

# Brigham Young University AUVSI Capstone Team (Team 45)

## Geolocation Algorithm Description

ID	Rev.	Date	Description	Author	Checked By
IM-004	1.0	12-12-	Initial release	Connor Olsen	Tyler Miller
		2018			



#### **Contents**

- Initial Parameters
- Rotation Matrices
- Geolocation Algorithm
- Publish

```
clear
clc
```

#### **Initial Parameters**

These initial parameters represent data that will be readily accessible on the MAV. Pixel Array, Field of View angle are specifications of the camera, and the normalized line of sight vector (I\_cusp\_c) will be determined by either the autonomous detection program, or the manual interface clients.

```
% The X and Y pixels denote the location of the target in the image plane
% and are centered about the center of the image, with +x extending to the
% 'right' and +y extending 'down'
x pixel = 10;
y pixel = 10;
M = 256; %square pixel array (width and height if square)
Ex = x pixel/M; %X pixels, normalized
Ey = y pixel/M; %Y pixels, normalized
fov ang = pi/2;%sym('fov ang'); %field of View angle --> A6000 83* - 32* (in radians)
f = M/(2*tan(fov ang/2)); %focal length in units of pixels
l_{cusp_c} = 1/sqrt(Ex^2 + Ey^2 + f^2) * [Ex;Ey;f];
roll = 0;%sym('roll'); %Radians
pitch = 0;%sym('pitch'); %Radians
yaw = 0;%sym('yaw'); %Radians
alpha az = 0;%sym('az'); %Azmuth Angle: Should equal yaw.
alpha el = 0; % sym('el'); % Elevation Angle: "Pitch" of the gimbal
StartingLat = 40.248459;
StartingLon = -111.645809;
%P vector should be in Meters
Pn = 20;
Pe = 20;
Pd = -100; %sym('Pd');
h = -Pd; %; sym('h');
k_i = [0;0;1];
```

#### **Rotation Matrices**

The location of the target is defined in the camera coordinate frame system, and must be transformed into the inertial frame



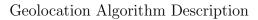
to provide GPS coordinates. This transformation is done using three rotation matrices: Moving Camera frame to Gimbal coordinates; Gimbal coordinates to body frame; body frame to inertial frame

```
% R b to i
% Found on page 15 of Small Unmanned Aircraft
R v to b = [\cos(pitch) * \cos(yaw) \cos(pitch) * \sin(yaw) - \sin(pitch); ...
   sin(roll)*sin(pitch)*cos(yaw)-cos(roll)*sin(yaw) sin(roll)*sin(pitch)*sin(yaw)+cos(rol
1) *cos(yaw) sin(roll) *cos(pitch); ...
   cos(roll)*sin(pitch)*cos(yaw)+sin(roll)*sin(yaw) cos(roll)*sin(pitch)*sin(yaw)-sin(rol
1) *cos(yaw) cos(roll) *cos(pitch)];
R b to v = R v to b'; %R v b is a translation of R i b. It may work, but I may need one mo
re step.
R_b_{to}i = R_b_{to}v;
% R_g_to_b
% Found on page 227 of Small Unmanned Aircraft
R b to g1 = [\cos(alpha az) \sin(alpha az) 0;...
    -sin(alpha az) cos(alpha az) 0;...
    0 0 1];
R g1 to g = [\cos(alpha el) 0 - \sin(alpha el);...
    0 1 0;...
    sin(alpha el) 0 cos(alpha el)];
R_b_{to_g} = R_g1_{to_g*R_b_{to_g}};
R g to b = R b to g';
% R_c_to_g
% Found on page 227 of Small Unmanned Aircraft
R g to c = [0 1 0;...]
    0 0 1;...
   1 0 0];
R c to g = R g to c';
% For simplicity, the three Rotation matrices are combined into one below
RbiRbgRcg = R_b_to_i * R_g_to_b * R_c_to_g;
```

### **Geolocation Algorithm**

The algorithm uses the line of sight vector in the camera frame and rotates it into the inertial frame. There, it is multiplied by a scaler (height) and divided by the projection of the transformed line of sight vector onto the unit vector k\_i in the inertial frame

```
P_i_mav = [Pn;Pe;Pd];
P_i_obj = P_i_mav + h*(RbiRbgRcg*l_cusp_c)/(dot(k_i,(RbiRbgRcg*l_cusp_c)));
```





```
disp("Coordinates of the MAV");
disp(P_i_mav);
disp("Coordinates of the Target");
disp(P_i_obj);
```

```
Coordinates of the MAV
20
20
-100

Coordinates of the Target
1.0e+05 *

3.2770
0.0012
0
```

Published with MATLAB® R2017a