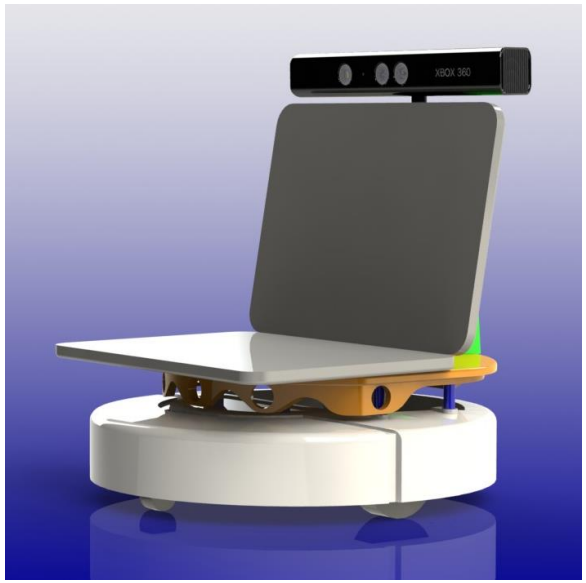
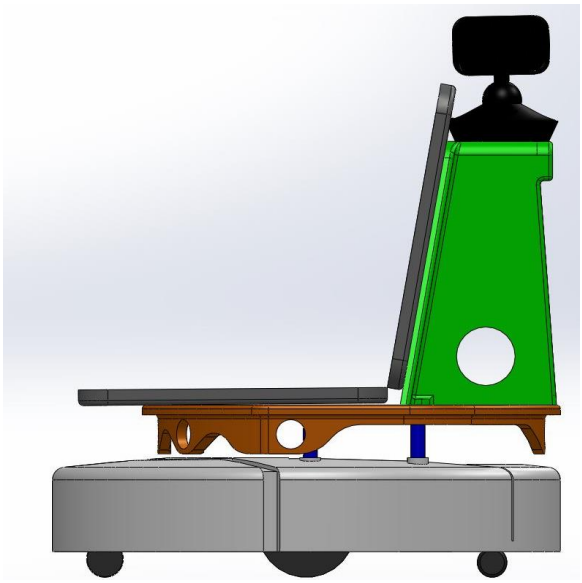
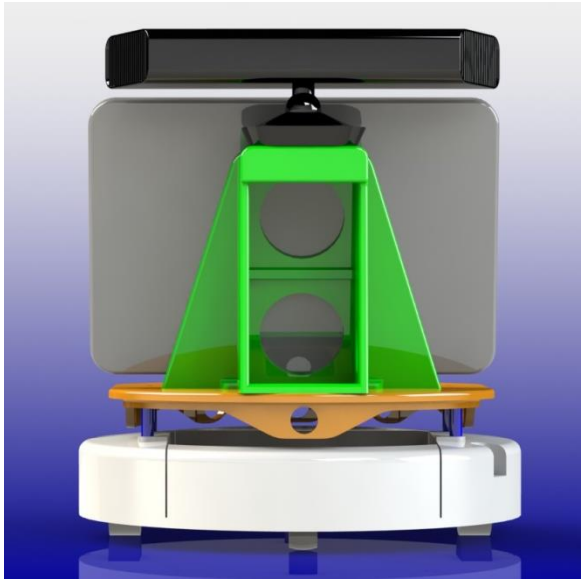
MAE 493G, CpE 493M, Mobile Robotics

9. Navigation and Robot Kinematics

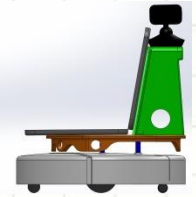






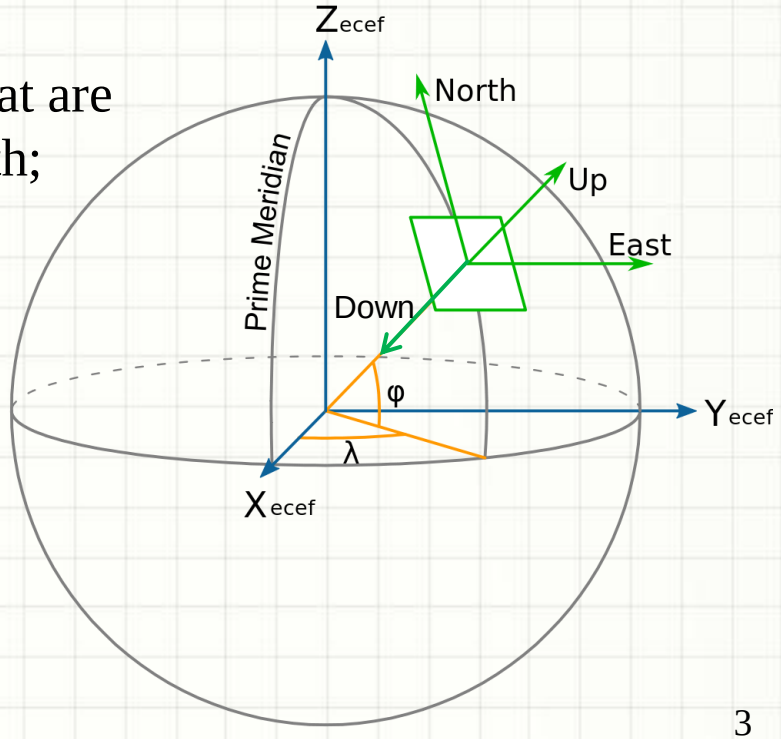
Navigation

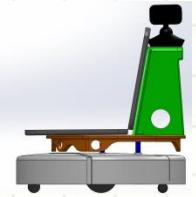
- A fundamental issue for any mobile robot is to figure out where it is at in its environment;
- For an outdoor robot, GPS makes navigation problem very easy;
- There are also indoor positioning systems to help determine a robot's position;
- Navigation in a general environment without using any man-made external aid is a very hard problem;
- However, humans can do it quite easily;
- Animals including humans are very good at *relative navigation*, but not *global navigation*.



Coordinate Systems

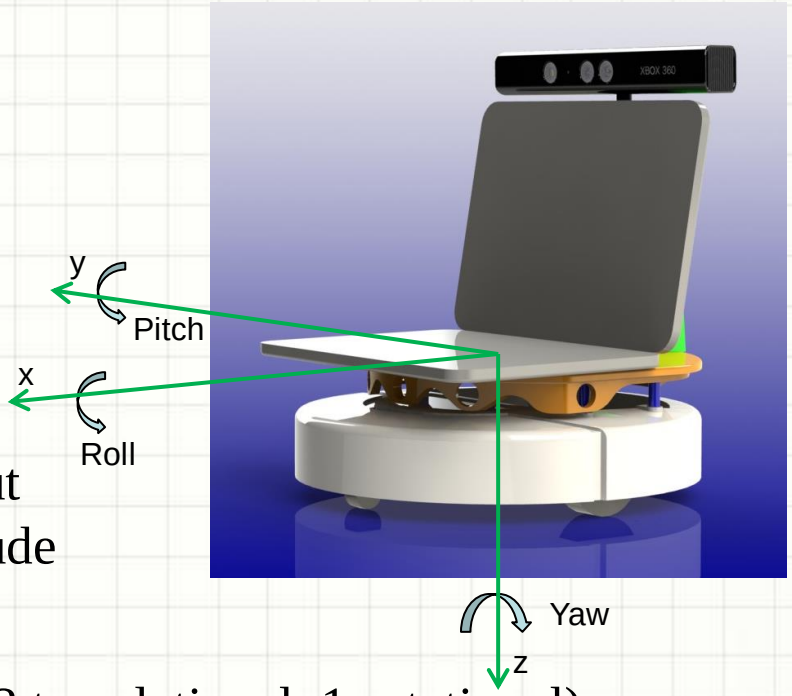
- To tell someone where you are, you need first to specify the *coordinate system*;
- A common global geographic coordinate system use latitude, longitude, and elevation to represent a position;
- ECEF (Earth-Centered, Earth-Fixed), is a Cartesian coordinate system with the origin defined at the Earth's center of mass. Its axes are aligned with the *International Reference Pole (IRP)* and *International Reference Meridian (IRM)* that are fixed with respect to the surface of the Earth;
- If you don't plan to travel for a long distance, the Local Tangent Plane (LTP) is easier to use;
- It's a Cartesian coordinate system with a local origin and the three axes typically pointing to North, East, and Down (NED).





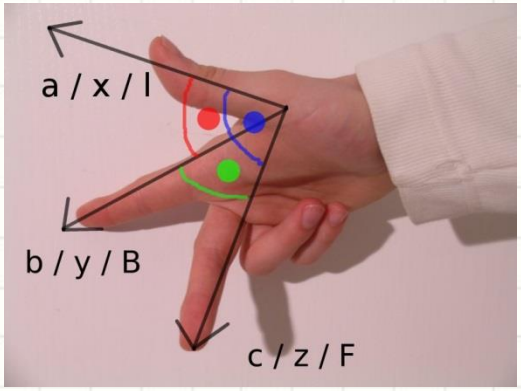
Robot Body Frame

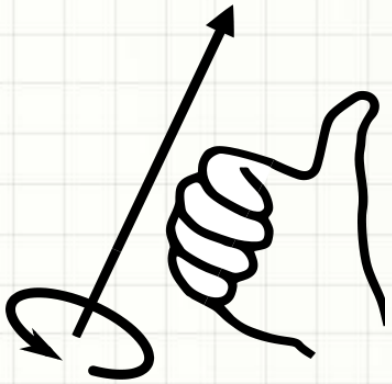
- From a robot's perspective, the body frame is carried by and stays fixed relative to the robot at all times;
- All the robot sensors are typically aligned with the body frame;
- For example with the IMU, we can measure a_x, a_y, a_z, p (roll rate), q (pitch rate), r (yaw rate), and M_x, M_y, M_z;
- Often we are interested in knowing how to transform from the body frame to a more global (or local) inertial frame;
- That is to determine from sensor measurements the state of the robot;
- For an aircraft, we typically worries about 6 degrees of freedom (DOF), which include 3 translational and 3 rotational DOFs.
- A ground robot typical only has 3 DOF (2 translational, 1 rotational).



Right Hand Rule Systems

- In addition to having the same coordinate system, we also need to maintain consistent conventions to be able to communicate effectively;
- Right-hand rule is a common convention for vectors in 3 dimensions;
- A different form of the right-hand rule, sometimes called the right-hand grip rule, is used when a vector (such as the Euler vector) must be defined to represent the rotation of a body;
- We will use both right hand rules for our robots.





2D Coordinate Transformation

- Giving a Cartesian coordinate system *A*, a point is represented by its coordinates (x_A, y_A) or as a vector: $p_A = x_A\hat{x}_A + y_A\hat{y}_A$

  Where $\hat{x}_A, \hat{y}_A$ are unit-vectors parallel to the axes;

- Now we want to represent the same point in a coordinate system *B*, which is offset by (x, y) and rotated counter-clockwise by an angle θ;

- To consider just rotation we create a new frame *V* whose axes are parallel to those of *A* but whose origin is the same as *B*;

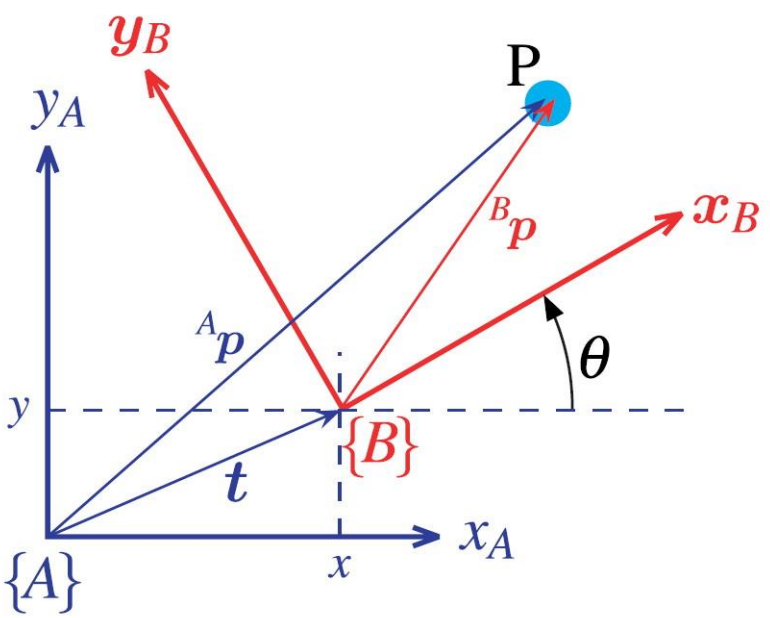
- The relationships between *B* & *V* are:

$$\hat{x}_B = \cos\theta\hat{x}_V + \sin\theta\hat{y}_V$$

$$\hat{y}_B = -\sin\theta\hat{x}_V + \cos\theta\hat{y}_V$$

- Or in the matrix form:

$$\begin{bmatrix}\hat{x}_B & \hat{y}_B\end{bmatrix} = \begin{bmatrix}\hat{x}_V & \hat{y}_V\end{bmatrix}\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$$

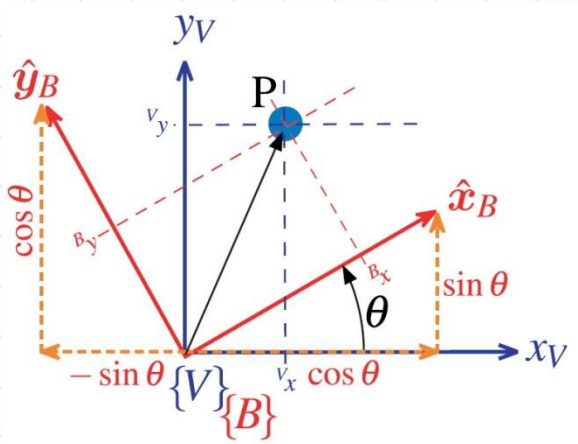


2D Transformation (Cont.)

- We can represent a point P with respect to B as:

$$p_B = x_B\hat{x}_B + y_B\hat{y}_B = \begin{bmatrix}\hat{x}_B & \hat{y}_B\end{bmatrix}\begin{bmatrix}x_B \\ y_B\end{bmatrix}$$

$$= \begin{bmatrix}\hat{x}_V & \hat{y}_V\end{bmatrix}\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x_B \\ y_B\end{bmatrix}$$



- Now we can describe how points are transformed from frame B to frame V, where $\mathbf{R}_B^V$ is called a rotation matrix.

$$\begin{bmatrix}x_V \\ y_V\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x_B \\ y_B\end{bmatrix} = \mathbf{R}_B^V\begin{bmatrix}x_B \\ y_B\end{bmatrix}$$

- We just need to transformed from frames V to B first, and then from B to A (translational):

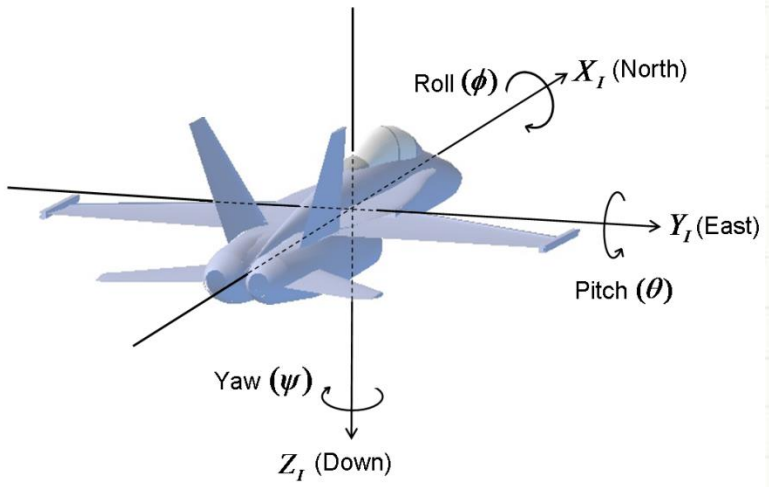
$$\begin{bmatrix}x_A \\ y_A\end{bmatrix} = \begin{bmatrix}x_V \\ y_V\end{bmatrix} + \begin{bmatrix}x \\ y\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x_B \\ y_B\end{bmatrix} + \begin{bmatrix}x \\ y\end{bmatrix}$$

Euler Angles

- Euler angles provide a way to represent the 3D orientation of an object using a combination of three rotations about different axes;
- Euler's rotation theorem states that any rotation can be considered as a sequence of rotations about different coordinate axes;
- There are many ways to rotate the frames... The one we are going to use here is to rotate along the *z*-axis first to get the yaw angle ψ, then the *y*-axis to get the pitch angle θ, then the *x*-axis to get the roll angle ϕ;
- The complete rotation matrix for moving from the *inertial frame to the body frame* is given by:

$$\mathbf{R}_I^B(\phi,\theta,\psi) = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)$$

- All the individual rotation matrixes are provided in the next page.



Euler Angles (Cont.)

$$\mathbf{R}_z(\psi)=\begin{bmatrix}\cos\psi & \sin\psi & 0\\ -\sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix},\ \mathbf{R}_y(\theta)=\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix},\ \mathbf{R}_x(\phi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & \sin\phi\\ 0 & -\sin\phi & \cos\phi\end{bmatrix}$$

$$\mathbf{R}_I^B(\phi,\theta,\psi)=\begin{bmatrix}c\psi c\theta & c\theta s\psi & -s\theta\\ c\psi s\phi s\theta - c\phi s\psi & c\phi c\psi + s\phi s\psi s\theta & c\theta s\phi\\ s\phi s\psi + c\phi c\psi s\theta & c\phi s\psi s\theta - c\psi s\phi & c\phi c\theta\end{bmatrix}$$

- The rotation matrix from the body frame to the inertial frame is given by:

$$\mathbf{R}_B^I(\phi,\theta,\psi)=\mathbf{R}_z(-\psi)\mathbf{R}_y(-\theta)\mathbf{R}_x(-\phi)=\begin{bmatrix}c\psi c\theta & c\psi s\phi s\theta - c\phi s\psi & s\phi s\psi + c\phi c\psi s\theta\\ c\theta s\psi & c\phi c\psi + s\phi s\psi s\theta & c\phi s\psi s\theta - c\psi s\phi\\ -s\theta & c\theta s\phi & c\phi c\theta\end{bmatrix}$$

- Singularity or Gimbal lock occurs when the orientation of the sensor cannot be uniquely represented using Euler Angles;
- In this case, when pitch equals to 90° we cannot really distinguish between yaw and roll angles.

3D Coordinate Transformation

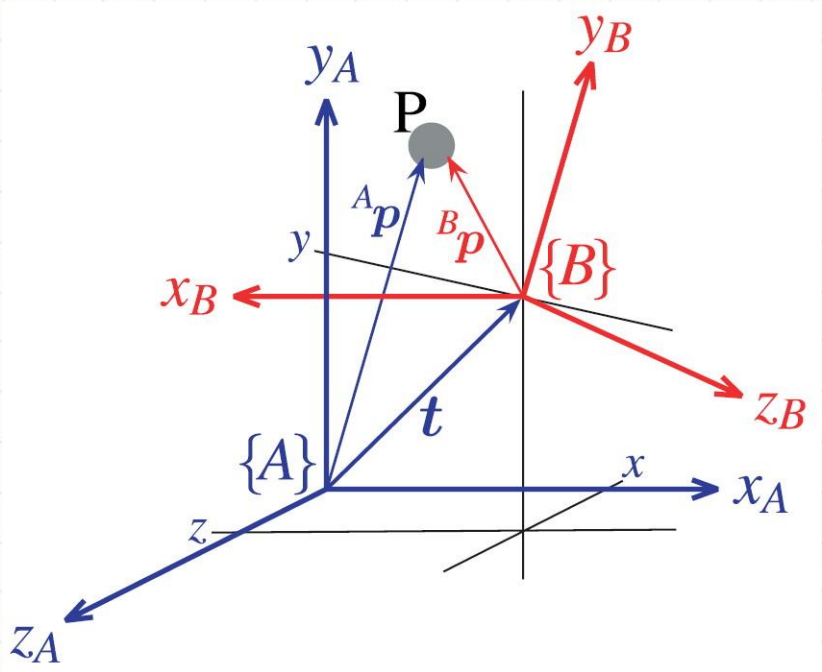
- Just like the 2D case, we can transform a position from frame *A* to *B* through rotation and translation:

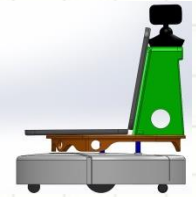
$$\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \mathbf{R}_B^A(\phi,\theta,\psi)\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- The rotation matrices are special in the sense that they are <u>orthogonal matrices</u> with determinant one:

$$\mathbf{R}^T = \mathbf{R}^{-1}, \det \mathbf{R} = 1$$

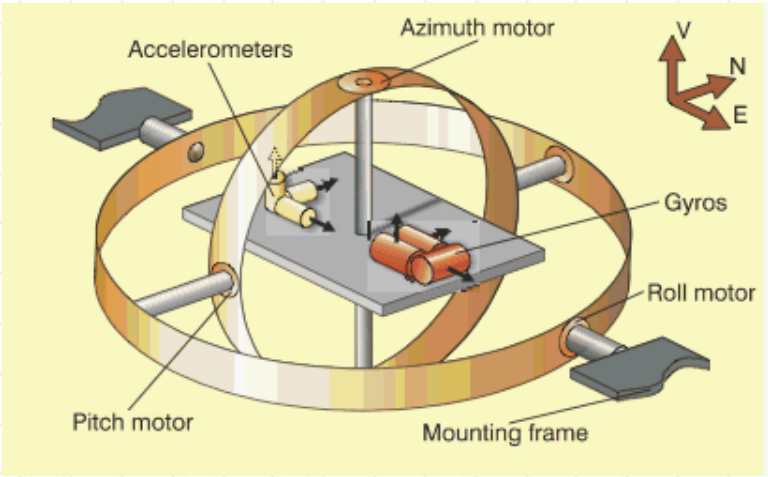
- When singularity is not acceptable, such as spacecraft navigation, other methods were developed to represent the vehicle orientation (e.g. quaternions).

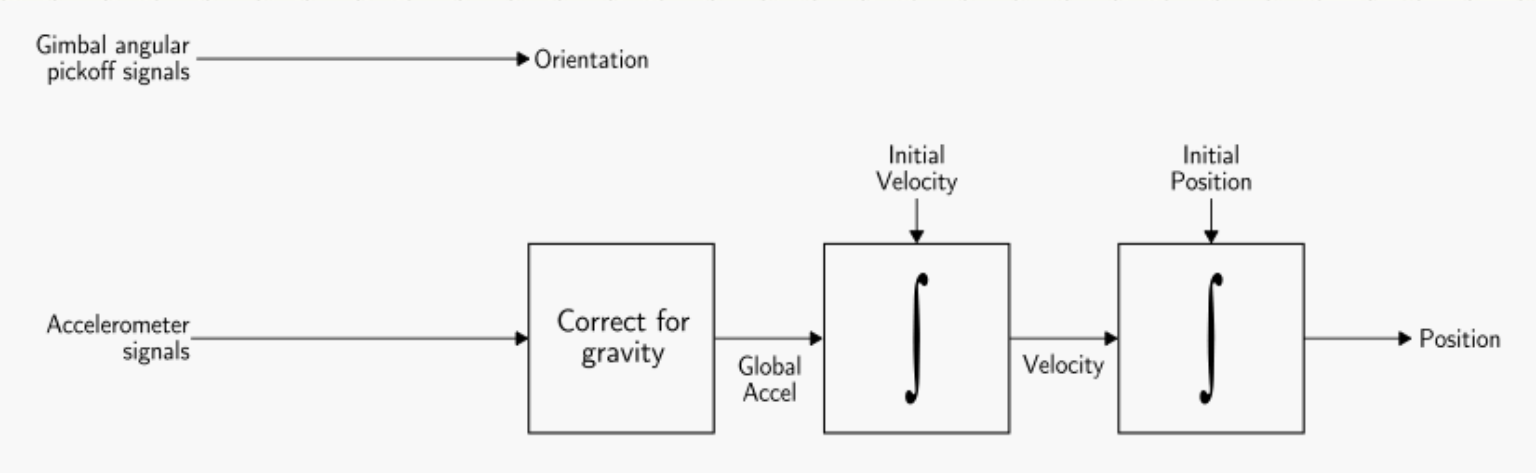


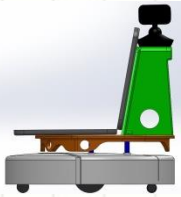


Gambled Inertial Navigation

- In a gambled inertial navigation system, all sensors including accelerometers and gyroscopes, are mounted on a platform which is isolated from any external rotational motion;
- The calculations are simple: everything are just based on straight integrations;
- However, the mechanical system is complex and heavy.

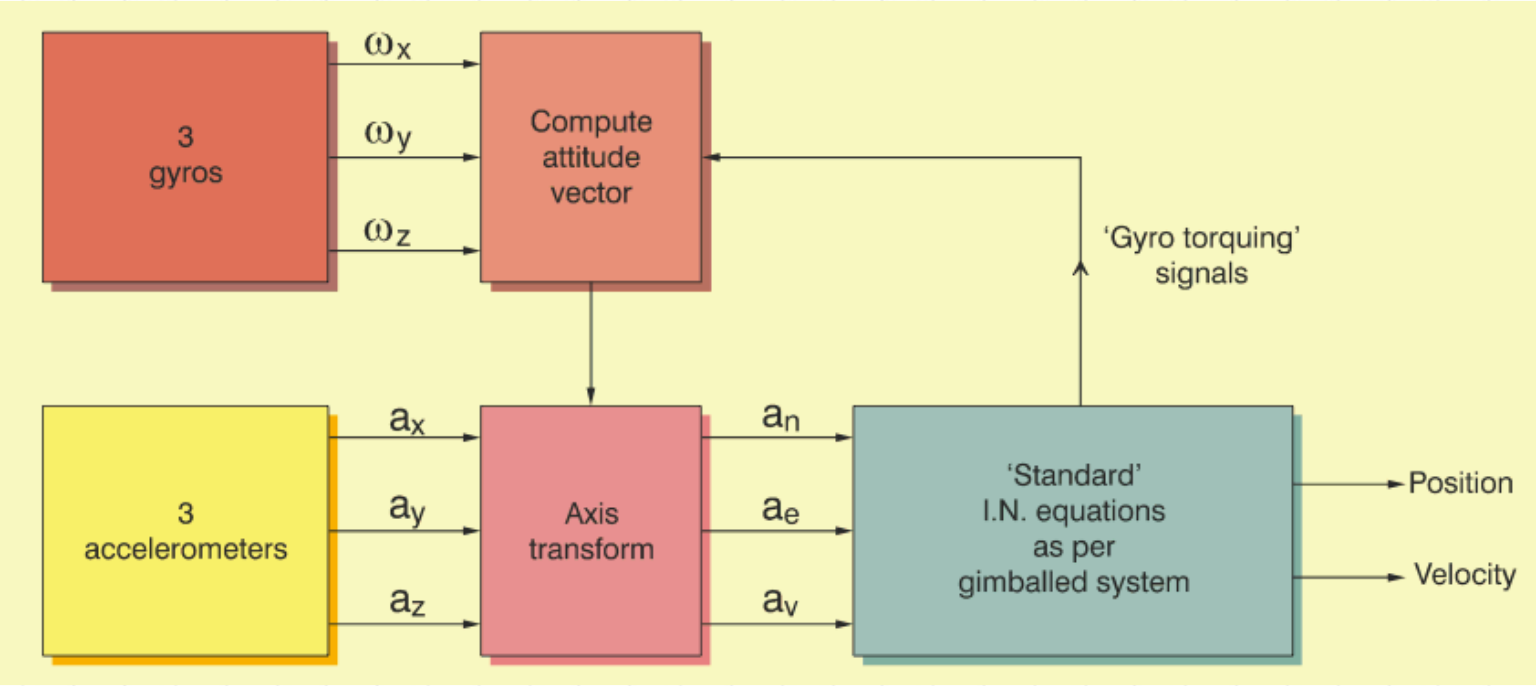


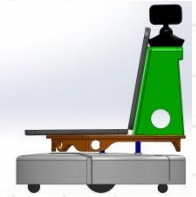




Strap-Down Inertial Navigation

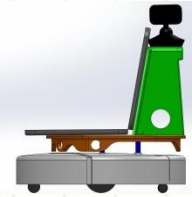
- In strap-down systems the inertial sensors are mounted rigidly onto the vehicle's body;
- It's a simpler system but with more computation requirements;
- Also the navigation equations are now *nonlinear* (with all the sine and cosine functions needed for coordinate transformations).





1D Dead Reckoning

- Dead reckoning is the process of estimating one's current position based on a previously determined position and advancing that position based upon known or estimated speeds over elapsed time, and course (e.g. inertial or encoder navigation).
- If we only have one degree of freedom motion, such as a train (translation) or a turn table (rotation), navigation is quite straight forward!
- *Translation*: $P = \int_t V = \iint_t a$, depending on what sensor you have (wheel encoder for V or accelerometer for a);
- *Rotation*: $\psi = \int_t r$, where the angular rate r can be measured with a rate gyro;
- Keep in mind that the integration process will accumulate errors over the time even if the sensor noise is zero mean (remember the *random walk* example earlier?);
- *Dead reckoning is subject to unbounded cumulative errors*!



2D Strap-Down Inertial Navigation

- For a 2D (level) ground robot navigation problem, we have five states to estimate: (x, y, V_x, V_y, ψ);
- Using inertial sensors, the first step is to figure out the robot yaw angle ψ using the yaw rate gyro measurement: $\psi = \int_t r$
- The second step is rotate the accelerometer measurements from the robot body frame to the inertial frame:

$$\begin{bmatrix} a_x^I \\ a_y^I \end{bmatrix} = \mathbf{R}_B^I \begin{bmatrix} a_x^B \\ a_y^B \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} a_x^B \\ a_y^B \end{bmatrix}$$

- Then we can do the straight integration of accelerations to get the robot position in the inertial frame;
- Writing in the state space form:

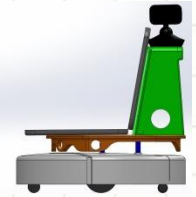
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{V}_x \\ \dot{V}_y \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ \cos\psi a_x^B - \sin\psi a_y^B \\ \sin\psi a_x^B - \cos\psi a_y^B \\ r \end{bmatrix}$$

3D Strap-Down Inertial Navigation

- For a 3D navigation problem, we now have nine states to estimate: (x, y, z, V_x, V_y, V_z, ϕ, θ, ψ);
- The idea is the same as the 2D navigation problem earlier: figure out the three Euler angles first, then rotate the accelerometer measurements to the inertial frame, then perform the double integration of accelerations to estimate the position;
- The Euler angles can be calculated through integrating the rate gyro measurements:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi/\cos\theta + r\cos\phi/\cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

- As we can see there is a division by cos(θ) so singularity could be a real issue sometimes.



3D Inertial Navigation (Cont.)

- In state space form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{V}_x \\ \dot{V}_y \\ \dot{V}_z \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ V_z \\ c\psi c\theta a_x^B + (c\psi s\phi s\theta - c\phi s\psi)a_y^B + (s\phi s\psi + c\phi c\psi s\theta)a_z^B \\ c\theta s\psi a_x^B + (c\phi c\psi + s\phi s\psi s\theta)a_y^B + (c\phi s\psi s\theta - c\psi s\phi)a_z^B \\ -s\theta a_x^B + c\theta s\phi a_y^B + c\phi c\theta a_z^B - g \\ p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi\sec\theta + r\cos\phi\sec\theta \end{bmatrix}$$

- We can see that the position calculation is the secondary process of the angle and velocity calculation. The error will build up in this process so getting a good position estimation using only inertial sensors can be quite hard!

GPS/INS Sensor Fusion Example

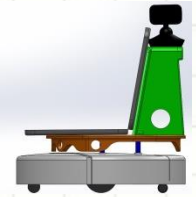
- GPS and INS (Inertial Navigation System) are meant to work together!
- GPS provides a statistically unbiased solution but could jump (high frequency noise) randomly around the true position;
- INS provides a smooth solution (through the integration process) but will drift over the time (low frequency error);
- Working together, the GPS/INS sensor fusion algorithm will smooth out the GPS jumps and stop the INS drift – best of both worlds!
- To do this, we can directly use the strap-down inertial navigation equations in the previous page for prediction;
- The input vector **u** will include all inertial measurements:

$$\mathbf{u} = \begin{bmatrix} a_x^B & a_y^B & a_z^B & p & q & r \end{bmatrix}^T$$

- The update will be based on the GPS measurements of the first six states: $\mathbf{y} = \begin{bmatrix} x & y & z & V_x & V_y & V_z \end{bmatrix}^T$

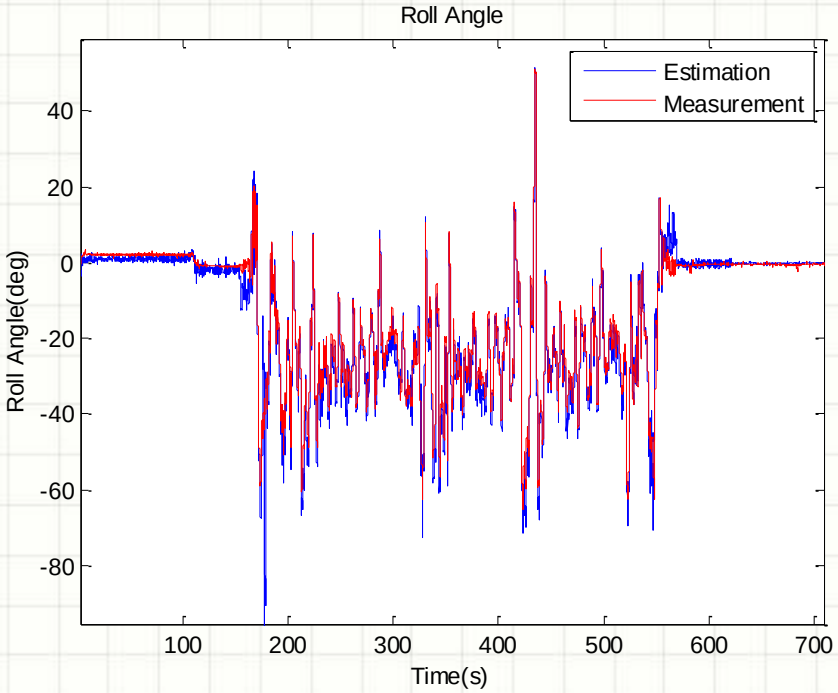
GPS/INS Sensor Fusion (Cont.)

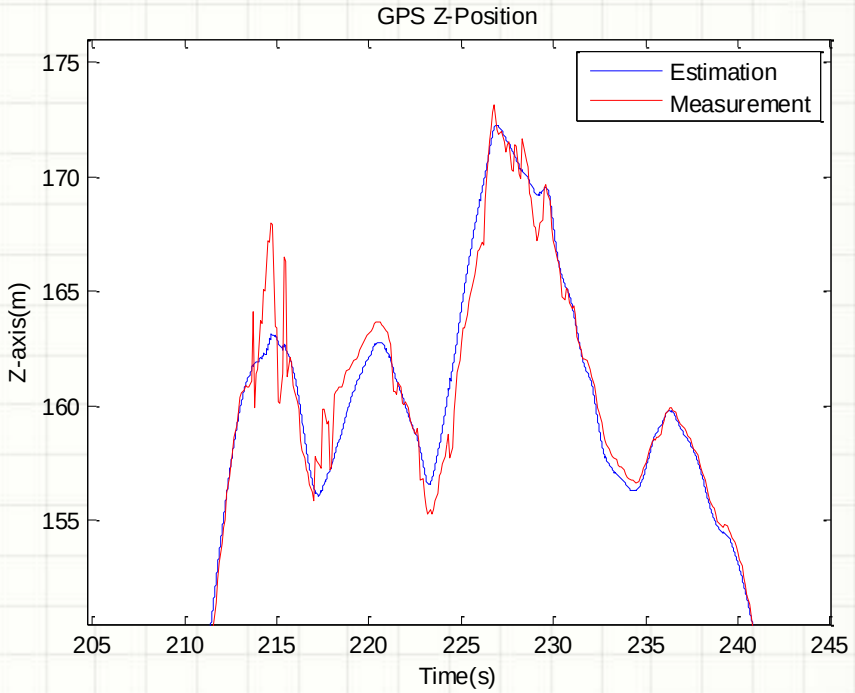
- We don't have a direct measurement of the three Euler angle states, but that's not a problem;
- These three angles have showed up in the velocity prediction equations so the update with the velocity measurement will also provide constraints on what the values of ϕ, θ, ψ can take;
- The rest are just the standard EKF procedures: discretization, calculate Jacobians, determine the process and measure noise covariance matrices, then apply the EKF equations, and maybe some tuning...
- There are also several other ways to perform the GPS/INS sensor fusion, which will not be discussed here;
- GPS/INS sensor fusion algorithms are very popular in autonomous systems.



GPS/INS Sensor Fusion Results

- Through this simple approach, we can achieved two goals:
  1. Smooth out noisy GPS measurements;
  2. Provide estimates of attitude angles that were not measured directly!





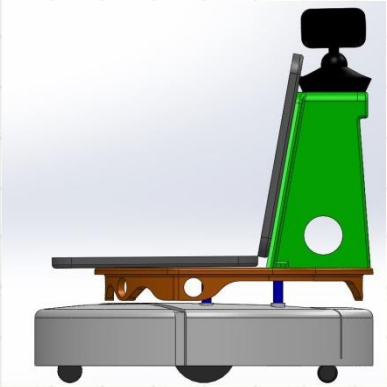
Tilt Sensing

- In addition to gyro integration, the robot orientation can also be determined by referencing some vectors of known directions;
- Three commonly accessible vectors are from the Earth's gravitational field (measured using accelerometers), magnetic fields (measured using magnetometers), and rotation (measured using rate gyros);
- The gravity vector is accessible everywhere and normally it cannot be interfered. It can be used to determine the roll and pitch angle for an object that is stationary or moving at a constant velocity:

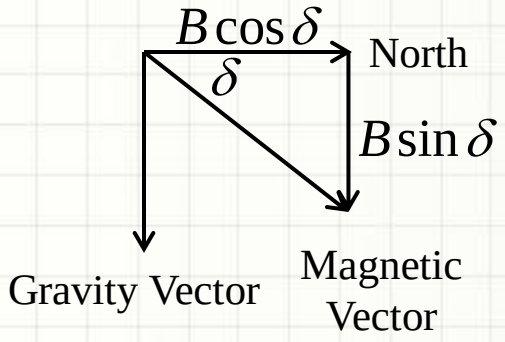
$$\phi = \arctan\left(\frac{a_y^B}{a_z^B}\right), \quad \theta = \arctan\left(\frac{-a_x^B}{a_y^B \sin\phi + a_z^B \cos\phi}\right)$$

- Keep in mind that this method will not work if the robot is in acceleration!
- We can get the heading angle with magnetometer readings, which has a lot of issues to be discussed next.

Magnetic Heading



- Pigeons, bats, and sea turtles all use magnetic information to find their way;
- The angle of inclination of the geomagnetic field, δ, is measured downwards from horizontal and varies over the earth's surface;
- It's about 67° near Morgantown;
- Assuming the magnetic north is the same as the geographic north and there are no hard or soft iron effects, which are not true, the magnetometer measurements are:



$$\mathbf{B}^B = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)B\begin{bmatrix}\cos\delta\\0\\\sin\delta\end{bmatrix}$$

- But we don't care too much about δ, what's more important is to figure out the magnetic heading in the horizontal plane;

Magnetic Heading (Cont.)

- With the angles ϕ and θ known from the accelerometers, the magnetometer readings can be de-rotated to correct for the sensor tilt:

$$\mathbf{R}_z(\psi)B\begin{bmatrix}\cos\delta\\0\\\sin\delta\end{bmatrix} = \mathbf{R}_y(-\theta)\mathbf{R}_x(-\phi)\mathbf{B}^B = \begin{bmatrix}B_x^H\\B_y^H\\B_z^H\end{bmatrix}$$

  Where H is a horizontal frame pointing to the same heading as the robot. Its x and y components are: $B_x^H = \cos\psi\cos\delta B, \quad B_y^H = -\sin\psi\cos\delta B$

- Now we can solve for the heading angle:

$$\psi = \arctan\left(\frac{-B_y^H}{B_x^H}\right) = \arctan\left(\frac{B_z^B\sin\phi - B_y^B\cos\phi}{B_x^B\cos\theta + B_y^B\sin\theta\sin\phi + B_z^B\sin\theta\cos\phi}\right)$$

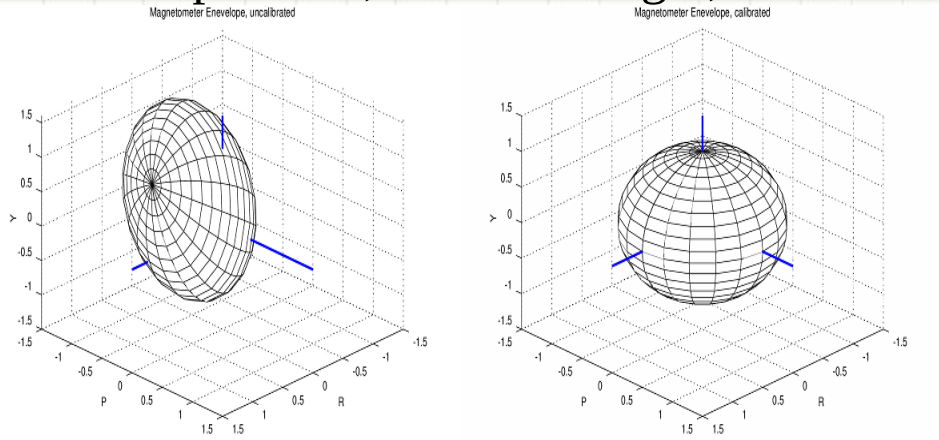
- We typically use the ATAN2 function (with output angle range -180° to 180°) to solve for ϕ and ψ, and use the ATAN function (with output angle range -90° to 90°) to compute θ.

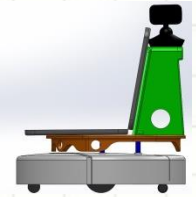
Magnetic Sensor Calibration

- In reality, the magnetometer data is corrupted by both *hard iron and soft iron effects*. The magnetometer measurements can be described as:

$$\mathbf{B}^{B}_{Measure} = \mathbf{W}\mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)B\begin{bmatrix}\cos\delta\\0\\\sin\delta\end{bmatrix} + \mathbf{V} = \mathbf{W}B^{B}_{ture} + \mathbf{V}$$

- Where **V** is a vector of three *hard iron* offsets and **W** captures the *soft iron* effect which includes three rotation and three scaling parameters;
- These parameters can be calibrated through rotating the robot to visit all different angles and then estimate the center position, the axes length, and the orientation of the generated ellipsoid;
- Once calibrated, the output should be close to a perfect sphere and centers at (0, 0, 0).

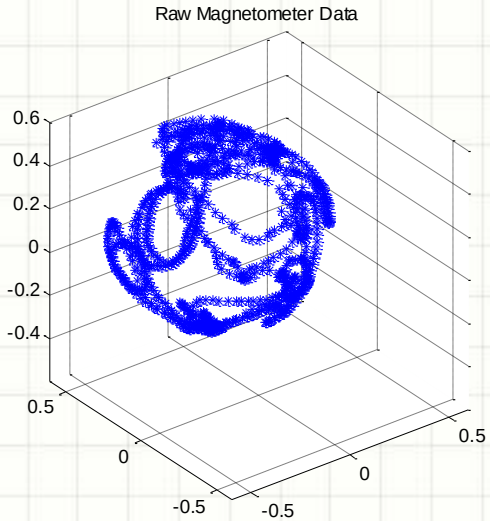




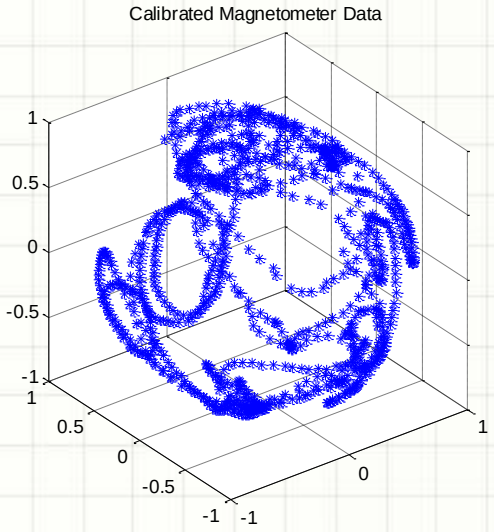
SMART Robot Calibration

- Below is the calibration result for one of the SMART robot:

Before:



After:



- Since we use the magnetometer only for heading, the converted results is normalized to a total strength of one. The calibration parameters are:

$$\mathbf{V} = \begin{bmatrix} 0.0067 \\ 0.2667 \\ 0.0473 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} 2.3750 & 0.2485 & -0.2296 \\ 0 & 2.6714 & -0.1862 \\ 0 & 0 & 2.5061 \end{bmatrix}$$

SMART EKF Example

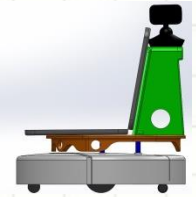
- The SMART robot also has a simple EKF for attitude estimation;
- It uses the gyro integration functions for prediction:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi/\cos\theta + r\cos\phi/\cos\theta \end{bmatrix}$$

- The predicted Euler angles are then updated with estimates using the gravity vector and the magnetic vector:

$$\phi = \arctan\left(\frac{a_y^B}{a_z^B}\right), \quad \theta = \arctan\left(\frac{-a_x^B}{a_y^B\sin\phi + a_z^B\cos\phi}\right)$$

$$\psi = \arctan\left(\frac{-B_y^H}{B_x^H}\right) = \arctan\left(\frac{B_z^B\sin\phi - B_y^B\cos\phi}{B_x^B\cos\theta + B_y^B\sin\theta\sin\phi + B_z^B\sin\theta\cos\phi}\right)$$



SMART EKF: Practical Issues

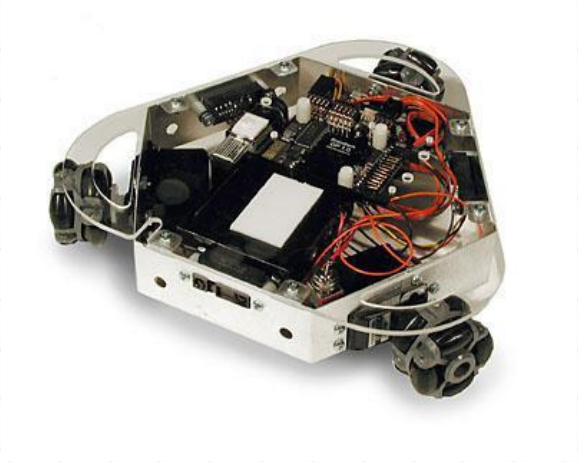
- Several issues had to be consider for this EKF to work properly;
- First, the initial gyro bias needs to be removed for a good dead reckoning performance;
- Second, we should only update the ϕ and θ predictions if the robot is not accelerating too hard; The total acceleration is: $a_{total} = \sqrt{a_x^2 + a_y^2 + a_z^2}$
- If it deviates too far from 1g, which only happens occasionally, the filter will rely only on dead reckoning without getting an update;
- Third, the magnetometer have to be calibrated for soft and hard iron effects;
- Finally, even if fully calibrated, local magnetic field (caused by objects outside of the robot) will still affect the heading estimation;
- So the total magnetic strength is also calculated and the update will stop if the magnetometers sense abnormal values;
- Even with all these approaches, the filter still works the best without any iron objects around the robot...

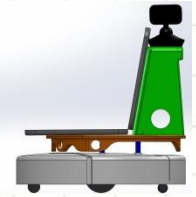
Robot Dynamics and Kinematics

- Robot dynamics is concerned with the effects of forces on the motion of the robot;
- The robot kinematics is mathematics of motion without considering the forces that affect the motion;
- If we are controlling a high speed robot or a walking robot, we may need to carefully model its dynamics to achieve the best performance or stability;
- For a (low-performance) ground robot such as the Create, we typically don't worry too much about dynamics;
- However, we need to be familiar with the robot kinematics for navigation, planning, and position control applications.

Kinematic Constraints

- A holonomic robot is one which is able to move instantaneously in any direction in the space of its degrees of freedom;
- Most robots however cannot move that way...
- For example, we can not directly drive sideways with our cars;
- When we are planning the motion of a robot, we should take into account of these limitations;
- From a positioning point of view, these constraints can sometime work to our favor!
- Because we know the robot cannot be at certain places at certain time...
- So that will limit the possible poses (position and orientation) the robot can be at any given time.



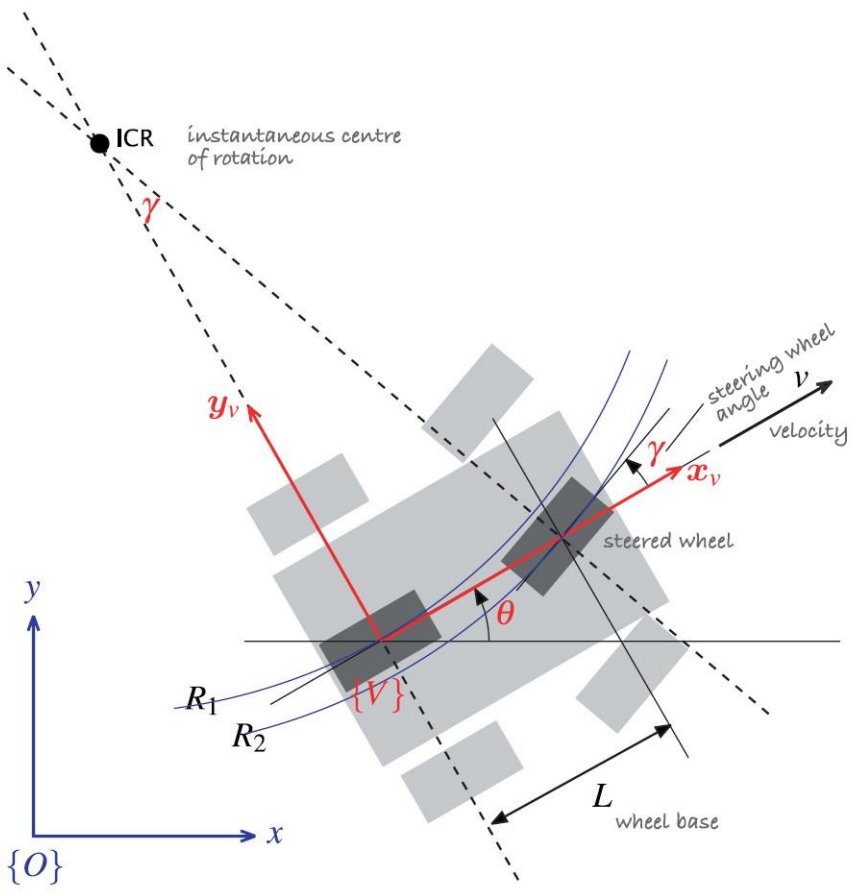


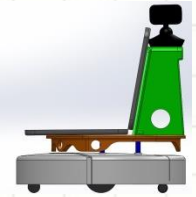
Robot Kinematics – Steering

- A commonly used model for a four-wheeled car-like vehicle is the bicycle model;
- The bicycle has a rear wheel fixed to the body and the plane of the front wheel rotates about the vertical axis to steer the vehicle;
- The robot cannot move sideways:

$$V_x^B = V, \quad V_y^B = 0$$

- The dashed lines show the direction along which the wheels cannot move, and these intersect at a point known as the Instantaneous Centre of Rotation (ICR) or Instantaneous Center of Curvature (ICC).





Steering (Cont.)

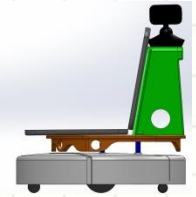
- The robot follows a circular path with center at ICR and angular velocity:

$$\dot{\theta} = \frac{V}{R_1} = \frac{V \tan\gamma}{L}$$

- Where R_1 is the turning radius, L is the length of the vehicle or wheel base, and γ is the steering angle;
- The front wheel must follow a longer path than the back wheel: $R_2 > R_1$;
- The two steered wheels follow circular paths of different radius and therefore the angles of the steered wheels γ_L and γ_R should be very slightly different;
- The velocity of the robot in the inertial frame is ($V\cos\theta$, $V\sin\theta$) so the equations of motion are:

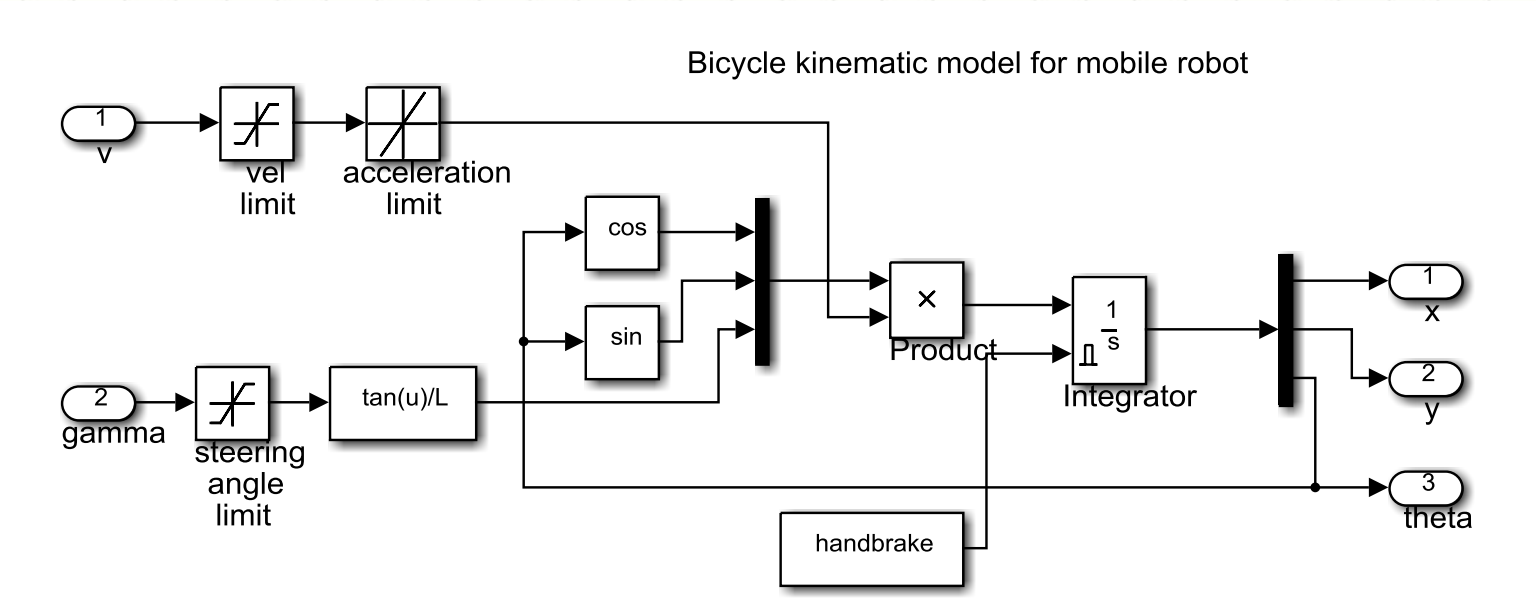
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ V\tan\gamma/L \end{bmatrix}$$

- When V is 0, the angle will not change, that is why cannot turn when the robot is not moving.



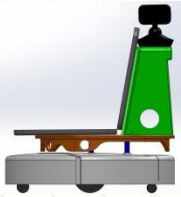
Steering in Simulink

- The bicycle model can be easily simulated in Simulink:



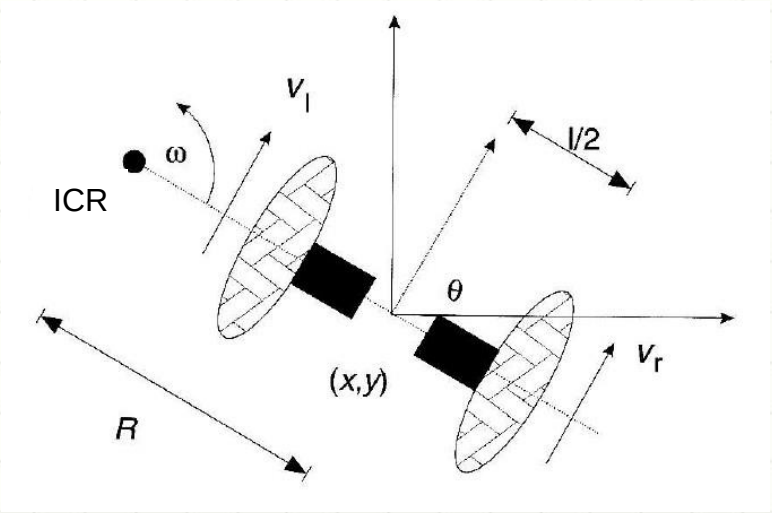
- This simulation is available in the Robotics Toolbox for MATLAB by Peter Corke;
- Make sure you try it out!

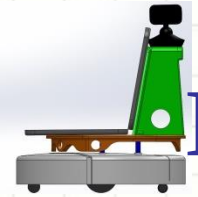
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ V\tan\gamma/L \end{bmatrix}$$



Kinematics – Differential Drive

- If we define left and right wheel velocities along the ground V_L and V_R, the signed distance from the ICR to the midpoint between the wheels is R, and width between wheels is l. The rate of rotation ω about the ICR must be the same for both wheels: $\omega(R + l/2) = V_r, \quad \omega(R - l/2) = V_l$
- Solving these equations, we get: $R = \frac{l}{2}\frac{V_l + V_r}{V_r - V_l}; \quad \omega = \frac{V_r - V_l}{l}$
- If $V_l = V_r$, the robot will drive straight forward;
- If $V_l = -V_r$, the robot will turn on the spot;
- Otherwise the center of the robot will both translate and rotate;
- If $V_l = 0$, and $V_r \neq 0$ the robot will rotate about the left wheel.



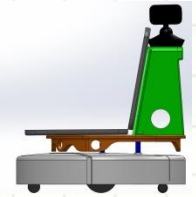


Kinematics Model in the Body Frame

- Solving the equations earlier, we can get: $V = \omega R = \frac{V_r + V_l}{2}; \quad \omega = \frac{V_r - V_l}{l}$
- In the body frame, we have the following relationships:

$$\begin{bmatrix} V_x^B \\ V_y^B \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ -r/l & r/l \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix}$$

- Where r is the wheel radius, ω_l and ω_r are the angular velocity of the left and right wheels;
- This robot model is useful for robot velocity control;
- The diameter of the wheels on Create is ~6.5 cm and the distance (center to center) between the two wheels is ~26.3 cm;



Model in the Inertial Frame

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \omega \end{bmatrix}$$

- Forward kinematic model of the differential drive robot:
- The location of the ICR is:ICR $= \left[x - R\sin\theta, y + R\cos\theta\right]$
- The forward kinematic model can be used for navigation; for example, we can do dead reckoning to find out the robot position based on wheel encoder readings;
- The opposite side of the question is: How can we control the robot to reach a given configuration (also called pose) (x, y, θ) – this is known as the inverse kinematics problem;
- Unfortunately, a differential drive robot has *non-holonomic constraints* so we cannot specify an arbitrary robot pose (position and orientation) and find the wheel velocities that will get us there;
- Some motion planning is generally needed to drive from one pose to another.

Introduction to SLAM

- Simultaneous Localization and Mapping (SLAM) is a group of methods that solves the (chicken and egg) problem of how to determine a robot's location in a map while building the map at the same time;
- The solving of the SLAM problem was viewed as one of the major breakthroughs in robotics;
- The idea is quite intuitive: when visiting a new place, we typically navigate around and get to know the place while keep tracking of our own location;
- This can be done with a EKF (among many other algorithms) in a prediction and update framework: use dead reckoning (with encoders or inertial sensors) for the prediction of the robot pose, and use sensors (laser scanner or camera) to identify landmarks and update the robot pose and landmark positions simultaneously;
- A YouTube video can be seen here.

Summary

- How to get around in the environment without needing the help from an external positioning system is a great challenge in robotics;
- Dead reckoning alone will always drift over the time;
- Vectors of known directions, such as gravity and magnetic vectors, can be used to help figure out the robot orientation (under certain limitations);
- Finding the robot position is in general a much hard problem than finding the robot orientation;
- Sensor fusion, such as GPS/INS, can provide good navigation solutions with low-cost but complementary sensors;
- SLAM is another sensor fusion algorithm that can help robots to navigate without the GPS;
- The forward kinematics can be used for navigation and simulation, while the inverse kinematics can be used for robot motion control.

Further Reading

- Our Textbook;
- Differential Drive Robots, http://chess.eecs.berkeley.edu/eecs149/documentation/differentialDrive.pdf
- Understanding Euler Angles, http://www.chrobotics.com/library/understanding-euler-angles
- Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors, http://freescale.com.hk/files/sensors/doc/app_note/AN4248.pdf
- Calibrating an eCompass in the Presence of Hard and Soft-Iron Interference, http://www.freescale.com/files/sensors/doc/app_note/AN4246.pdf
- Search Wikipedia for keywords ‘Inertial Navigation’, ‘Rotation Matrix’, ‘Dead Reckoning’, and ‘SLAM’.