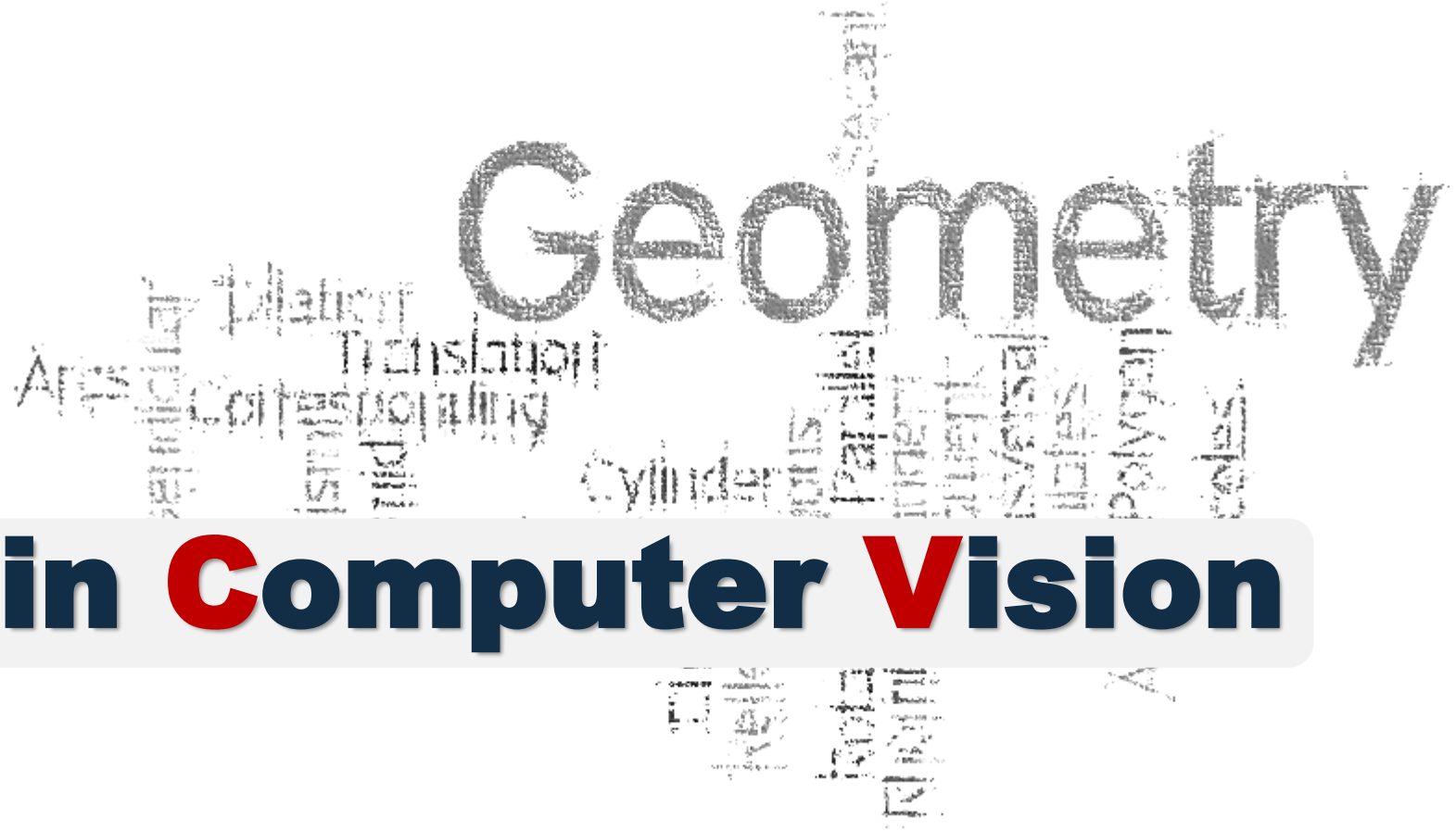
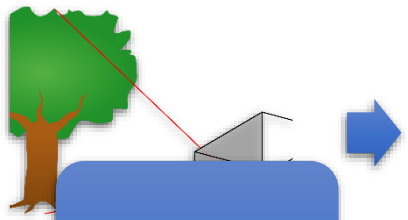


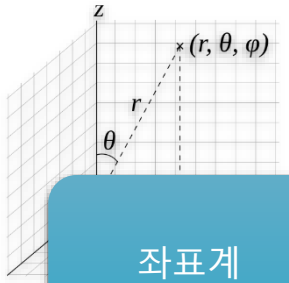
Geometry in Computer Vision

Jeon Hyun Ho

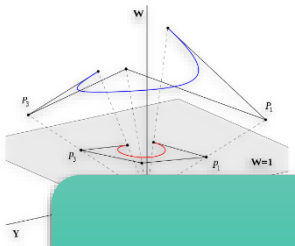
Contents



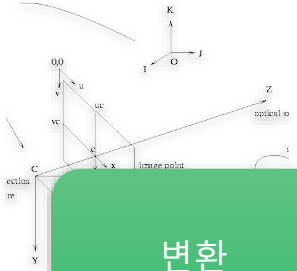
카메라 모델
Pin hole
camera



좌표계
Coordinate
System



동차 좌표
Homogeneous
coordinates

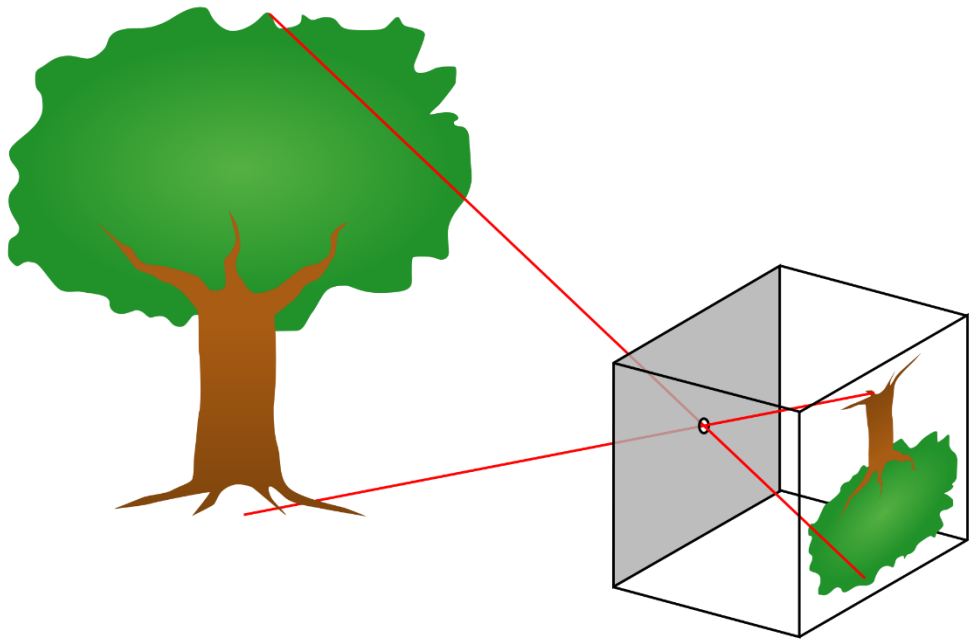


변환
Transformation
- 2D, 3D

영상 기하
Imaging
geometry

에피폴라 기하
Epipolar
geometry

Pin hole camera model



Pin hole camera model

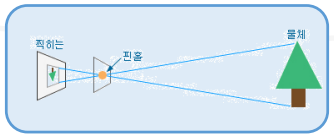


image plane

Normalized image plane

(X_c, Y_c, Z_c)

초점

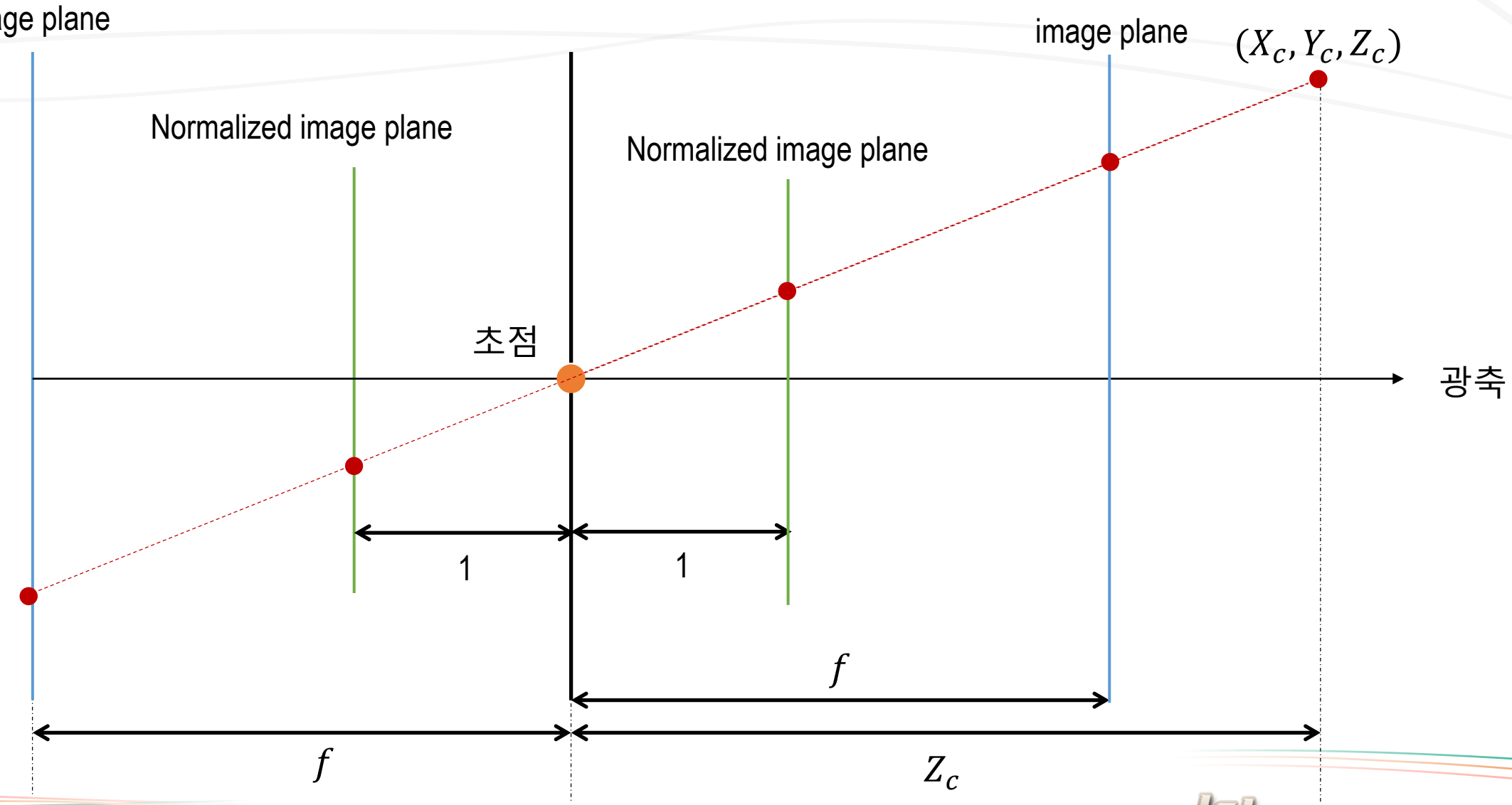
광축

1

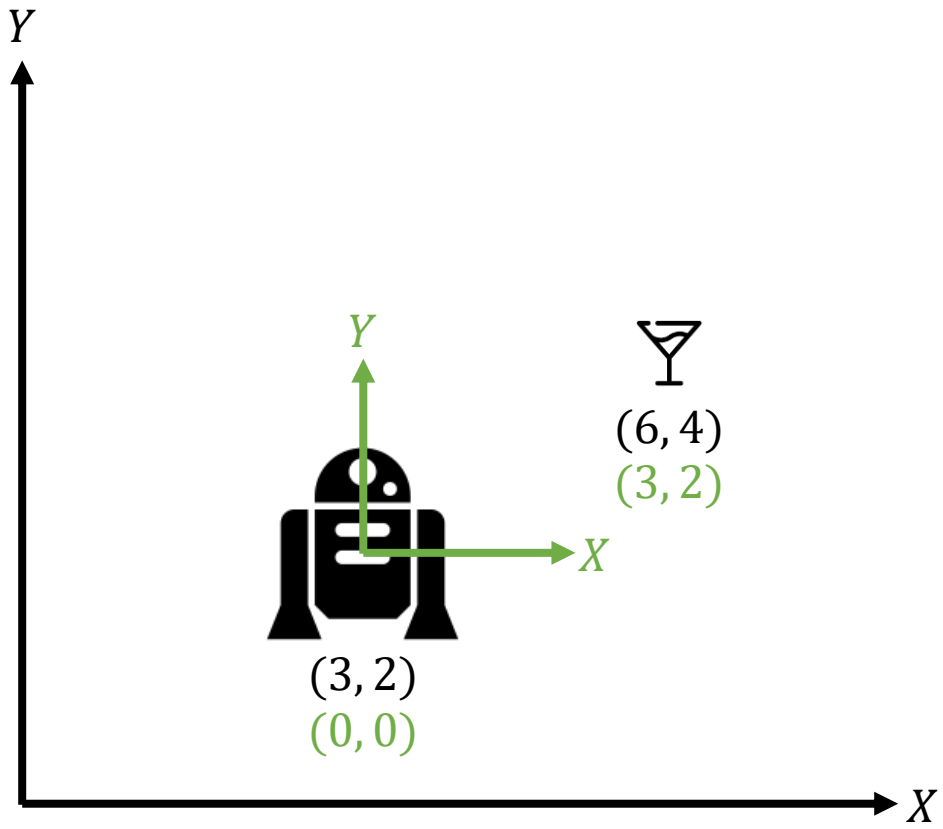
f

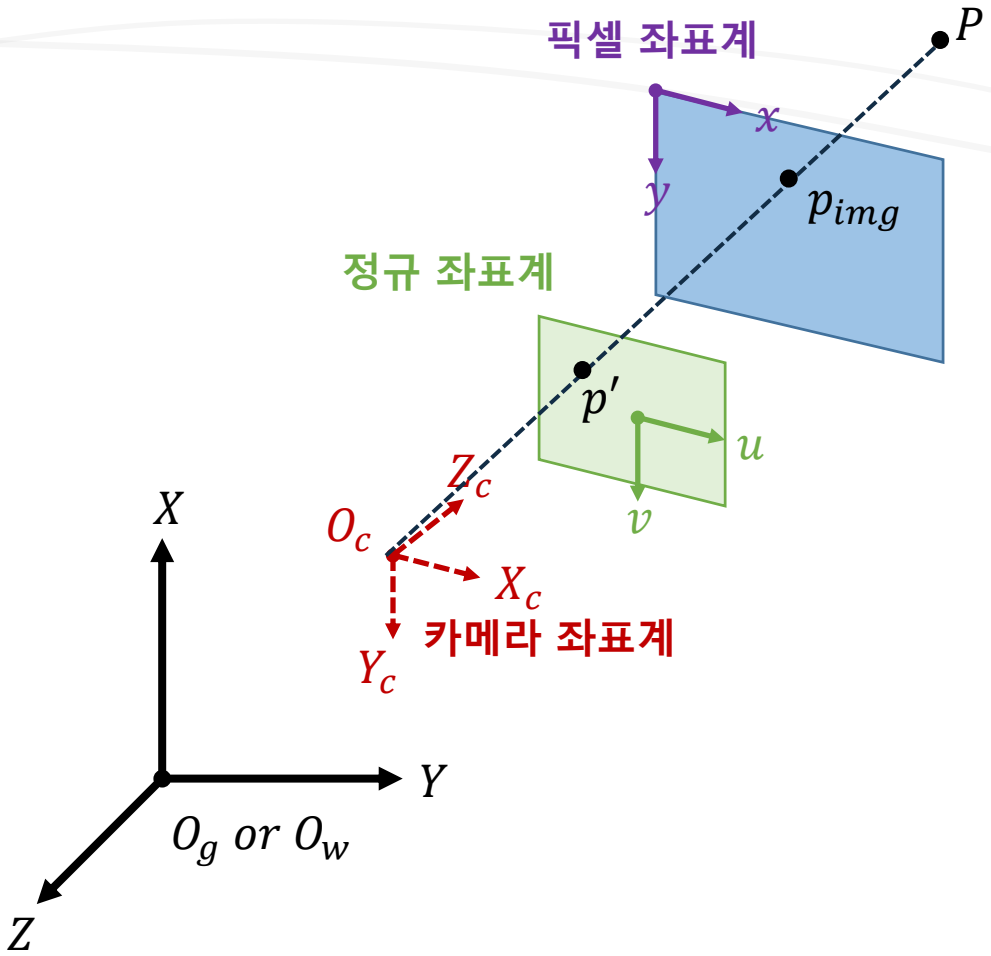
Z_c

Pin hole camera model



Coordinate System

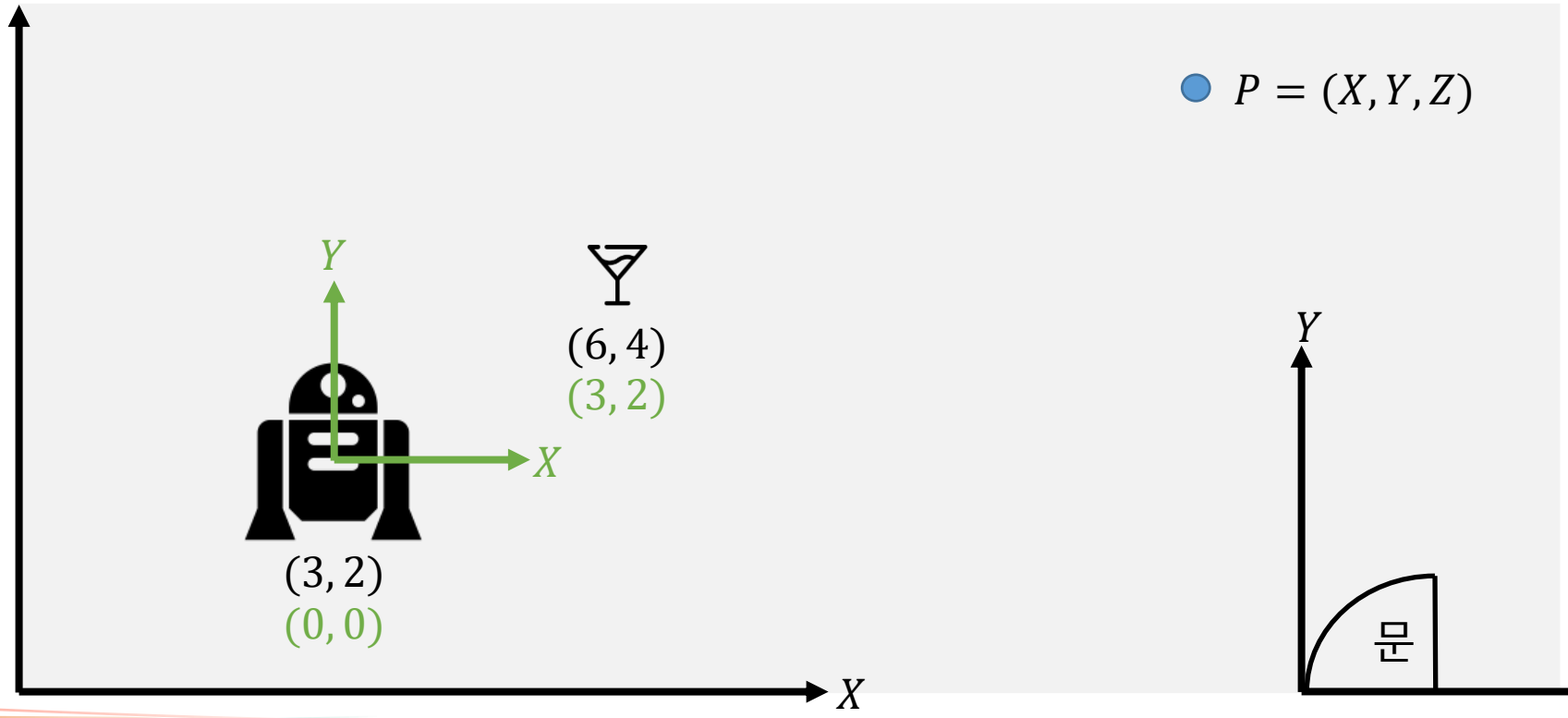




Coordinate System

- **World Coordinate System**

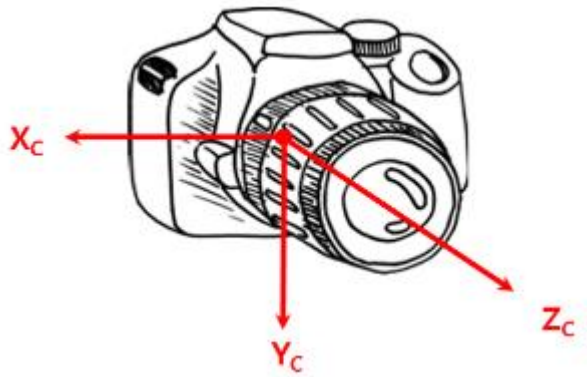
  - 사물의 위치를 표현할 때 기준으로 사용하는 좌표계 중 하나
  - 정해져 있지 않으며 임의로 설정하여 사용하는 좌표계



Coordinate System

- **Canera Coordinate System**

  - 카메라를 기준으로 하는 좌표계
  - 카메라의 초점(렌즈의 중심)을 원점으로 설정

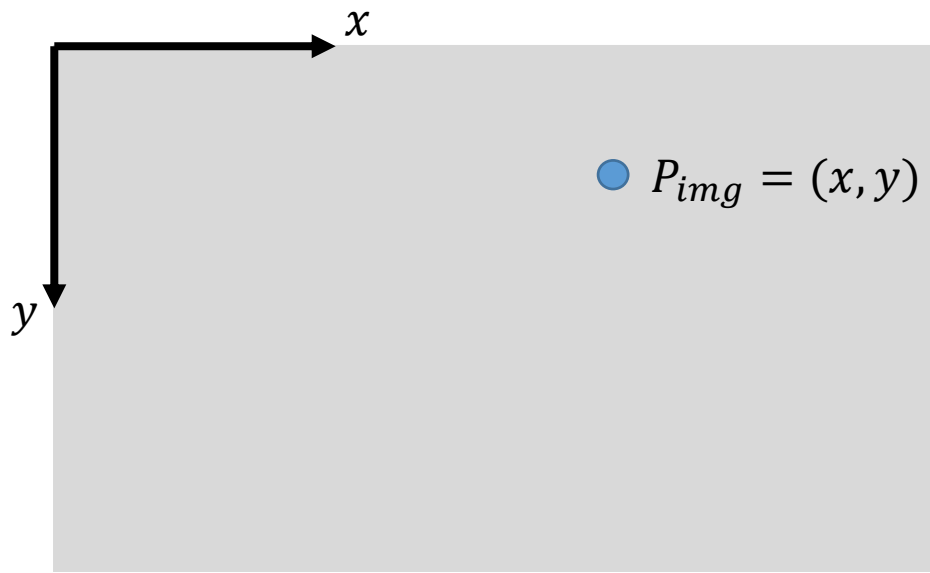


- $P_c = (X_c, Y_c, Z_c)$

Coordinate System

- **Pixel Image Coordinate System**

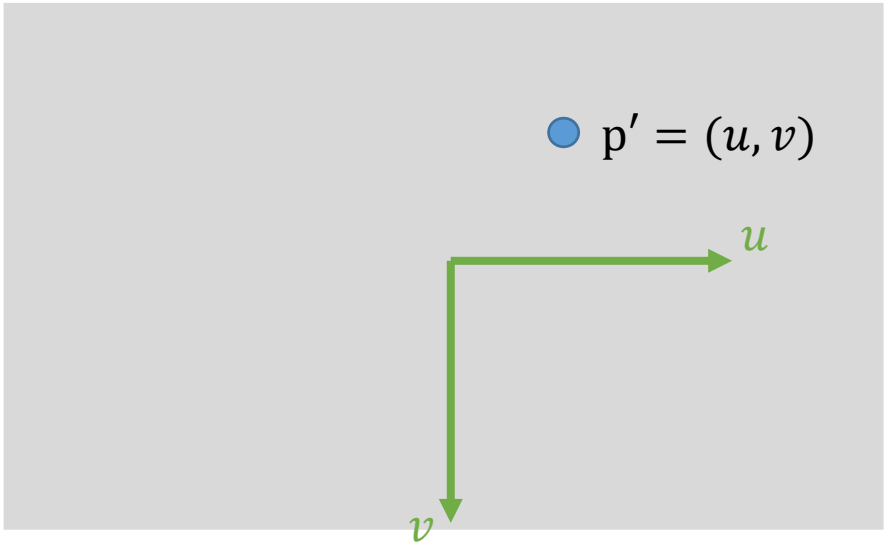
  - 영상에 대한 좌표계
  - 왼쪽 상단 모서리를 원점으로 함
  - OpenCV의 Mat의 접근 방식과 다르므로 주의 (x, y) != (row, col)



Coordinate System

- **Normalized Image Coordinate System**

  - 카메라 내부 파라미터의 영향을 제거한 이미지 좌표계
  - 이미지 평면을 평행 이동시켜 카메라 초점과의 거리가 1인 지점으로 옮긴 이미지 평면
  - 광축과의 교점이 원점



$$p_{img} = Kp'$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$x = f_x u + c_x$$
$$y = f_y v + c_y$$

$$u = (x - c_x)/f_x$$
$$v = (y - c_y)/f_y$$

Homogeneous Coordinates

- **Homogeneous Coordinates**

  - (x, y)를 $(x, y, 1)$로 표현하는 형태
  - 일반적으로 0이 아닌 임의의 상수 w에 대해 (x, y)를 (wx, wy, w)로 표현
  - 스케일이 무시되며 한 좌표에 대한 표현이 무한함
  - 원 좌표를 구하려면 끝 자리가 1이 되도록 스케일을 변환

$$(x, y, w) \longrightarrow \left(\frac{x}{w}, \frac{y}{w}, 1\right) \Rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)$$

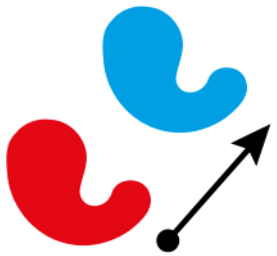
Transformation

- **Rigid Transformation**

  - 유클리디언 변환이라고도 표현함
  - 형태와 크기를 유지하며 위치와 방향만 바뀌는 변환

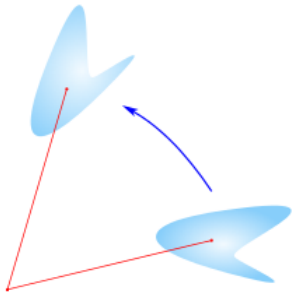
- Translation (평행 이동)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



- Rotation (회전)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



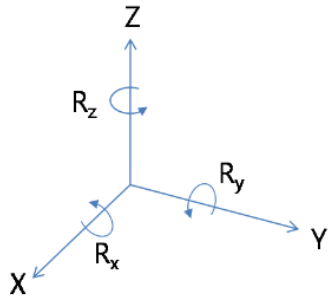
- Rigid transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix}$$

Transformation

- **Rigid Transformation**

- 3D transformation



- 변환 행렬

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_z(\theta_3)R_y(\theta_2)R_x(\theta_1)$$

- 3D 변환식

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

- Homogeneous 좌표계 표현

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{[R|t]} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Imaging Geometry

- **Image Projection model**
  - [R|t] : 월드 좌표계를 카메라 좌표계로 바꾸는 rigid 변환 행렬
  - Tpers(1) : 카메라 좌표계 상의 3D좌표를 정규 이미지 평면에 투영 시키는 projection 행렬
  - K : 카메라 내부 파라미터 행렬, 정규 이미지 좌표를 픽셀 좌표로 변환

$$s\begin{bmatrix}x\\y\\1\end{bmatrix} = T\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \quad \text{T를 분해하면} \quad s\begin{bmatrix}x\\y\\1\end{bmatrix} = KT_{pers}(1)[R|t]\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

$$s\begin{bmatrix}x\\y\\1\end{bmatrix} = \begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Imaging Geometry

- **Image Projection model**

  - 최종 모델

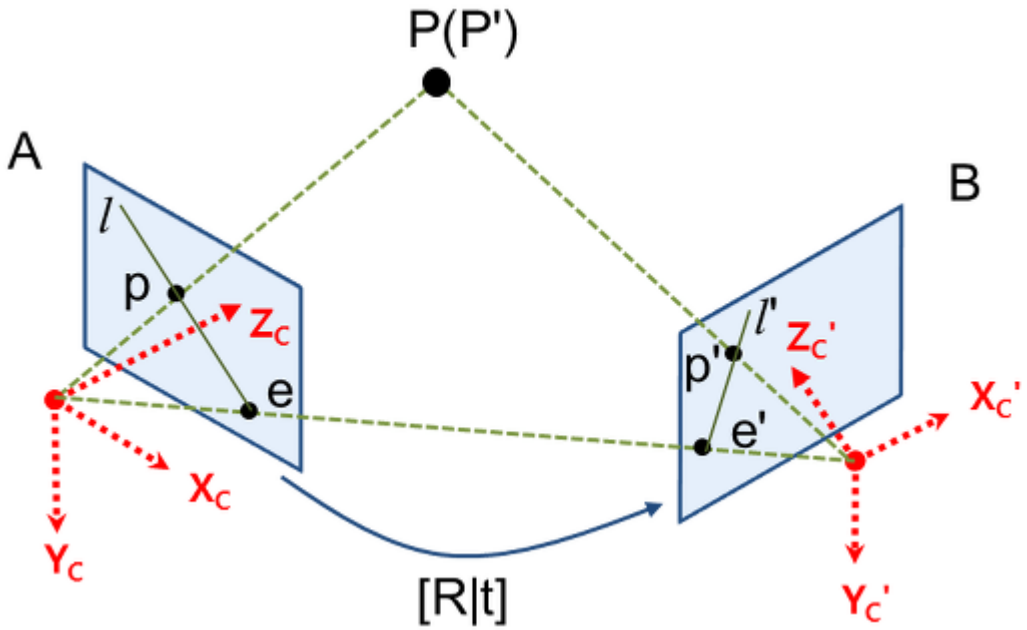
$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- K : 카메라내부 파라미터를 나타냄 (intrinsic parameter)
- [R|t] : 카메라 외부 파라미터를 나타냄 (extrinsic parameter)

Epipolar Geometry

- 2 – view vision



Epipolar Geometry

- alignment stereo vision

$$(x_2 - x_1) : f = B : Z$$

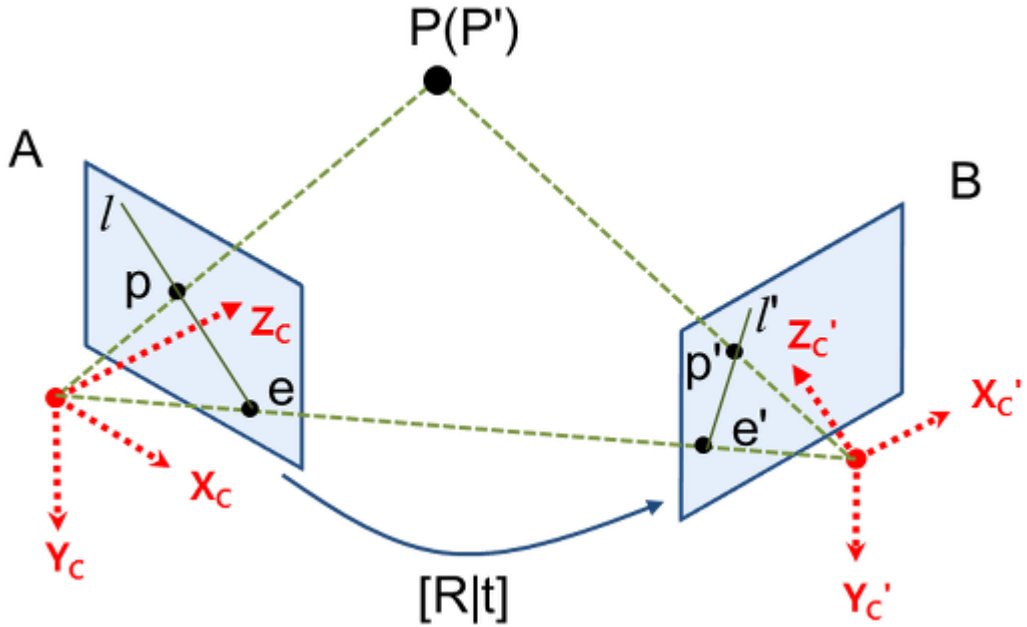
$$fB = (x_2 - x_1)Z$$

$$\frac{fB}{x_2 - x_1} = Z$$

$$Z = \frac{fB}{x_2 - x_1}$$

Epipolar Geometry

- Epipolar Geometry



Epipolar Geometry

- **Essential Matrix**

  - 3D공간 상의 한점 P가 카메라 A, B에 각각 p, p'으로 투영 되었을 때 $p'^T Ep = 0$을 만족하는 E가 항상 존재

$$p'^T Ep = 0$$

$$[u' \quad v' \quad 1]E\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

  - Mat **findEssentialMat**(InputArray **points1**, InputArray **points2**, double **focal=1.0**, Point2d **pp=Point2d(0, 0)**, int **method=RANSAC**, double **prob=0.999**, double **threshold=1.0**, OutputArray **mask=noArray()**) :

    1) 두 이미지 사이의 E를 계산하여 반환함
    2) 5쌍 이상의 매칭 이미지 좌표쌍과 카메라 파라미터를 입력받음
    3) 픽셀좌표를 입력

Epipolar Geometry

- **Fundamental Matrix**

  - 카메라 파라미터까지 포함한 두 이미지의 픽셀 좌표 사이의 기하학적 관계를 표현하는 행렬

$$p'^{T}_{img} F p_{img} = 0 \qquad E = K'^{T} F K \qquad E = K^{T} F K$$

$$[x' \;\; y' \;\; 1] F \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \qquad F = (K'^{T})^{-1} E K^{-1} \qquad F = (K^{T})^{-1} E K^{-1}$$

  - Mat **findFundamentalMat**(InputArray **points1**, InputArray **points2**, int **method=FM_RANSAC**, double **param1=3.**, double **param2=0.99**, OutputArray **mask=noArray**()) :

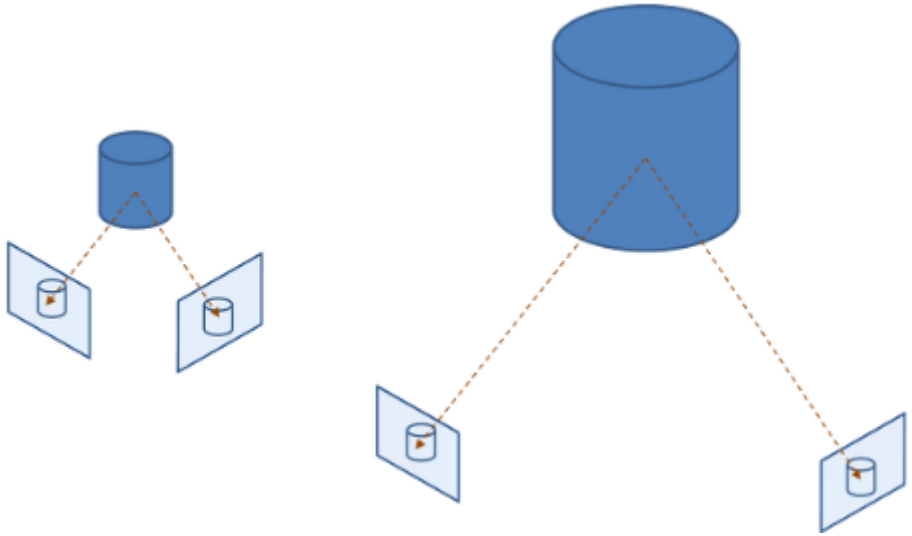
    1) 두 이미지 사이의 F를 계산하여 반환함
    2) 8쌍 이상의 매칭 이미지 좌표쌍을 입력 (method따라 7쌍도 가능)

Epipolar Geometry

- **Scale Problem**
  - 에피폴라 기하에서는 두 영상간의 관계를 보기 때문에 스케일을 결정할 수 없다는 문제가 있음

    => E는 R, t로 구성됨

    회전변환 R이 3자유도 스케일을 무시한 평행이동 t가 2자유도, 총 5쌍의 매칭점이 필요



Epipolar Geometry

- **recoverPose – [R|t] 추출**

  - int **recoverPose**(InputArray **E**, InputArray **points1**, InputArray **points2**, OutputArray **R**, OutputArray **t**, double **focal=1.0**, Point2d **pp=Point2d(0, 0)**, InputOutputArray **mask=noArray()**) :

    1) 입력된 E와 두 이미지상의 매칭쌍으로부터 두 이미지의 상대적인 [R|T] 관계를 추출해 줌

    2) 부가적인 기하학적 조건 검사를 통해 [R|t]를 유일하게 결정하여 반환함

    3) 항상 |t| = 1

Epipolar Geometry

- **triangulatePoints – feature 4D point 추출**

  - void **triangulatePoints**(InputArray **projMatr1**, InputArray **projMatr2**, InputArray **projPoints1**, InputArray **projPoints2**, OutputArray **points4D**) :

    1) 두 카메라의 Projection Matrix (K[R|t])를 입력 받음

    2) 두 카메라 상의 매칭쌍

Epipolar Geometry

- **solvePnP – [R|t] 추출**

  - bool **solvePnP**(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, bool useExtrinsicGuess=false, int flags=ITERATIVE) :

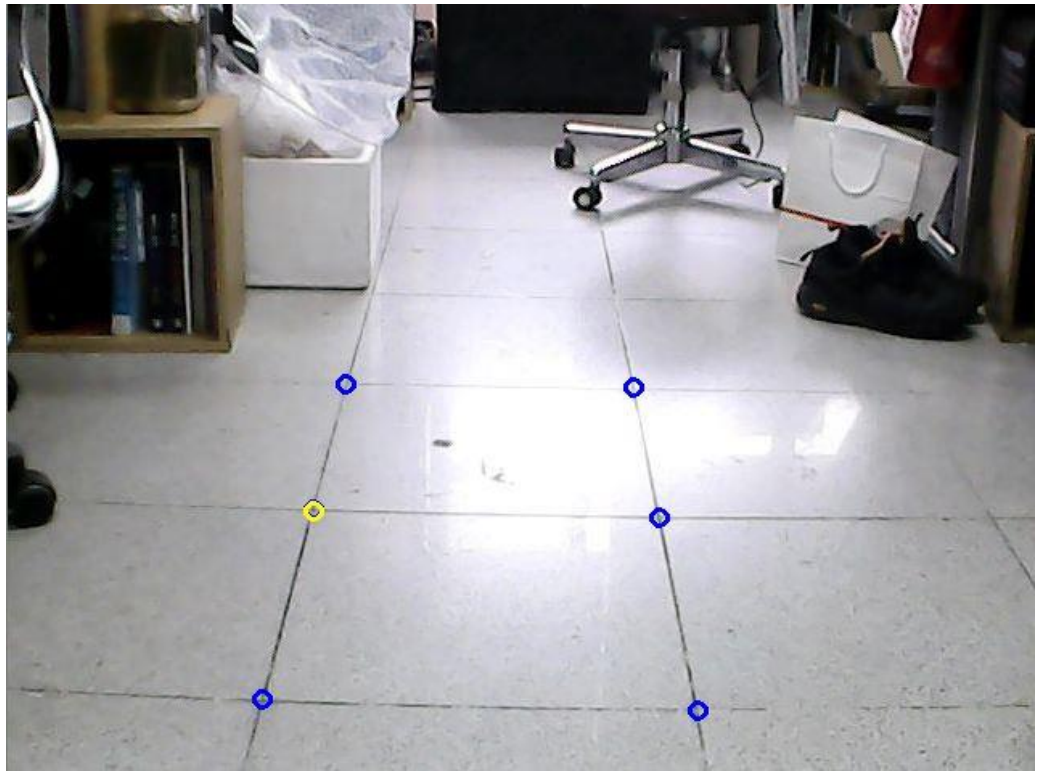
  - void **solvePnPRansac**(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, bool useExtrinsicGuess=false, int iterationsCount=100, float reprojectionError=8.0, int minInliersCount=100, OutputArray inliers=noArray(), int flags=ITERATIVE) => 반환함

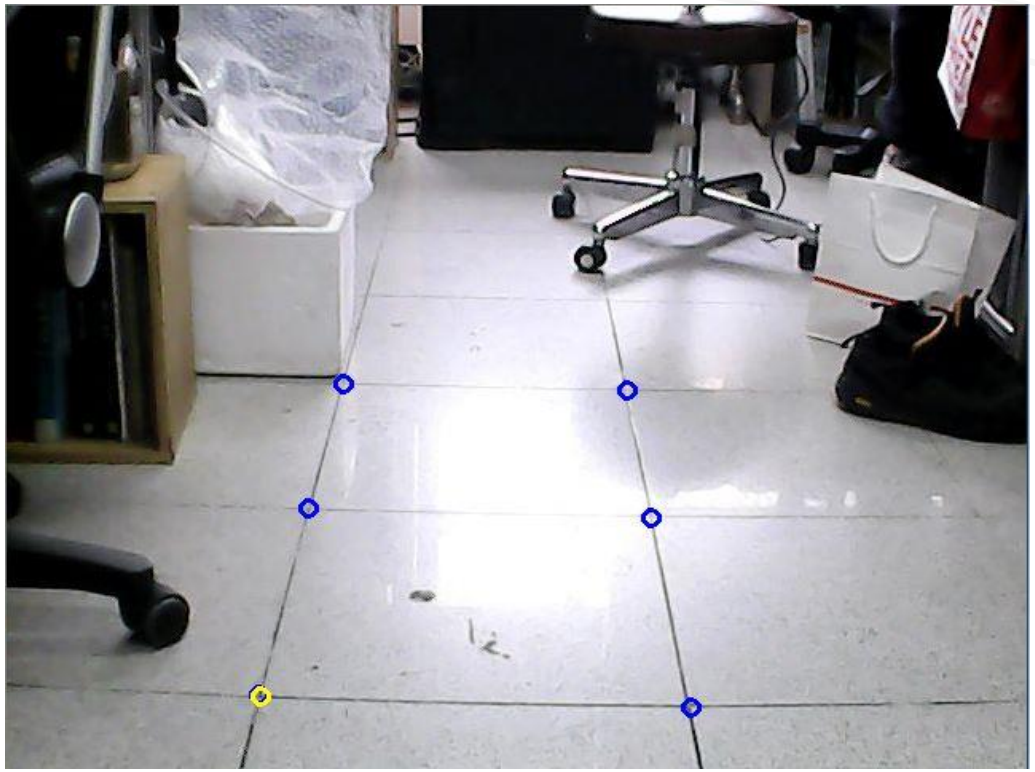
    1) 한대의 카메라내의 2D 특징들과 해당 특징들의 3D 좌표

    2) 이론적으로 최소 4개의 특징이 필요 (K[R|t] : 3x4 Mat)

Experiments

- solvePnPRansac

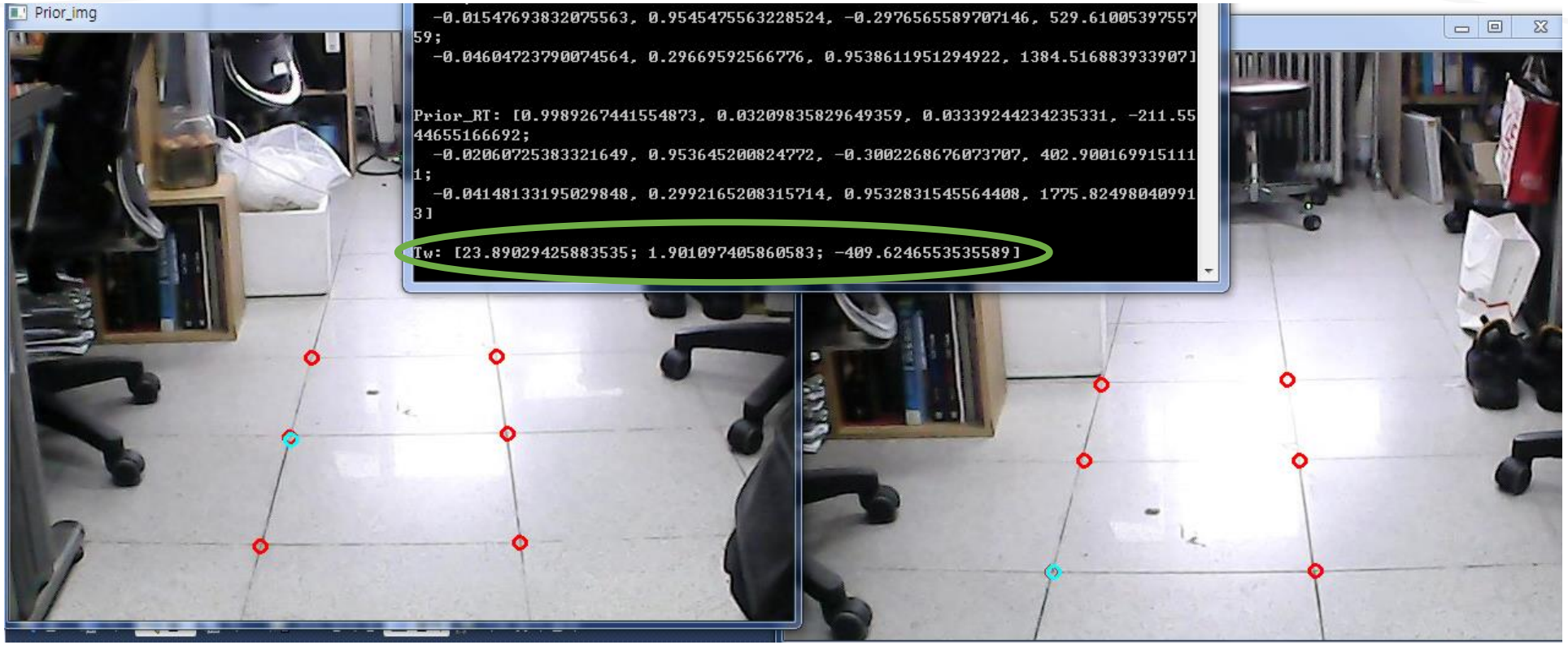




이동 전 → 1타일 이동 (40cm) → 이동 후

Experiments

- solvePnPRansac



결과

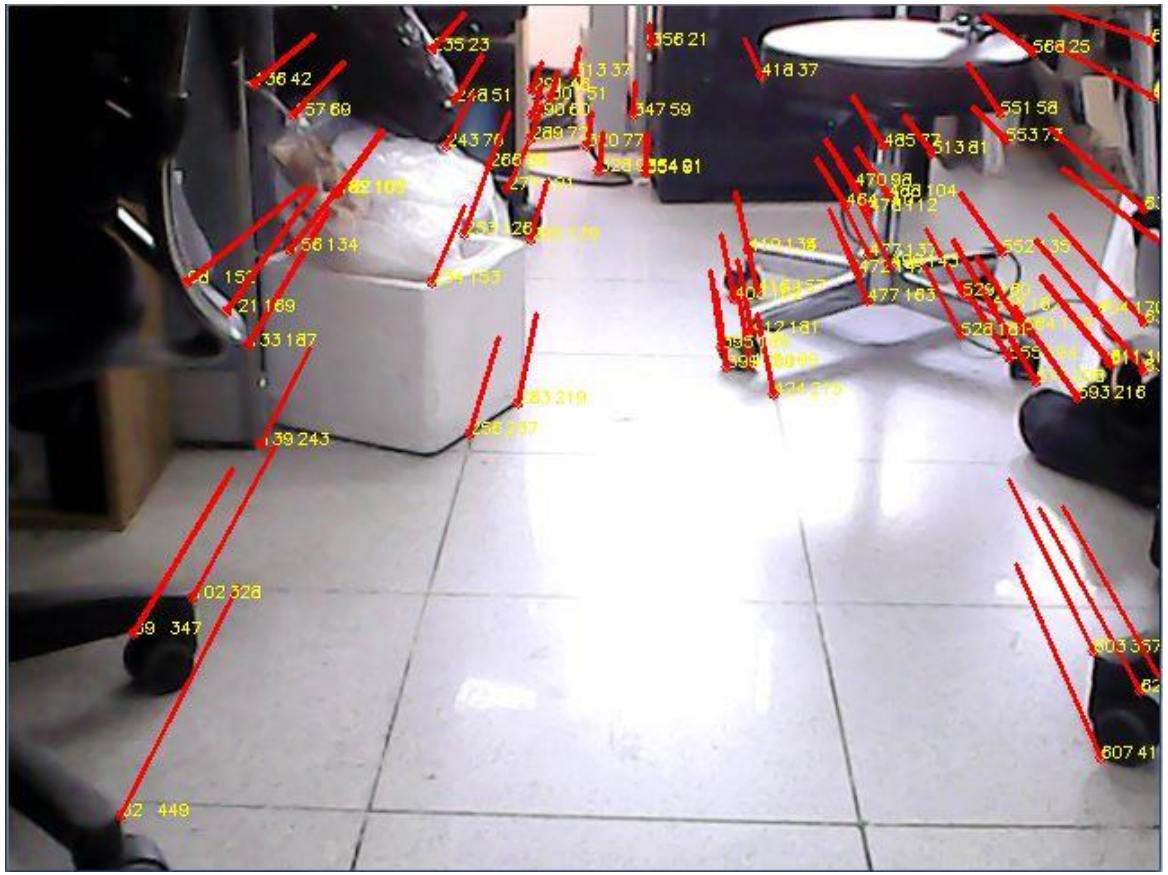
Experiments

- Find E – recoverPose - triangulatePoints

Feature Detection – SIFT, SURF, FAST, GFTT

Experiments

- Find E – recoverPose - triangulatePoints

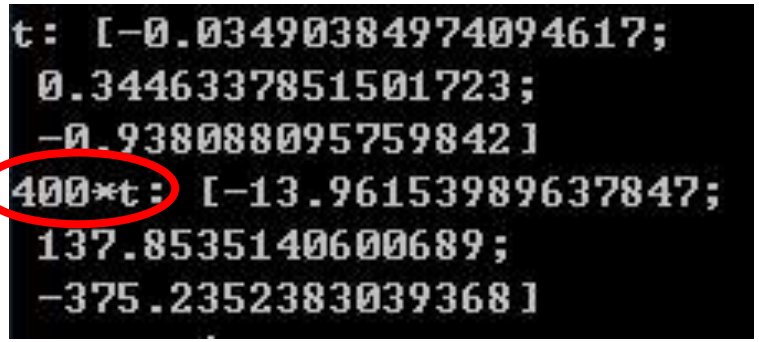


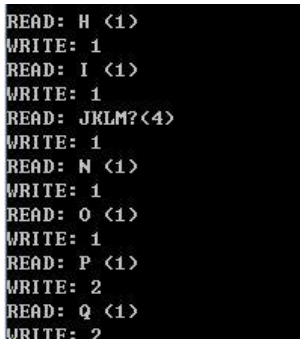
Feature Tracking - OpticalFlowPyrLK
Feature Matching - KnnMatch

Experiments

- **Find E – recoverPose - triangulatePoints**

```
Point2d pp(CamBridge.GetCx(), CamBridge.GetCy());
E = findEssentialMat(vecPtpre23, vecPtcur23, CamBridge.GetFocalLength(), pp, RANSAC, 0.999, 1.0, E_mask);
Scalar num_Inlier_E = sum(E_mask);
int inlier = recoverPose(E, vecPtpre23, vecPtcur23, R, t, CamBridge.GetFocalLength(), pp, E_mask, 50);
```



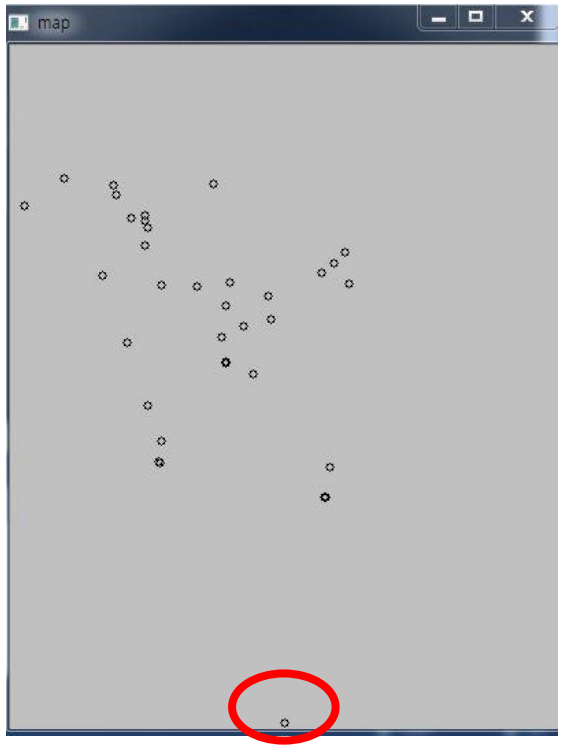


<= 시리얼 통신으로 엔코더의 값을 받아 곱해주는 방식으로 해결

Experiments

- Find E – recoverPose - triangulatePoints

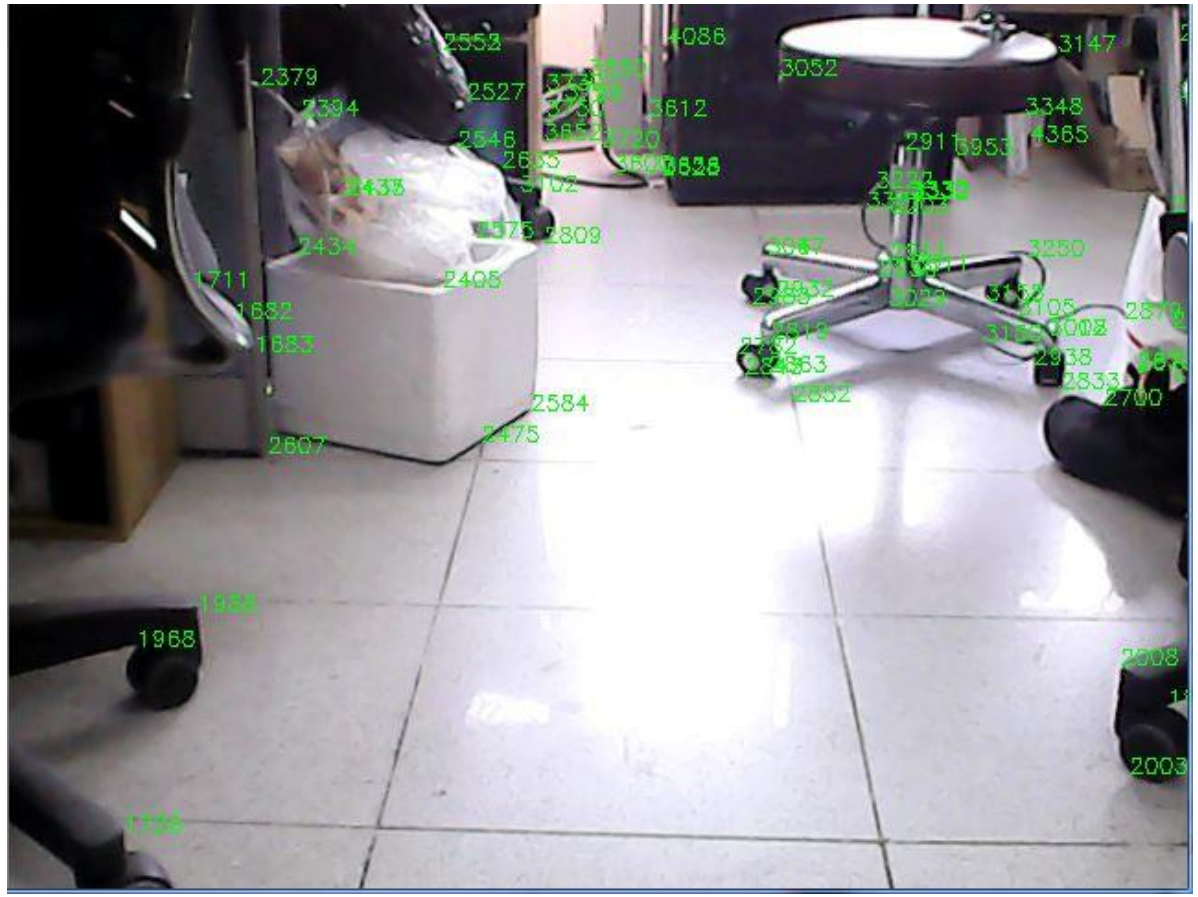
```
triangulatePoints(matP_ref, P, vecPtcur23, vecPtpre23, tmp4D);
```



triangulatePoints로 얻어진 각 특징의 3D좌표

Experiments

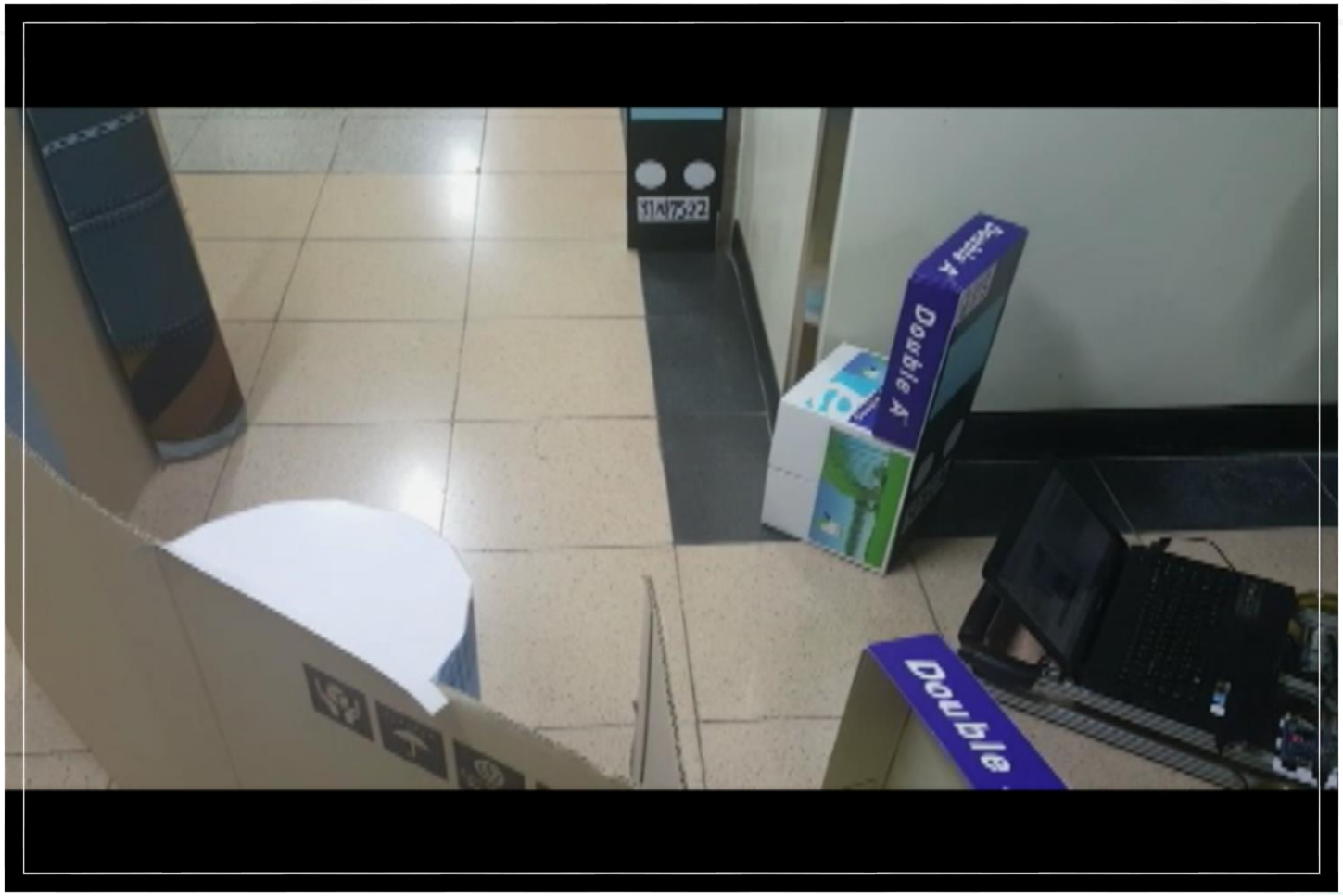
- Find E – recoverPose - triangulatePoints



triangulatePoints 함수로 얻어진 각 특징의 3D좌표를 이용하여 거리 계산

Experiments

- 영상



Q&A