

```
function dcm = LatLongAttToDCM(lat, lon)
% Convert euler angles to DCM
C_NE = [0 0 1; 0 1 0; -1 0 0];
C_NdN = [cos(lat) 0 sin(lat); 0 1 0; -sin(lat) 0 cos(lat)];
C_NNd = [1 0 0; 0 cos(lon) -sin(lon); 0 sin(lon) cos(lon)];

% Convert DCM
dcm = C_NNd * C_NdN * C_NE;

end
```

```
function dcm = eulerToDCM(yaw, pitch, roll)
% Convert euler angles to DCM angles in Radians.
C_Y = [cos(yaw) -sin(yaw) 0; sin(yaw) cos(yaw) 0; 0 0 1];
C_P = [cos(pitch) 0 sin(pitch); 0 1 0; -sin(pitch) 0 cos(pitch)];
C_R = [1 0 0; 0 cos(roll) -sin(roll); 0 sin(roll) cos(roll)];

% Convert DCM
dcm = C_R * C_P * C_Y;

end
```

```

function [lat, lon, alt] = cartesianToGeodetic(x, y, z)

% WGS84 ellipsoid parameters
a = 6378137; % Semi-major axis (meters)
f = 1/298.257223563; % Flattening
b = a * (1 - f); % Semi-minor axis (meters)

% Eccentricity squared
e2 = (a^2 - b^2) / a^2;

% Calculate longitude
lon = atan2(y, x);




% Calculate latitude
p = sqrt(x^2 + y^2);
theta = atan2(z * a, p * b);
lat = atan2(z + e2 * b * sin(theta)^3, p - e2 * a * cos(theta)^3);

% Calculate altitude
N = a / sqrt(1 - e2 * sin(lat)^2);
alt = p / cos(lat) - N;

end

```

Sample Times for 'translationalkinematicsii'

Color	Annotation	Description	Value
	Cont	Continuous	0
	FiM	Fixed in Minor Step	[0,1]
	Inf	Constant	Inf