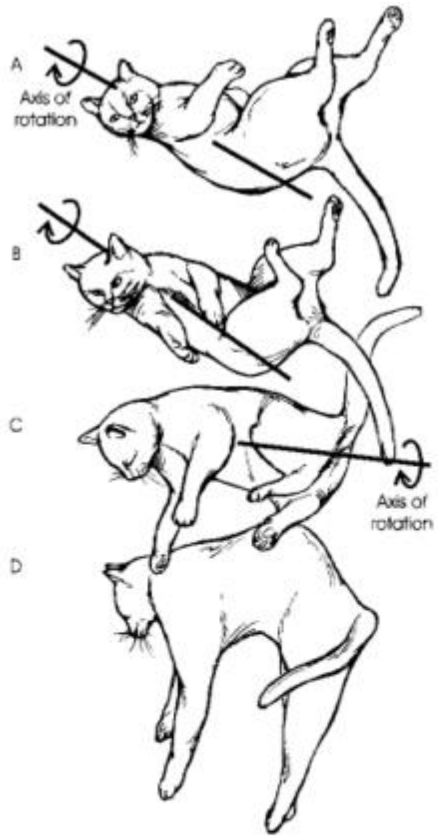
Advanced Mechatronics Project 2: Propeller

Members: Manoj Bandri, Wenjie Chen

Presentation Date: 4/19/16

Motivation

- Same motivation as last time
- When a cat falls in the air with its back facing the ground, it knows how to maneuver itself to land upright on its feets
- Robotic systems can also take advantage of such maneuver to properly orient itself in the case of falling from heights



Conservation of Angular Momentum

- Motors and body will turn in a manner to conserve angular momentum since no external forces are applied to the system

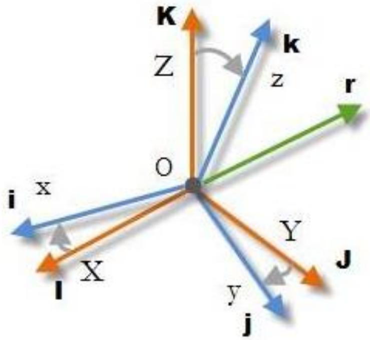
$$\Sigma M_o = \dot{H}_o$$

$$\dot{H}_o = 0 \qquad H_o = \text{constant}$$

Directional Cosine Matrix

- Assign the world frame with global coordinate system {X, Y, Z}, Frame {A}
- Assign body local coordinate system {x, y, z}, Frame {B}
- Directional Cosine Matrix, also known as Rotation Matrix, maps the local coordinate system onto global coordinate system

$$
{}^{A}_{B}R = \begin{bmatrix} X \cdot x & X \cdot y & X \cdot z \\ Y \cdot x & Y \cdot y & Y \cdot z \\ Z \cdot x & Z \cdot y & Z \cdot z \end{bmatrix}
$$



Rotation Matrix

$$ {}^{A}_{B}R_{X,\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} $$

$$ {}^{A}_{B}R_{Y,\beta} = \begin{bmatrix} cos\beta & 0 & sin \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} $$

$$ {}^{A}_{B}R_{Z,\gamma} = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{A}_{B}R_{XYZ}(\alpha, \beta, \gamma) $$

$$ = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\beta & 0 & sin \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} $$

Equivalent Angle Rotation

- A sequence of rotations can be modeled as the rotation about an equivalent axis K by an angle θ

- Rotation Matrix:

$$
{}^{A}_{B}R(\hat{K},\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix}
$$

$$
= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}
$$

Equivalent Angle Rotation

$$\theta = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\hat{K} = \frac{1}{2sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

- Ill defined if θ = 0 or π

Materials

Materials	Quantity
Brushless Motors	3
Brushless Motor ESCs	3
Lipo Battery 3S	1
Propeller	1
Printed enclosure	6
Velcro	Many
3-Axis Accelerometer	1
3-Axis Magnetometer	1

Use of Materials

- Accelerometer measures acceleration in 3 axis

- Gravity acts in direction parallel to global Z axis, but of opposite sense

- Accelerometer provides information regarding directional cosine with respect to the global Z axis (3rd row of Rotation Matrix)

$$ {}^{A}_{B}R = \begin{bmatrix} X \cdot x & X \cdot y & X \cdot z \\ Y \cdot x & Y \cdot y & Y \cdot z \\ Z \cdot x & Z \cdot y & Z \cdot z \end{bmatrix} $$

Use of Materials

- Magnetometer provides information regarding alignment to magnetic fields

- Assume Earth's Magnetic field acts in direction parallel to its surface

- Magnetometer can be used to establish directional cosine with respect to global X axis

$$ {}^{A}_{B}R = \begin{bmatrix} X \cdot x & X \cdot y & X \cdot z \\ Y \cdot x & Y \cdot y & Y \cdot z \\ Z \cdot x & Z \cdot y & Z \cdot z \end{bmatrix} $$

- Second row can be calculated by cross product

PLAN

- Using Equivalent Axis Rotation, we can determine the axis the body should rotate to obtain its reference position

$$\hat{K}\theta = \begin{bmatrix} k_x\theta \\ k_y\theta \\ k_z\theta \end{bmatrix}$$

- Extract information to determine the required rotation about each individual axis to reproduce same effect (how to actuate motors)

- Use accelerometer to detect fall to start actuation

Problems

- Magnetometer was not properly calibrated in time to be of good use
- Cannot determine Rotation Matrix beyond third row
- Result in non-unique solution as to how to rotate body

Solution?

- Hard code a fixed orientation to be tested
- Apply the same theory to test accuracy of model

Program

```
#include "servo.h"
#include "simpletools.h"
#include "mma7455.h"
#include "compass3d.h" // Include compass3d header
#include "simplei2c.h"
#include "math.h"
void motor_x(void *par);
void motor_y(void *par);
void motor_z(void *par);

unsigned int stack1[40+25];
unsigned int stack2[40+25];
unsigned int stack3[40+25];

int DATA = 7, CLK = 8, ENABLE = 6; /// accel pins
signed char p, q, r;
//stores information on magnitude of acceleration
float u;
//hardcoded rotation matrix
int rot_matrix [3][3]=
{
{0,1,0},
{0,0,-1},
{-1,0,0}
};

//calculcated corresponding K vector
float k[3][1]=
{
{sqrt(1/3)},
{sqrt(1/3)},
{-sqrt(1/3)}
};
```

```
//main cog gathers sensor data
int main()
{
  //process action for motor in x axis
  cogstart(&motor_x,NULL,stack1,sizeof(stack1));
  //process action for motor in y axis
  cogstart(&motor_y,NULL,stack2,sizeof(stack2));
  //process action for motor in z axis
  cogstart(&motor_z,NULL,stack3,sizeof(stack3));
  ////// ACCELEROMETER//////
  //initialize
  MMA7455_init(DATA, CLK, ENABLE);
  //set offsets from experiment
  MMA7455_setOffsetX(16);
  MMA7455_setOffsetY(61);
  MMA7455_setOffsetZ(-6);
  //calculated angle of rotation
  float theta= 120;
  int row,column;
  for ( row = 0; row < 3; row++ )
  for ( column = 0; column < 1; column++ )
  //isolates necessary actuation for each motor
  k[row][column] *= theta;
  pause(10);
  while(1)
  {
  //// ACCEL CODE  //////

    MMA7455_gRange(4);
    MMA7455_getxyz8(&p, &q, &r);
    //computes magnitude of acceleration
    u=sqrt(p*p+q*q+r*r);
    pause(50);
  }
}
```

PROGRAM

```
void motor_x(void *par) { //pin3

  float b;
  //calculate corresponding pulse
  float a= k[1][1]/0.5;
      if(a>0){
        b= -1.58 * a +1500;
      }
      else {
        b= 1.58 * a +1500;
      }
  //equivalently arms motor
  servo_speed(3,00);
  pause(5000);
  while(1){
   //actuate motor when fall detected
   if(u<24){
   servo_angle(3,1500); //980-1030
   pause(500);
   servo_angle(3,1000);
   pause(500);
   }
   pause(50);
  }
}
```

```
//same logic as motor_x
void motor_y(void *par) {
  float b;
  float a=k[2][1]/0.5;
    if(a>0) {
    b= -1.58 * a +1500;
    }
    else {
    b= 1.58 * a +1500;
    }
  servo_speed(4,00); //4
  pause(5000);
  while(1){
    if(u<27) {

    servo_angle(4,1500);
    pause(500);
    servo_angle(4,1000);
    pause(500);
    }
    pause(50);
  }
}
```

```
//same logic as motor_x
void motor_z(void *par) {
  float b;
  float a=k[3][1]/0.5;
  if(a>0) {
  float b= -1.58 * a +1500;
  }
  else {
  float b= 1.58 * a +1500;
  }
  servo_speed(5,00); //5,3
  pause(5000);
  while(1){
    if(u<27){
      servo_angle(5,305);
      pause(500);
      servo_angle(5,1000);
      pause(500);
    }
    pause(50);
  }
}
```