73.019: 사진측량학특론

Lab #1: Basic Transformation, Data Import & Export, Visualization

석사과정 전일서

1-1. 회전행렬에서 3 개의 회전 각 값을 계산하는 식 유도하기

Ans) x 축에 대한 회전 각을 ω, y 축에 대한 회전 각을 ϕ, z 축에 대한 회전 각을 κ라고 한다면, 회전행렬에 대한 계산은 다음과 같이 정리되어 있다.

$$R = R_\omega \cdot R_\phi \cdot R_\kappa = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

$$= \begin{bmatrix} \cos\phi\cos\kappa & -\cos\phi\sin\kappa & \sin\phi \\ \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa & \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa & -\sin\omega\cos\phi \\ \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa & \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa & \cos\omega\cos\phi \end{bmatrix}$$

따라서 각 행렬 요소는 다음을 의미한다.

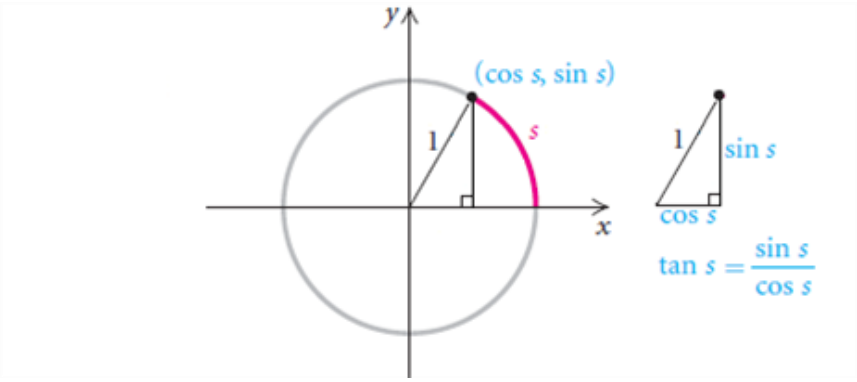
$r_{11} = \cos\phi\cos\kappa$
$r_{12} = -\cos\phi\sin\kappa$
$r_{13} = \sin\phi$
$r_{21} = \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa$
$r_{22} = \cos\omega\cos\kappa - \sin\omega\sin\phi\sin\kappa$
$r_{23} = -\sin\omega\cos\phi$
$r_{31} = \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa$
$r_{32} = \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa$
$r_{33} = \cos\omega\cos\phi$

각 행렬 요소 값을 알고 있다면, 세 축에 대한 회전 값은 다음과 같이 유도할 수 있다. 우선, r_{13} 과 $(\sin\phi)^2 + (\cos\phi)^2 = 1$ 을 이용하여 $\sin\phi, \cos\phi$ 값을 다음과 같이 표현한다.

$$\sin\phi = r_{13}$$
$$\cos\phi = \pm\sqrt{(1 - r_{13}^2)}$$

사진측량에 사용되는 드론을 생각해보면, 영상에 대한 자세 값은 0 ~ 360° 범위로 산출될 수 있다. 따라서 범위가 한정되지 않도록 ϕ, ω, κ 를 구하기 위해 단위원 개념을 사용하여 삼각함수를 정의한다. 단위원 기반의 삼각함수는 아래 그림과 같이 원점 O 를

중심으로 한 반지름이 1 인 원(단위원)을 생각하고, 그 단위원 상의 점 P(x, y)를 사용해서 정의하는 것을 의미한다.



ϕ, ω, κ 는 다음을 통해 얻을 수 있다.

$$\phi = \tan^{-1}\left(\frac{r_{13}}{\pm\sqrt{(1-r_{13}^2)}}\right)$$

$$\omega = \tan^{-1}\left(\frac{r_{23}}{-r_{33}}\right) or\ \omega = \tan^{-1}\left(\frac{-r_{23}}{r_{33}}\right)$$

$$\kappa = \tan^{-1}\left(\frac{r_{12}}{-r_{11}}\right) or\ \kappa = \tan^{-1}\left(\frac{-r_{12}}{r_{11}}\right)$$

다시 탄젠트 값을 결정하는 분모 값이 +인지 –인지의 부호에 따라 다음의 2 가지 경우의 수를 생각할 수 있다.

(i) $+\sqrt{(1-r_{13}^2)} = \cos\phi > 0$ 일 때,

$$\phi = \tan^{-1}\left(\frac{r_{13}}{+\sqrt{(1-r_{13}^2)}}\right), \omega = \tan^{-1}\left(\frac{-r_{23}}{r_{33}}\right), \kappa = \tan^{-1}\left(\frac{-r_{12}}{r_{11}}\right)$$

(ii) $-\sqrt{(1-r_{13}^2)} = \cos\phi < 0$ 일 때,

$$\phi = \tan^{-1}\left(\frac{r_{13}}{-\sqrt{(1-r_{13}^2)}}\right), \omega = \tan^{-1}\left(\frac{r_{23}}{-r_{33}}\right), \kappa = \tan^{-1}\left(\frac{r_{12}}{-r_{11}}\right)$$

1-2. 회전 각에서 회전 행렬 구하는 함수, 회전행렬에서 회전각 구하는 함수 만들기 (MATLAB 이용)

실행 파일: Lab1_1.m

함수 파일: A2R_RPY, A2R_OPK1, R2A_RPY, R2A_OPK1

Lab1_1.m

①Mavic pro 로 촬영한 영상을 불러온다. 2019 년 6 월 25 일 연도에서 촬영된 영상이다.

```
fid = fopen( 'mavic2_eoini_tm.txt', 'r');
% % 차례대로 영상이름 Easting, Northing, Altitude, Yaw, Pitch, Roll
imgexif = textscan(fid, '%s %f %f %f %f %f %f', 'CommentStyle','#');
fclose(fid);
```

②모든 영상에 대한 yaw, pitch, roll 값을 새로운 변수에 저장한다.

```
no_data = size(imgexif{2},1);
angles = zeros(no_data, 3);
for n = 1:no_data
    angles(n,:) = [imgexif{5}(n), imgexif{6}(n), imgexif{7}(n)];
end
```

③A2R_RPY 함수를 이용하여 회전행렬을 구한다.

```
% angle to rotation matrix
% ypr => pr(-y)
% R = Rz * Rx * Ry;
A2R_1 = cell(no_data, 1);
for n = 1:no_data
    A2R_1{n} = A2R_RPY(angles(n,:));
end
```

A2R_RPY 함수를 살펴보면, 3 개의 축에 대한 회전 값 roll, pitch, yaw 를 NED 좌표계로의 회전 행렬로 만드는 함수에 해당한다. 입력 순서가 yaw pitch roll 이라면, yaw, pitch 는 시계 반대 방향이 양의 방향으로 회전하도록, kappa 는 반시계 방향이 양의 방향으로 회전하도록 정의되어 있다. 또한, Rx 자리에는 pitch 가, Ry 자리에는 roll 이, Rz 자리에는 kappa 를 이용하도록 정의되어 있다.

```
om = ra(2);
ph = ra(1);
kp = -ra(3);
```

각 축에 대한 행렬 곱의 순서는 다음과 같이 정해져 있다.

```
Rx = [1 0 0; 0 cos(om) -sin(om); 0 sin(om) cos(om)];
Ry = [cos(ph) 0 sin(ph); 0 1 0; -sin(ph) 0 cos(ph)];
Rz = [cos(kp) -sin(kp) 0; sin(kp) cos(kp) 0; 0 0 1];

R = Rz * Rx * Ry;
```

④A2R_OPK1 함수를 이용하여 회전행렬을 구한다.

```
% ypr => ypr
% R = Rx * Ry * Rz;
A2R_2 = cell(no_data, 1);
for n = 1:no_data
    A2R_2{n} = A2R_OPK1(angles(n,:));
end
```

3 개의 축에 대한 회전 값 omega, phi, kappa 를 지상좌표계로의 회전 행렬로 만드는 함수에 해당한다. 입력 순서가 yaw pitch roll 이라면, 반시계방향 회전이 양의 방향으로 정의되어 있다. Omega 에는 yaw 가, phi 에는 pitch 가, kappa 에는 roll 이 입력되도록 되어 있다.

```
om = ra(1);
ph = ra(2);
kp = ra(3);
```

각 축에 대한 행렬 곱의 순서는 다음과 같이 정해져 있다.

```
Rx = [1 0 0; 0 cos(om) -sin(om); 0 sin(om) cos(om)];
Ry = [cos(ph) 0 sin(ph); 0 1 0; -sin(ph) 0 cos(ph)];
Rz = [cos(kp) -sin(kp) 0; sin(kp) cos(kp) 0; 0 0 1];

R = Rx * Ry * Rz;
```

⑤R2A_RPY 를 이용하여 다시 각 축에 대한 회전각을 구한다.

```
R2A_1 = zeros(no_data, 3);
for n = 1:no_data
    R2A_1_tmp = R2A_RPY(A2R_1{n});
    R2A_1_a = R2A_1_tmp(1,:);
    R2A_1_b = R2A_1_tmp(2,:);
end
```

1-1 에서 유도한 식이 구현되어 있다.

```
s_om = R(3,2);
c_om = zeros(2,1);
c_om(1) = sqrt(1-s_om^2);
c_om(2) = -sqrt(1-s_om^2);

kp = zeros(2,1);
ph = zeros(2,1);
kp(1) = atan2(-R(1,2), R(2,2));
ph(1) = atan2(-R(3,1), R(3,3));
kp(2) = atan2(R(1,2), -R(2,2));
ph(2) = atan2(R(3,1), -R(3,3));
```

A2R_RPY 에서 입력되었던 시계방향, 반시계방향과 omega, phi, kappa 의 자리, 행렬 곱의 순서가 고려되어 다시 다음과 같이 얻는다.

```
ra(:,1) = ph;
ra(:,2) = atan2(s_om, c_om);
ra(:,3) = -kp;
```

1-1 에서 확인 하였듯이 2 가지 경우의 값을 구할 수 있다.

⑥R2A_OPK1 을 이용하여 다시 각 축에 대한 회전각을 구한다.

```
% rotation matrix to angle
% R = Rx * Ry * Rz;
R2A_2 = zeros(no_data, 3);
for n = 1:no_data
    R2A_2_tmp = R2A_OPK1(A2R_1{n});
    R2A_2_a = R2A_2_tmp(1,:);
    R2A_2_b = R2A_2_tmp(2,:);
end
```

마찬가지로 1-1 에서 유도한 식이 구현되어 있고, A2R_OPK1 에서 입력되었던대로 고려되어 다시 다음과 같이 각 값을 얻을 수 있다.

```
ra(:,1) = om;
ra(:,2) = atan2(s_ph, c_ph);
ra(:,3) = kp;
```

2.a exif 태그에서 영상 위치/자세 값 추출하기

%copyright 강민

사용 언어: 파이썬

사용 라이브러리: pillow, csv 등

Extract data 함수 안에서 영상에 태깅 되어 있는 yaw, pitch, roll 값을 추출한다.

```
for filename, gps_dict_elem, xm_elem, model in tqdm(zip(self.img_base, self.gpsinfo_list, xm_list, self.phonemodel_list)):

    if len(self.xmlinfo_list)!=0:
        # Yaw, Roll, Pitch Extraction
        for yrp in ["Yaw", "Roll", "Pitch"]:
            target = "<Camera:" + yrp + ">(.*?)/100"
            m = re.search(target, str(xm_elem))
            if yrp == "Yaw":
                Yaw = eval(m.group()[12:])
            elif yrp == "Roll":
                Roll = eval(m.group()[13:])
            elif yrp == "Pitch":
                Pitch = eval(m.group()[14:])
    elif len(self.xmpinfo_list)!=0:
        target_yaw = 'FlightYawDegree="(.*?)"'
        target_pitch = 'FlightPitchDegree="(.*?)"'
        target_roll = 'FlightRollDegree="(.*?)"'
        yaw_part = re.search(target_yaw, str(xm_elem))
        pitch_part = re.search(target_pitch, str(xm_elem))
        roll_part = re.search(target_roll, str(xm_elem))
        Yaw = float(yaw_part.group()[17:-1])
        Pitch = float(pitch_part.group()[19:-1])
        Roll = float(roll_part.group()[18:-1])
    else:
        Yaw = -9999
        Pitch = -9999
        Roll = -9999
```

다음의 코드에서 GPS 값을 입력받는다.

```
# Lat, Lon, Altitude Extraction
if gps_dict_elem != "None":
    if gps_dict_elem.get("GPSLatitude") != None:
        lat = gps_dict_elem["GPSLatitude"][0][0] / gps_dict_elem["GPSLatitude"][0][1] + \
              gps_dict_elem["GPSLatitude"][1][0] / gps_dict_elem["GPSLatitude"][1][1] / 60 + \
              gps_dict_elem["GPSLatitude"][2][0] / gps_dict_elem["GPSLatitude"][2][1] / 3600
    else:
        lat = -9999
    if gps_dict_elem.get("GPSLongitude") != None:
        lon = gps_dict_elem["GPSLongitude"][0][0] / gps_dict_elem["GPSLongitude"][0][1] + \
              gps_dict_elem["GPSLongitude"][1][0] / gps_dict_elem["GPSLongitude"][1][1] / 60 + \
              gps_dict_elem["GPSLongitude"][2][0] / gps_dict_elem["GPSLongitude"][2][1] / 3600
    else:
        lon = -9999
    if gps_dict_elem.get("GPSAltitude") != None:
        alt = gps_dict_elem["GPSAltitude"][0] / gps_dict_elem["GPSAltitude"][1]
    else:
        alt = -9999
else:
    lat = -9999
    lon = -9999
    alt = -9999
```

저장 방식은 tsv 혹은 csv 형태가 있다.

2.b 지상좌표계에서 카메라 좌표계와 영상 바운더리 표현하기,

영상을 읽는다. 영상 자료는 2019 년 6 월 연도에서 mavic pro 로 취득되었다.
위치 값은 위경도 좌표계에서 중부원점 tm 좌표계로 변환되었다. flight yaw, pitch, roll 을 입력받아 카메라 자세를 변환하도록 한다.

②지상 좌표계에 표현된 카메라 위치/자세를 입력한다.
C_BG 는 차례로 Easting, Northing, Altitude 를 의미한다.
A_BG 는 차례로 yaw, pitch, roll 을 의미한다.
R_BG 는 차례로 A2R_RPY 함수로 계산된 각에서 행렬로 변환된 것을 의미한다.

```
%% 지상 좌표계에 표현된 카메라 위치/자세 입력
C_BG = zeros(no_data,3);
A_BG = zeros(no_data,3);
R_BG = cell(no_data,1);

for n = 1:no_data
    C_BG(n,:) = [imgexif{2}(n) imgexif{3}(n) imgexif{4}(n)];
    A_BG(n,:) = [imgexif{5}(n) imgexif{6}(n) imgexif{7}(n)];
    R_BG{n} = A2R_RPY(A_BG(n,:));
end
```

③카메라 렌즈와 영상이 평행하다고 가정이 되어 영상 각 4 개의 모서리의 좌표 값을 지상기준점으로 표현되게 구한다. 이 과정에서 초점거리, 픽셀사이즈와 해상도가 필요하다. 실제 단위 m 로 변환하려면 /1000 을 해야하지만, 그래프에 가시화하기 위해 임의로 단위를 더 크게 한다…

```
%% 지상 좌표계에 표현된 영상 위치
%mavic pro spec
p_size = (0.00241228)/10; %단위 : m
f= (8.8)/10; %단위 : m
f_mat = [0 0 f];
s_width = 5472;
s_height = 3648;
```

④영상좌표계에서 카메라 좌표계로 변환한다. 영상에서 4 개의 꼭짓점을 정의하고 각 점을 카메라 좌표계로 원점 이동한 뒤 좌표 변환한다.

```
% ICS -> PCS
u_l_pcs = zeros(no_data, 3);
d_l_pcs = zeros(no_data, 3);
u_r_pcs = zeros(no_data, 3);
d_r_pcs = zeros(no_data, 3);

for n=1:no_data
    u_l_pcs(n,:) = (u_l_ics-img_origin)*p_size-f_mat;
    d_l_pcs(n,:) = (d_l_ics-img_origin)*p_size-f_mat;
    u_r_pcs(n,:) = (u_r_ics-img_origin)*p_size-f_mat;
    d_r_pcs(n,:) = (d_r_ics-img_origin)*p_size-f_mat;

    u_l_pcs(n,2) = -u_l_pcs(n,2);
    d_l_pcs(n,2) = -d_l_pcs(n,2);
    u_r_pcs(n,2) = -u_r_pcs(n,2);
    d_r_pcs(n,2) = -d_r_pcs(n,2);
end
```

⑤카메라좌표계에서 지상좌표계로 변환한다. 영상의 4 꼭지점을 다시 지상좌표계 기준으로 표현하기 위해 회전변환과 이동변환을 수행한다.

```
%PCS -> GCS
u_l_gcs = zeros(no_data, 3);
d_l_gcs = zeros(no_data, 3);
u_r_gcs = zeros(no_data, 3);
d_r_gcs = zeros(no_data, 3);

for n=1:no_data
    u_l_gcs(n,:) = (R_BG{n}'*u_l_pcs(n,:)'+C_BG(n,:)')';
    d_l_gcs(n,:) = (R_BG{n}'*d_l_pcs(n,:)'+C_BG(n,:)')';
    u_r_gcs(n,:) = (R_BG{n}'*u_r_pcs(n,:)'+C_BG(n,:)')';
    d_r_gcs(n,:) = (R_BG{n}'*d_r_pcs(n,:)'+C_BG(n,:)')';
end
```

⑥지상좌표계에 표현된 카메라 좌표계와 영상 위치를 그린다. 여러장의 영상을 표현하기에 실제 영상의 크기가 너무 작아 한 장의 영상만 표현하였다.

```
pt=zeros(no_data, 3);
cs = {'r', 'g', 'b'};
img_x=cell(no_data, 1);
img_y=cell(no_data, 1);
img_z=cell(no_data, 1);
figure ()
hold on
for n = 1:no_data
    %% 카메라 좌표계 그리기
    for k = 1:3
        cstr = cs{k};
        pt(n,:) = C_BG(n,:)' + 0.005 * R_BG{n}(k,:)';
        plot3([C_BG(n,1), pt(n,1)], [C_BG(n,2), pt(n,2)], [C_BG(n,3), pt(n,3)], 'r-', 'LineWidth', 1, 'Color', cstr);
    end
    %% 영상 위치 그리기
    img_x{n} = [u_l_gcs(n,1) u_r_gcs(n,1) d_r_gcs(n,1) d_l_gcs(n,1) u_l_gcs(n,1)]';
    img_y{n} = [u_l_gcs(n,2) u_r_gcs(n,2) d_r_gcs(n,2) d_l_gcs(n,2) u_l_gcs(n,2)]';
    img_z{n} = [u_l_gcs(n,3) u_r_gcs(n,3) d_r_gcs(n,3) d_l_gcs(n,3) u_l_gcs(n,3)]';
    plot3(img_x{n}, img_y{n}, img_z{n}, 'b--', 'LineWidth', 4);
end
view(3)
grid on, axis equal
title('R(BG)')
xlabel('X'), ylabel('Y'), zlabel('Z')
```

아래는 결과 화면이다. 카메라 좌표계 위치/자세와 영상의 위치/자세를 표현하였다. 카메라 좌표계의 파란색 축이 z 축을, 빨강색 축이 x 축을, 초록색 축이 y 축을 나타내고 있다.

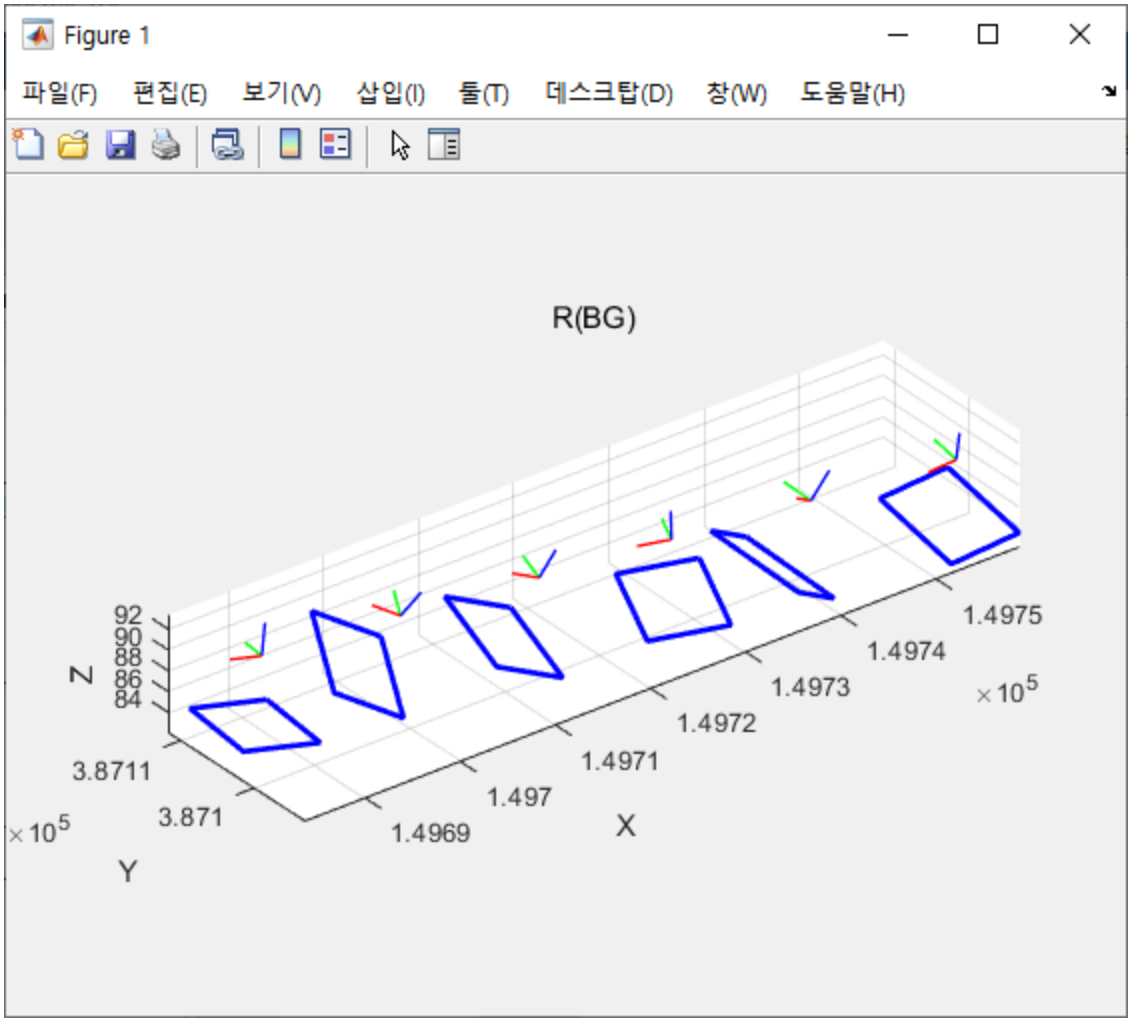


Figure 1
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
R(BG)
92
90
88
86
84
Z
3.8711
3.871
×10^5
Y
1.4969
1.497
1.4971
1.4972
1.4973
1.4974
1.4975
×10^5
X