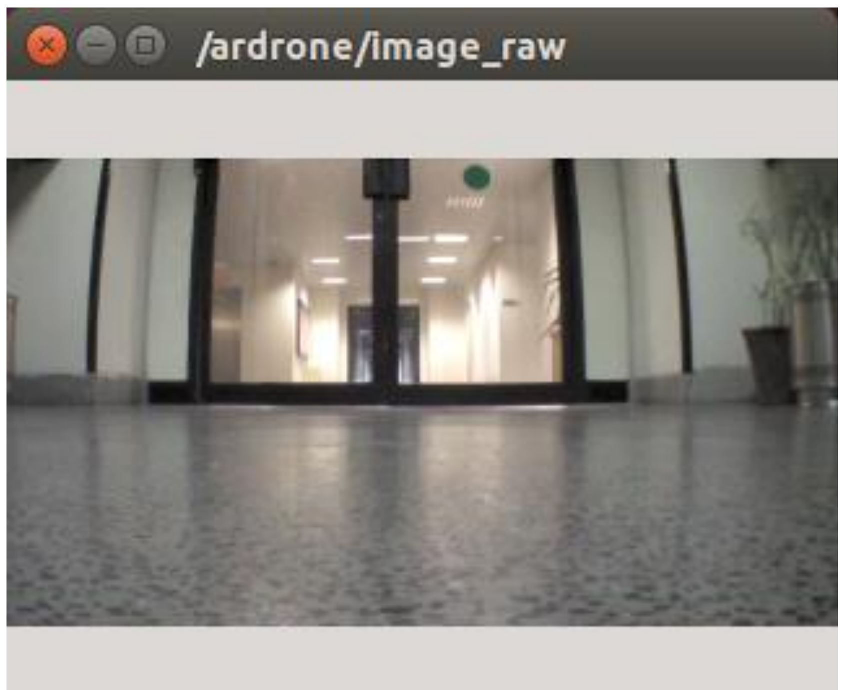
Plot real-time data

- Use ardrone/imu topic to acquire raw IMU sensor information, use following command to view a live plot
  - **rqt_plot /imu/linear_acceleration/x:y:z**



AR-Drone Camera

- To view the live camera stream
  rosrun img_view img_view /img_view:=/ardrone/front/image_raw



AR-Drone teleop

- AR-Drone can be controlled either by using a joystick or by a keyboard. In both cases geometry_msgs/Twist message must be published to cmd_vel topic.
  - -**linear.x**: move backward, **+linear.x**: move forward
  - **-linear.y**: move right, **+linear.y**: move left
  - **-linear.z**: move down, **+linear.z**: move up
  - **-angular.z**: turn left, **+angular.z**: turn right
- Value range: **-1.0 to +1.0**

AR-Drone teleop (Cont.)

- rostopic pub -1 std_msgs/Empty /ardrone/takeoff
- rostopic pub -1 std_msgs/Empty /ardrone/land
- rostopic pub -1 std_msgs/Empty /ardrone/reset
- Download (LMS) and run following node for controlling quadrotor using keyboard
  - **roslaunch ardrone_tutorials keyboard_controller.launch**
- To navigate the AR-Drone using joypad
  - **roslaunch ardrone_tutorials joystick_controller.launch**

Angles from Gyro-Rate/Accelerometer Sensors

- **Gyro-rate sensors:** Angles from body rate
  - $$\begin{bmatrix}\phi_t\\ \theta_t\\ \psi_t\end{bmatrix} = \begin{bmatrix}\phi_{t-1} + \dot{\phi}_t \cdot \Delta t\\ \theta_{t-1} + \dot{\theta}_t \cdot \Delta t\\ \psi_{t-1} + \dot{\psi}_t \cdot \Delta t\end{bmatrix}$$
- **Accelerometer sensors:** Angles from gravity vector

$$A_b = C_i^b(\theta,\phi,\psi) \cdot A_i = R_x(\phi) \cdot R_y(\theta) \cdot R_z(\psi) \cdot A_i$$

$$\begin{bmatrix} a_x \cos(\theta) + a_y \sin(\phi)\sin(\theta) + a_z\cos(\phi)\sin(\theta) \\ a_y \cos(\phi) - a_z\sin(\phi) \\ -a_x \sin(\theta) + a_y \sin(\phi)\cos(\theta) + a_z\cos(\phi)\cos(\theta) \end{bmatrix} = \begin{bmatrix}0\\0\\g\end{bmatrix}$$

$$\begin{bmatrix}\phi\\ \theta\end{bmatrix} = \begin{bmatrix} atan2(ay, ax) \\ -atan2\left(ax, \sqrt{a_y^2 + a_z^2}\right)\end{bmatrix}$$

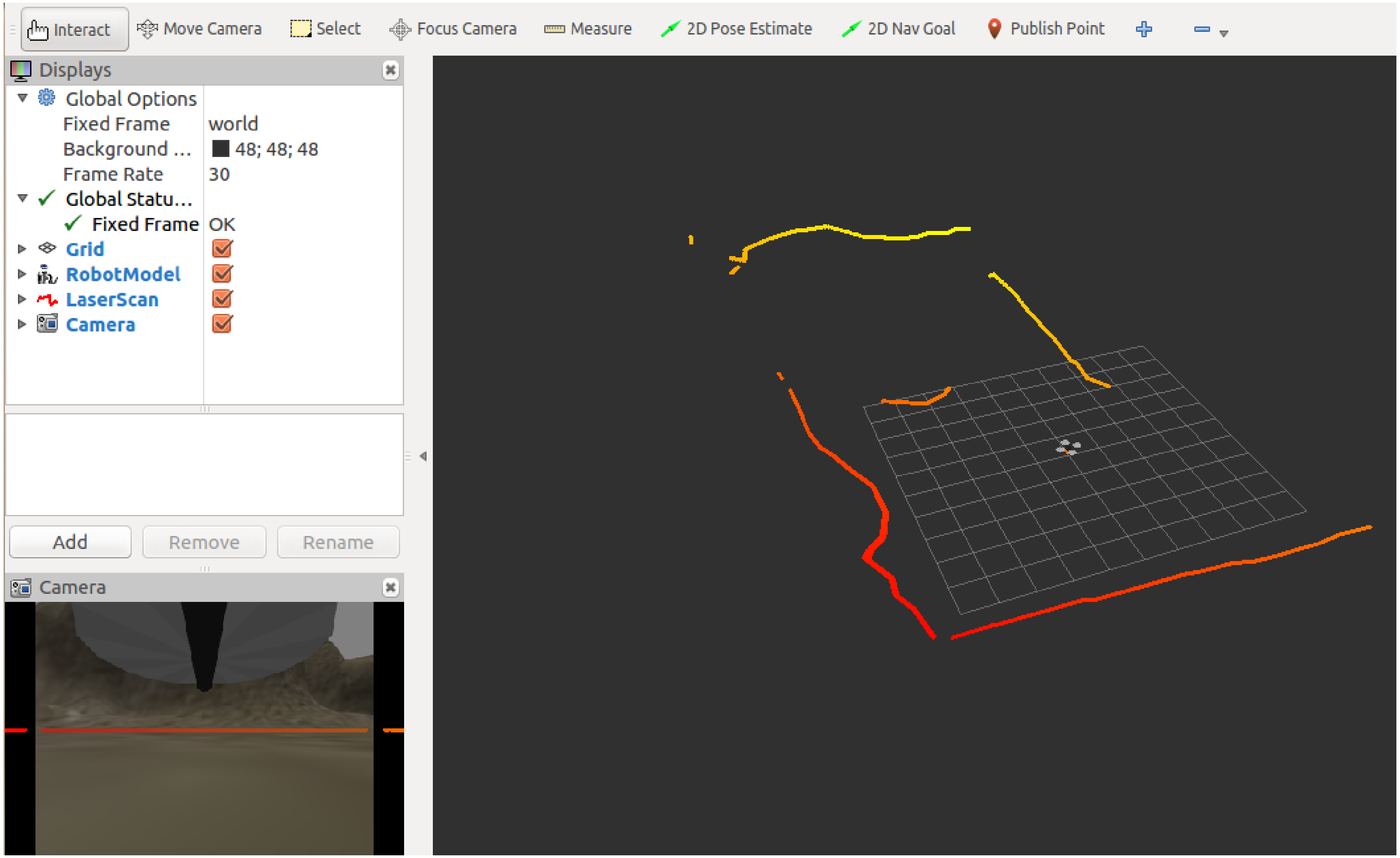
Task1: Hardware Experiment

- Install ardrone_autonomy packages
- launch the quadrotor ROS driver, make sure wireless connection between AR-Drone and Computer is already established
- Plot real-time navdata: [rotX,rotY,rotZ]
- Visulize live video stream
- Teleop the AR-Drone using keyboard/joypad
- Create a rosbag of the real experiment
- Estimate roll, pitch, yaw angles from gyroscope
- Estimate roll, pitch angles from accelerometer

Quadrotor model with Gazebo

- To install the quadrotor gazebo simulation model
  - **sudo apt-get install ros-indigo-hector-quadrotor***

Quadrotor topics visualization in Rviz



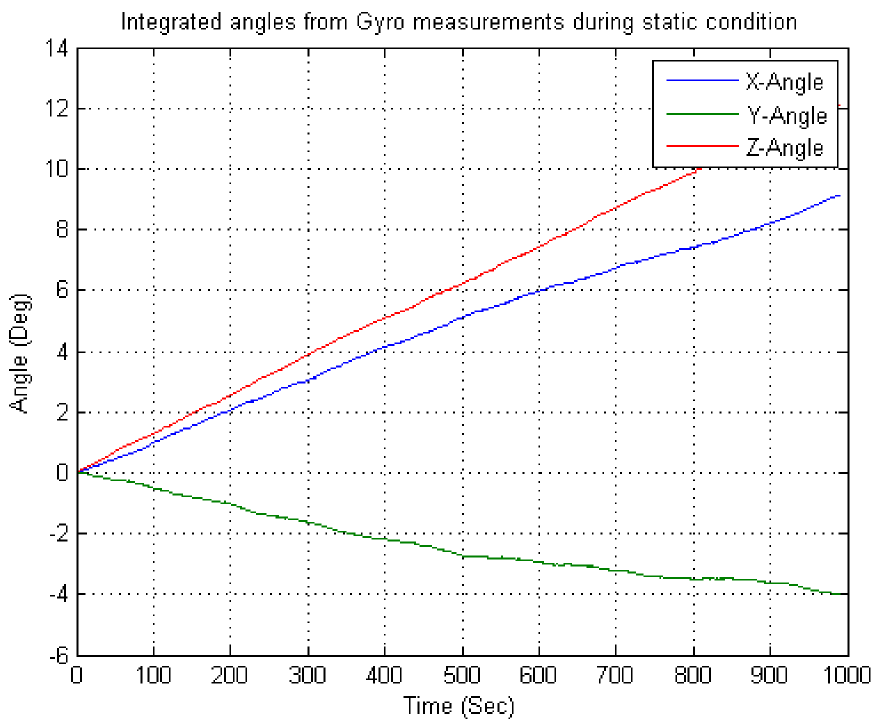
robot_pose_ekf

- Implements an extended Kalman filter for 3D pose estimation {url}
- roslaunch robot_pose_ekf.launch

```
<launch>
  <node pkg="robot_pose_ekf" type="robot_pose_ekf" name="robot_pose_ekf">
    <param name="output_frame" value="odom"/>
    <param name="freq" value="30.0"/>
    <param name="sensor_timeout" value="1.0"/>
    <param name="odom_used" value="true"/>
    <param name="imu_used" value="true"/>
    <param name="vo_used" value="true"/>
    <param name="debug" value="false"/>
    <param name="self_diagnose" value="false"/>
  </node>
</launch>
```

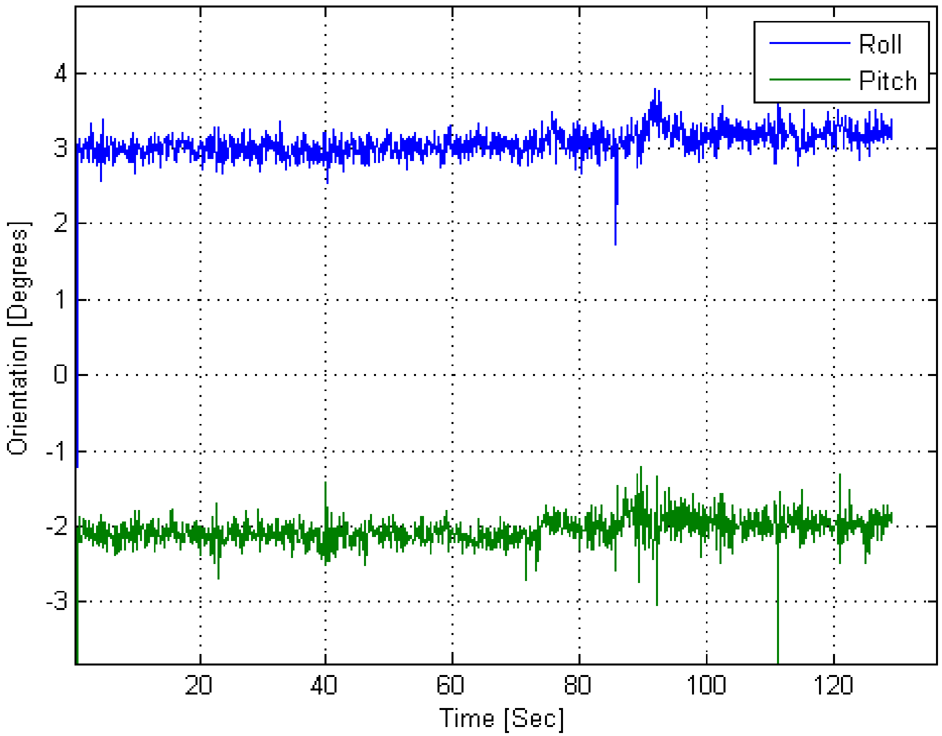
Euler Angles From Gyroscope

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}_t = \begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}_{t-1} + \begin{bmatrix}1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta)\\ 0 & \cos(\phi) & -\sin(\phi)\\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta)\end{bmatrix} \cdot \begin{bmatrix}p\\ q\\ r\end{bmatrix} \cdot \Delta t$$



Roll, Pitch angles from Accelerometer

$$\begin{bmatrix}\phi \\ \theta\end{bmatrix} = \begin{bmatrix} atan2(ay, ax) \\ -atan2\left(ax, \sqrt{a_y^2 + a_z^2}\right)\end{bmatrix}$$



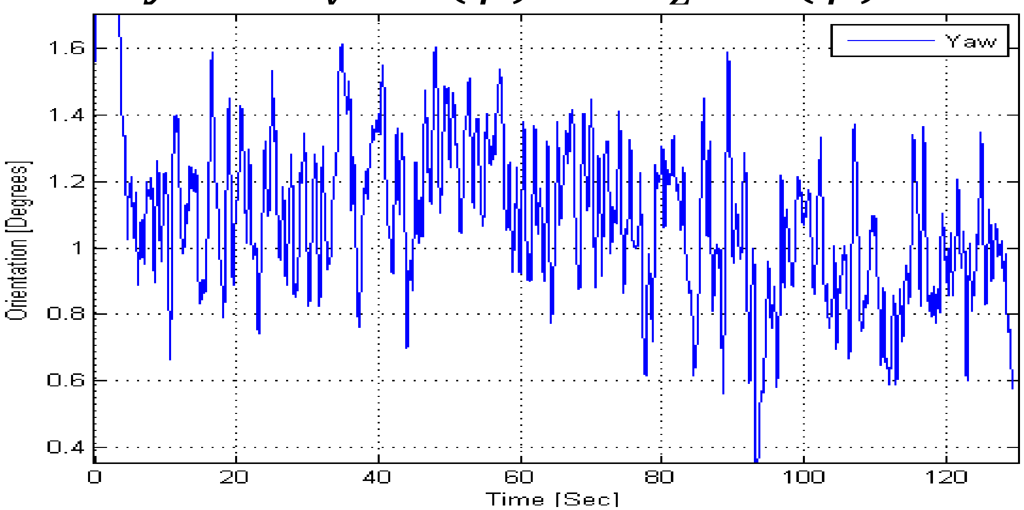
Yaw angle from magnetometer

$$M_b = R_x(\phi) \cdot R_y(\theta) \cdot R_z(\psi) \cdot M_i$$

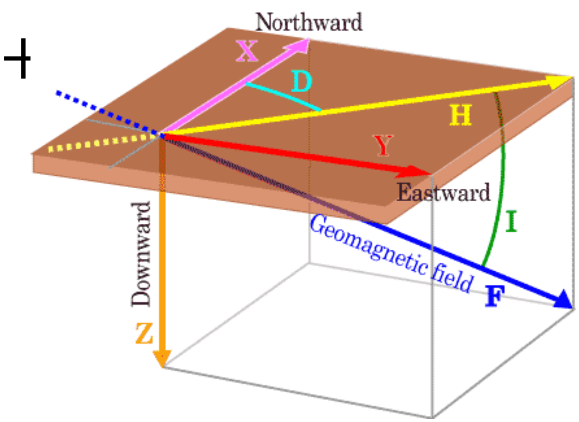
$$\begin{bmatrix} (m_x)\cos(\theta) + (m_y)\sin(\phi)\sin(\theta) + (m_z)\cos(\phi)\sin(\theta) \\ (m_y)\cos(\phi) - (m_z)\sin(\phi) \\ -(m_x)\sin(\theta) + (m_y)\sin(\phi)\cos(\theta) + (m_z)\cos(\phi)\cos(\theta) \end{bmatrix}$$

$$= \begin{bmatrix} B \cdot \cos(\delta) \cdot \cos(\psi) \\ -B \cdot \cos(\delta) \cdot \sin(\psi) \\ B \cdot \sin(\delta) \end{bmatrix}$$

$$y = m_y\cos(\phi) - m_z\sin(\phi)$$



) +



Task 2: Simulation Experiment

- Install quadrotor model for quadrotor
- Navigate the simulated quadrotor model using keyboard and joypad
- Setup robot_pose_ekf node for quadrotor

Lab Assignment

- To get the understanding of Kalman filer we shall implement it in a simple case where the quadrotor is stationary. Suppose we wish to filter accelerometer value which is almost constant except some small random noise. Therefore, the process model is as follows

$$x_t = x_0 + \mathcal{N}\left(0, \sigma_p^2\right)$$

- Accelerometer measurements are also subjected to random noise, therefore, the measurement model is as follows

$$y_t = x_t + \mathcal{N}(0, \sigma_m^2)$$

- Write a simple node which can subscribe to IMU topic and able to separately filter the three accelerometer values using Kalman filter methodology. Publish the estimated state and variance as a custom message consist of two fields. Using rqt_plot plot the published message.
  - Record the accelerometer measurements and measure the variance of accelerometer readings.
  - Since the measurement variance is fixed, observe the behavior of filter using different process noise variance
  - Now observe the estimated state and its variance using different initial values of the state and its variance.

Lab Assignment (Cont.)

- Calculate Euler angles for an Attitude and Heading Reference System (AHRS) using gyro-rate sensor, accelerometer and magnetometer.
  - Calculate the Euler angles from gyroscope's body-rate measurements as follows

$$\begin{bmatrix}\phi\\\theta\\\psi\end{bmatrix}_t = \begin{bmatrix}\phi\\\theta\\\psi\end{bmatrix}_{t-1} + \begin{bmatrix}1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta)\\0 & cos(\phi) & -sin(\phi)\\0 & sin(\phi)\,/cos(\theta) & cos(\phi)\,/cos(\theta)\end{bmatrix}\cdot\begin{bmatrix}g_x\\g_y\\g_z\end{bmatrix}\cdot\Delta t$$

  - The roll and pitch angle from accelerometer can be calculated as follows

$$\begin{bmatrix}\phi\\\theta\end{bmatrix} = \begin{bmatrix}atan2(ay, ax)\\-atan2\left(ax, \sqrt{a_y^2 + a_z^2}\right)\end{bmatrix}$$

  - The yaw angle from the magnetometer readings can be calculated as follows

$$y = m_y\cos(\phi) - m_z\sin(\phi)$$
$$x = m_x\cos(\theta) + m_y\sin(\theta)\sin(\phi) + m_z\sin(\theta)\cos(\phi)$$
$$\psi = -atan2(y, x)$$

Questions