컴퓨터비전개론

Chap 1. Image Filtering

• separable filter
- rank = 1
- gaussian filter : input의 가운데 픽셀값을 강조하는 효과
- sobel convoultion kernel
: x_dir(vertical edges), y_dir(horizontal edges)

***x* direction**

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

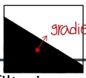
***y* direction**

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

• median filter
- filter 내의 숫자를 sorting한 후 정렬한 것
- salt-and-pepper noise를 지울 수 있음

• Image edges
- intensity가 급격히 변하는 곳
- gradient의 방향은 edge의 수직인 방향

Computing Image Gradients



gradient $\nabla = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$

▪ 1. Select your favorite derivative filtering.

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

▪ 2. Convolve with the image to compute derivatives.

$$\frac{\partial f}{\partial x} = S_x \otimes f \qquad \frac{\partial f}{\partial y} = S_y \otimes f$$

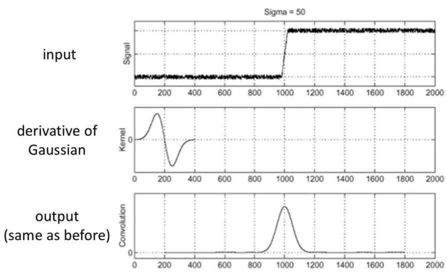
▪ 3. Form the image gradient,
and compute its direction and magnitude.

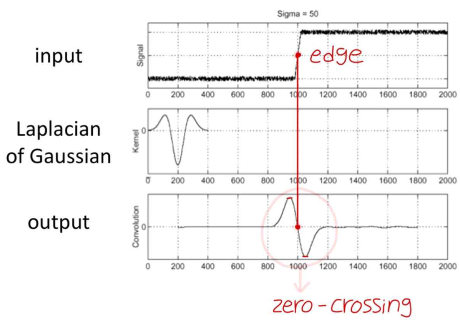
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right) \qquad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient direction magnitude

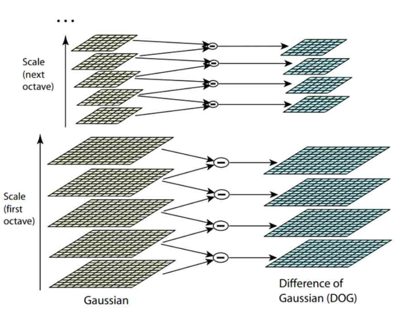
• Canny Edge Detector
① compute x and y gradient ② find magnitude and orientation of gradient
③ non-maximum suppression ④ hysteresis thresholding

• DoG(Derivative of Gaussian) vs LoG(Laplacian of Gaussian)





★ Laplacian을 DoG를 이용하여 근사하는 법



DoG는 입력영상에 Gaussian 필터를 점진적으로 적용하여 블러링(blurring)시킨 이미지들에서 인접 이미지들간의 차(subtraction) 영상을 의미하며 이론적으로는 LoG(Laplacian of Gaussian) 필터를 적용한 것과 거의 동일한 결과를 갖습니다.

Chap 2. Image Pyramids

• Aliasing
: sampling 시 서로 다른 신호를 구별할 수 없게 만드는 효과
- 원인
① sampling rate mismatch
② moire : object이 한 픽셀 중 R, G, B 한 곳에 집중되게 matching
③ temporal aliasing : FPS(frame per second)

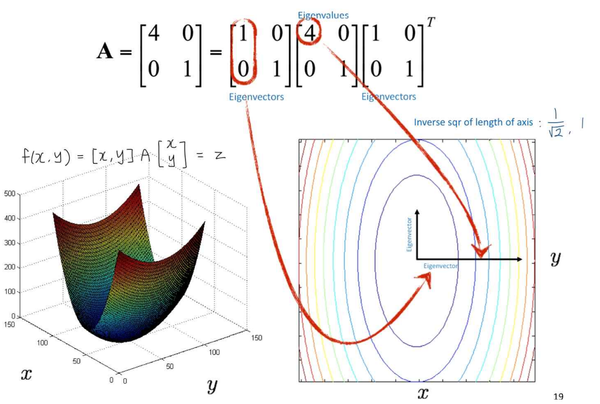
• Anti-Aliasing
- Oversample the signal : 한 픽셀에 여러번 sampling 한 후 평균값을 구함
- smooth ths signal : aliasing을 일으키는 요소를 제거, 이는 정보손실을 유발하지만 aliasing을 줄일 수 있음

• Gaussian Image Pyramid
- 왜 필요한가? anti-aliasing 하기 위해 smoothing(gaussian filter) 후 delete(행, 열 삭제)하는데 반복 작업일 경우 오래 걸림
- mip map을 만들면 원래 용량의 4/3배 증가함
- 하지만 reconstruct를 하면 화질이 많이 떨어짐 -> super resolution을 해야함

• Laplacian Image Pyramid
- 각 레벨에서 blurred image 대신에 residual을 남겨서 reconstruct할 때도 화질이 복원이 되도록 함

Chap 3. Corner Detection

• Paraboloid



★ Eigenvector & Eigenvalues
: 행렬 A를 선형변환으로 봤을 때, 선형변환 A에 의한 변환 결과가 자기 자신의 상수배가 되는 0이 아닌 벡터를 eigenvector라 하고 이 상수배 값을 eigenvalue라 한다.
- λ : 행렬 A의 고유값
- v : 행렬 A의 λ에 대한 고유벡터

$$A\mathbf{v}=\lambda\mathbf{v} \quad \text{---(1)}$$

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}=\lambda\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

★ Eigendecomposition (only 정방행렬)
행렬 A의 고유벡터들을 열벡터로 하는 행렬을 P, 고유값들을 대각원소로 하는 대각행렬을 Λ라 하면 다음 식이 성립

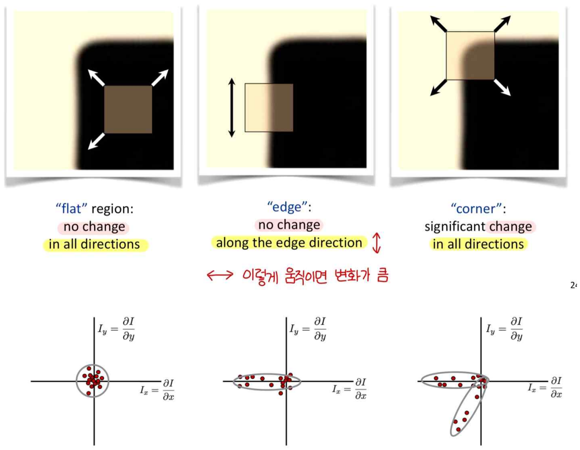
$$AP=P\Lambda$$
$$A=P\Lambda P^{-1}$$
: 행렬 A는 고유벡터들을 열벡터로 하는 행렬과 고유값을 대각원소로 하는 행렬의 곱으로 대각화 분해 가능
- eigendecomposition을 알면 det(A), A의 거듭제곱, 역행렬, 대각합(trace), 행렬의 다항식을 쉽게 계산 가능
- 가능 조건
① n x n 정방행렬 A가 n개의 일차독립(linearly independent)인 고유벡터를 가져야 함.
② $A^T = A$ (모든 i, j에 대해 $a_{ij} = a_{ji}$)인 대칭행렬(symmetric matrix)
: 특히, 직교행렬로 대각화가 가능함

$$A=P\Lambda P^{-1}$$
$$=P\Lambda P^{T} \quad \text{--- (24)}$$
단, $PP^T=E$ $(P^{-1}=P^T)$

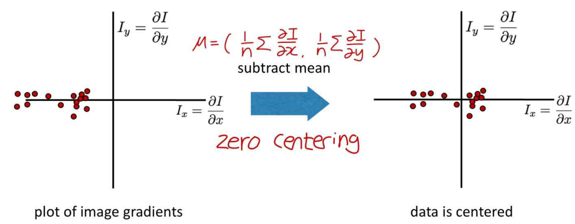
★ 직교행렬(orthogonal matrix)
: 자신의 전치행렬(transpose)를 역행렬로 갖는 정방행렬으로 각각의 열벡터의 크기는 1이며 열벡터끼리의 곱은 0임.

- Harris Corner Detector (only grayscale image)

① compute image gradients over small region



② subtract mean from each image gradient



→ 다양한 데이터를 같은 기준으로 판단 가능

★ Covariance Matrix

$$\mathrm{Var}[X] = \begin{bmatrix} \mathrm{Var}[X_1] & \mathrm{Cov}[X_1, X_2] \\ \mathrm{Cov}[X_2, X_1] & \mathrm{Var}[X_2] \end{bmatrix}$$

- 분산 : x들이 평균을 중심으로 얼마나 흩어져 있는지
- 공분산 : x1, x2의 흩어진 정도가 어떤 상관관계를 가지고 흩어졌는지

③ compute the covariance matrix of gradient

$$\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

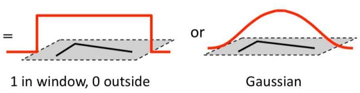
- why? change of intensity for the shift [u, v]을 구하기 위해서

$$E(u,v) = \sum_{(x,y)} w(x,y)[\,I(x+u,y+v) - I(x,y)\,]^2$$

: Error function, Window function, Shifted intensity, Intensity

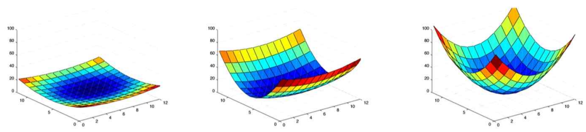
Window function $w(x,y) =$

유저가 보고 싶은 영역만 값을 가짐



← $I(x+u, y+v) \approx I(x,y) + I_x u + I_y v$ 대입 (Taylor Series)

$$= u^2 \sum_{(x,y)} w(x,y) I_x^2 + 2uv \sum_{(x,y)} w(x,y) I_x I_y + v^2 \sum_{(x,y)} w(x,y) I_y^2 = [u, v] M \begin{bmatrix} u \\ v \end{bmatrix} \quad \left(M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right)$$



: E(u, v) 시각화

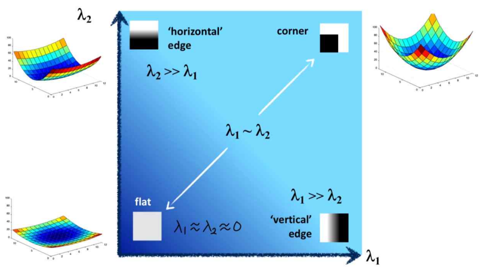
★ Taylor Series

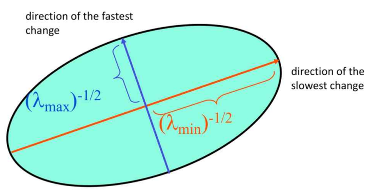
$$f(x) = p_\infty(x)$$

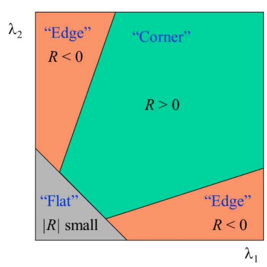
$$p_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k$$

: 좌변과 우변이 모든 x에 대해 같은 것이 아니라 x = a 근처에서만 성립

④ compute eigenvector and eigenvalues







⑤ use threshold on eigenvalues to detect corners : threshold(R) = det(M) = k x $(\mathrm{trace}(M))^2$

⑥ Nonmaximal Suppression

- Harris detector는 영상의 평행이동, 회전변화에는 불변(invariant)이고 affine 변화, 조명(illumination) 변화에도 어느 정도는 강인성을 가짐. 하지만 영상의 크기(scale) 변화에는 영향을 받음.

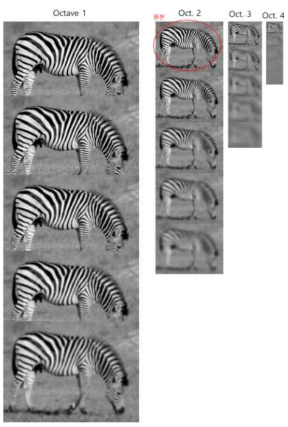
Chap 4. Feature Descriptors

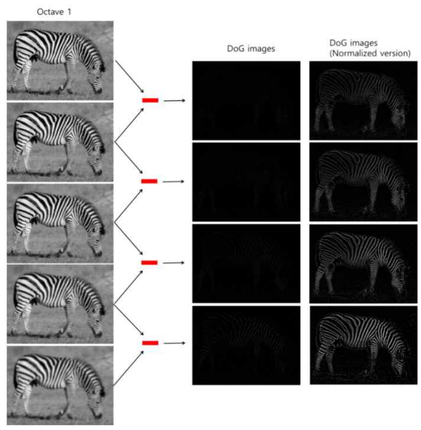
• 전통적인 feature descriptors
① Image Gradient = binary descriptor : intensity에 강함. 하지만 이동/회전에 약함
② Color Histogram : scale/rotate에 강함. 하지만 색이 유사한 물체와 혼동하기 쉬움
③ Spatial Color Histogram : deformation에 강함. 하지만 rotate에 약함
④ Orientation Normalization

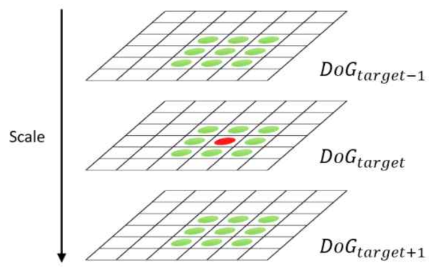
• GIST Descriptor → Rough spatial distribution of image gradients 구해짐
① Compute filter responses (filter bank of Gabor filters = Directional Edge Detector)
- gabor filter 예시 : $e^{\frac{-x^2}{2\sigma^2}\sin(2\pi\omega x)}$, $e^{\frac{-x^2}{2\sigma^2}\cos(2\pi\omega x)}$
② Divide an image patch on 4x4 cells
③ Compute filter response average for each cell : Descriptor 개수 = 4x4xN (N = filter bank 수)

• SIFT (Scale Invariant Feature Transform)
: feature pointer인 Blob을 찾아(feature detector) 벡터화시킨 feature descriptor 만들기
- 기존의 Harris 코너가 영상의 스케일 변화에 민감한 문제를 해결하기 위하여 Laplacian을 근사한 DoG(Difference of Gaussian)를 기반으로 이미지 내에서 뿐만 아니라 스케일 축(이미지 피라미드)으로도 코너성이 극대점을 찾음
- 스케일에 불변인 특징점을 찾으면 입력 이미지의 스케일이 어떻게 주어지더라도 해당 특징점을 찾아낼 수 있음

◉ Feature Detector
① Convolve image with scale-normalized Laplacian(blob filter) at several scales

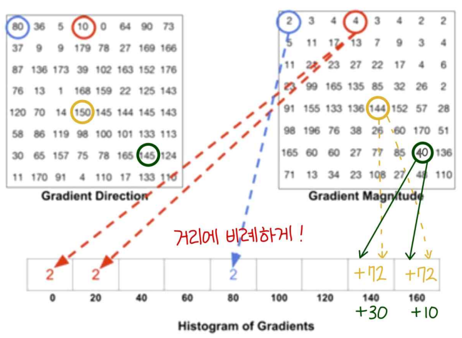


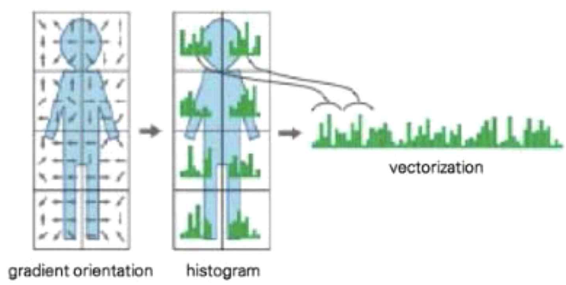




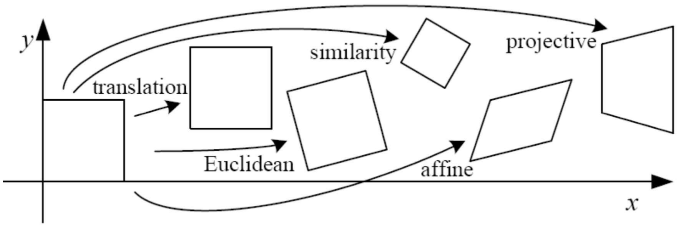
② Find maxima of squared Laplacian response in scale-space
③ Eliminate the edge responses based on Harris response function (det(M) = k x (trace(M))^2)

◉ Feature Description = HOG(Histogram of Oriented Gradients) Descriptor
① keypoint 주변의 16개 pixel을 4 x 4 cell로 만들어줌
② 한 cell마다 Histogram of Oriented Gradients를 구함
③ concatenate and l2 normalization





Chap 5. 2D Transformations



- Euclidean : rotation + translation
- Similarity : scaling + rotation + translation
- Affine : scaling + shearing + rotation + translation
- projective = homography

Transformation	Matrix	#DoF	Preserves	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}$	3	lenghts	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}$	4	angles	
affine	$\begin{bmatrix} A \end{bmatrix}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}$	8	straight lines	

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \beta_x & 0 \\ \beta_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

shearing

Chap 6. Image Homography

• Solving for Homography matrix

① For each correspondence, create 2 x 9 matrix A_i

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\rightarrow \begin{aligned} x'(h_7x + h_8y + h_9) &= (h_1x + h_2y + h_3) \\ y'(h_7x + h_8y + h_9) &= (h_4x + h_5y + h_6) \end{aligned}$$

$$\rightarrow \quad A_i h = 0$$

$$A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}, \quad h = [\, h_1 \; h_2 \; h_3 \; h_4 \; h_5 \; h_6 \; h_7 \; h_8 \; h_9 \,]^T$$

② Concatenate into single 2n x 9 matrix A

③ Homogeneous linear least squares problem : Compute SVD of $A = U\Sigma V^T$

④ Store singular vector of the smallest singular value $h = v_{\hat{i}}$

⑤ Reshape to get H

• Robust Feature-based Alignment

① Extract feature (detection & description)

② Compute potential matches

③ RANSAC (RANdom Sample Consensus) Loop

- Randomly select a four point correspndences.
- Compute H
- Count inliers to the current H
- Keep H if largest number of inliers

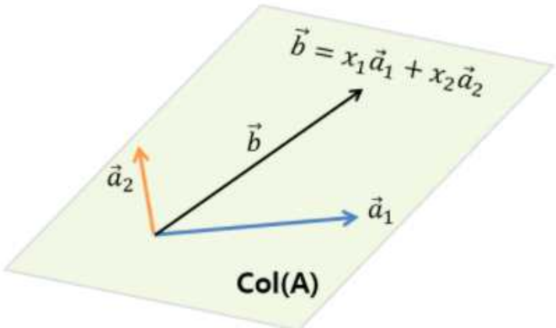
④ Recompute H using all inliers

★ Least Square Method : 데이터의 residual2의 합을 최소화하도록 모델의 파라미터를 구하는 방법

Ax = b에서,

- A 역행렬 존재 : x = A^{-1}b
- **A 역행렬 존재 X** : x = A^{+}(A의 pseudo inverse)b, 이는 ‖Ax-b‖를 최소화하는 해임.
- $A = U\Sigma V^T$ → A의 pseudo inverse(A^{+}) = $V\Sigma^{+}U^T$

(Σ^{+} : 원래의 Σ에서 0이 아닌 singular value들의 역수를 취한 후 transpose를 시킨 행렬)



A의 열(column)을 이루는 열벡터의 생성공간(span)인 A의 열공간 col(A)에 포함되어 있는 $\vec{b}$를 구하려면 $\vec{a_1}, \vec{a_2}$를 얼마만큼 조합해주어야 하는가의 문제

하지만 역행렬이 존재하지 않는다는 것은 span에 $\vec{b}$가 존재하지 않는다는 것으로 $\vec{b}$와 가장 가까우면서 $\vec{a_1}, \vec{a_2}$의 선형결합을 통해 얻을 수 있는 최적의 $\vec{p}$는 $\vec{b}$를 col(A)에 정사형 시킨 것

- 한계점 : outlier가 포함된 데이터에서는 적용하기 힘든 방법임. 왜냐하면 전체 데이터의 residual2 합을 최소화하기 때문에 outlier의 residual도 같이 줄이려고 하다보면 전혀 잘못된 근사 결과를 낼 수 있음. 따라서, outlier가 존재하는 경우에는 RANSAC, LMedS, M-estimator 등과 같은 robust한 파라미터 추정 방법 사용해야 함

★ Homogeneous Least Square Method

Ax = 0에서,

- 특이값에 0이 포함된 경우

: 만일 A의 특이값들 중 0이 존재하면 0인 특이값에 대응하는 right singular vector (V의 열벡터)는 Ax = 0을 만족하는 0이 아닌 해가 된다. 만일, 0의 값을 갖는 특이값이 여러개 존재할 경우에는 대응되는 right singular vector들이 모두 Ax = 0의 해가 된다. 또한 이들 해의 임의의 일차결합도 역시 해가 된다.

- 특이값이 모두 양수인 경우

: 특이값이 모두 양수인 경우에는 Ax = 0을 정확히 만족하는 해는 X = 0을 제외하고는 존재치 않는다. 따라서, 이 경우에는 근사적으로 해를 구해야 하는데 ‖AX‖를 최소로 하는 근사해는 A의 최소 singular value에 대응하는 right singular vector이다.

★ Eigen Decomposition

: 행렬 A를 자신의 고유벡터들을 열벡터로 하는 행렬과 고유값을 대각원소로 하는 행렬의 곱으로 대각화 분해한 것

- 고유값 분해가 가능하려면 행렬 A가 n개의 일차독립인 교유벡터를 가져야 함
- 대칭행렬($A^T = A$)의 고유값 분해 : 항상 교유값 분해가 가능하며, 직교행렬($A^{-1} = A^T$)로 대각화가 가능함

★ SVD (Singular Value Decompostion)

: 정방행렬이 아닌 n x m 행렬에 대해서도 Eigen Decomposition이 가능함

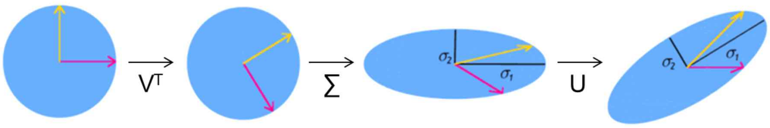
$$A = U\Sigma V^T$$

- U : AA^T를 Eigen Decomposition해서 얻어진 orthogonal matrix → U의 열벡터 : A의 left singular vector
- V : A^TA를 Eigen Decomposition해서 얻어진 orthogonal matrix → V의 열벡터 : A의 right singular vector
- Σ : AA^T, A^TA를 Eigen Decomposition해서 나오는 eigenvalue^2들을 대각원소로 하는 m x n 직사각 대각행렬

= A의 특이값(singular value)

- AA^T, A^TA은 symmetric matrix이므로 Eigen Decomposition이 가능하며 orthogonal matrix로 대각화 가능
- AA^T, A^TA의 공통의 고유값 $\sigma_1^2 > \sigma_2^2 > .... > \sigma_s^2 > 0$을 구한 후 이들의 square root를 취한 $\sigma_1 > \sigma_2 > .... > \sigma_s > 0$이 A의 singular value들이고 이들을 대각원소로 하는 m x n 직사각 대각행렬이 Σ임

★ SVD의 기하학적 표현



- orthogonal matrix(U, V)의 기하학적 의미 : 회전변환(rotate transformation) or reflected 회전변환
  det(AA^T) = det(A)det(A^T) = det$(A)^2$ = 1이므로 det(A)는 항상 +1 or −1임. 만일 det(R)=1라면 이 직교행렬은 회전변환을 나타내고 det(R)=-1라면 reflected 회전변환을 나타냄
- diagonal maxtrix(Σ)의 기하학적 의미 : 각 좌표 성분으로의 scale transformation

★ RANSAC 알고리즘의 하이퍼 파라미터 설정
- 파라미터 : 샘플링 과정을 몇 번 (N) 반복할 것인지, inlier와 outlier의 경계를 (T) 어떻게 정할 것인지

① N
- RANSAC이 성공하기 위해서는 N번의 시도 중 적어도 한번은 inlier들에서만 샘플 데이터가 뽑혀야 함.
- RANSAC 반복회수를 N, 한번에 뽑는 샘플 개수를 m, 입력 데이터들 중에서 inlier의 비율을 α라 하면 N번 중 적어도 한번은 inlier에서만 샘플이 뽑힐 확률 $p = 1-(1-\alpha^m)^N$

② T
- inlier들의 residual 분산을 σ라 할 때, T = 2σ or 3σ 정도로 잡음
- inlier들로만 구성된 실험 데이터들을 획득하고, inlier 데이터들에 대해서 Least Square Method을 적용하여 가장 잘 근사되는 모델을 구함. 이렇게 구한 모델과 inlier들과의 residual ($r_i = |y_i - f(x_i)|$)들을 구한 후, 이들의 분산을 구해서, 이에 비례하게 T를 결정

Chap 7. Geometric Camera Models

• Pinhole Camera

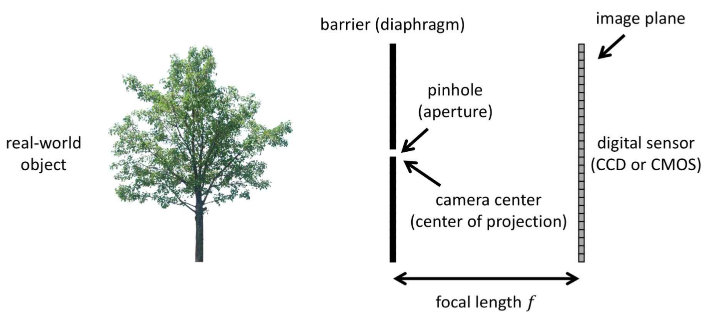
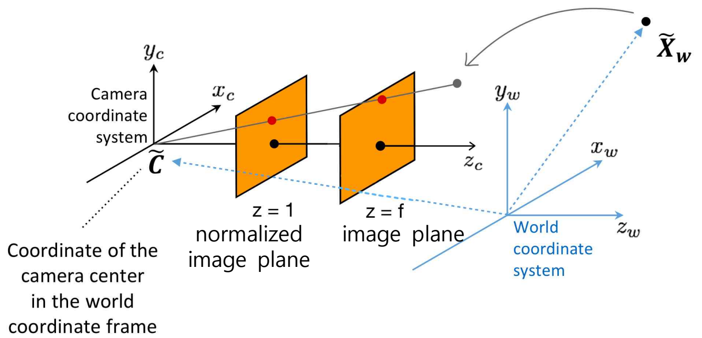


image plane이 실제론 pinhole 뒤에 있지만, 상이 뒤집어지는 것을 방지하여 pinhole 앞에 있다고 가정할 것임

3D → 2D 될 때,
- 직선, incidence 유지
- 각도, 길이 유지 못함

• Camera Matrix



$$\boldsymbol{x} = \mathbf{PX}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image coordinates 3 x 1 | camera matrix 3 x 4 | homogeneous world coordinates 4 x 1

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{array}\right]$$

intrinsic parameters | extrinsic parameters

- Camera Matrix는 10개의 degress of freedom가 존재, 만약 skew도 고려한다면 11개

• Extrinsic Parameter
: world 좌표계에서 camera 좌표계로 변환하는 과정에서 필요한 파라미터로 translation과 rotation 변환 포함

• Intrinsic Parameter

: camera 좌표계에서 image 좌표계로 변환하는 과정에서 필요한 파라미터로 이를 거치면 normalized image plane 에서의 좌표를 알 수 있음. 이미지 좌표계는 왼쪽 아래에 위치하므로 주점(p_x, p_y)만큼 이동하며, 픽셀의 영향을 해결하기 위해 초점거리(f)를 곱하여 instrinsic matirix을 완성함

\+ skew : 카메라 렌즈 왜곡

• Normalized image plane을 구하는 이유

: 동일한 장면을 동일한 위치와 동일한 각도에서 찍더라도 사용한 카메라에 따라서 또는 카메라 세팅에 따라서 서로 다른 영상을 얻게 됨. 따라서, 이러한 요소를 제거한 normalized image plane(z=1)에서 공통된 기하학적 특성을 분석하고 이론을 수립.

• Camera Calibration

$$P = K\,[\,R \mid t\,] = K\,[\,R \mid -Rc\,] = [\,M \mid -Mc\,]$$

Given a set of matched points

$$\{\mathbf{X}_i, \boldsymbol{x}_i\}$$

point in 3D space point in the image

and camera model

$$\underbrace{\boldsymbol{x} = \boldsymbol{f}}_{\text{projection model}}(\mathbf{X}; \underbrace{\boldsymbol{p}}_{\text{parameters}}) = \underbrace{\mathbf{P}}_{\text{Camera matrix}}\mathbf{X}$$

Find the (pose) estimate of

$$\mathbf{P}$$

$$P = \underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{translation}}$$

① 3차원 보정판에서 world 좌표계 좌표 최소 6개를 구함.

② Homography matrix를 구하는 방법도 동일하게 camera matrix(P)를 구함.

③ camera center c를 찾기 위해서 camera matrix SVD 하여 구한 가장 작은 singular vector를 구함 ($\because\ PC = 0$) = translation matrix를 구함

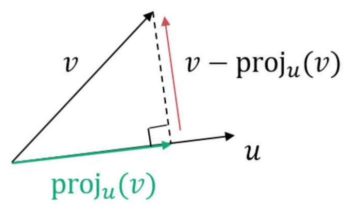
④ M를 QR decomposition하여 Intrinsic matrix K(right upper triangle)와 Rotation Matrix R(orthogonal)을 구함.

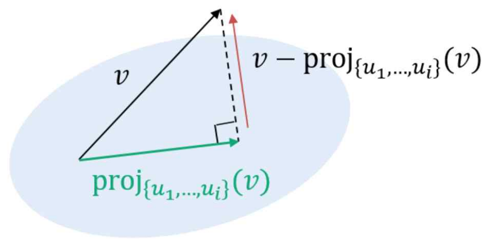
★ 그람슈미트 과정(Gram-Schmidt Process)

: 주어진 벡터들을 이용해서 서로 수직인 벡터들을 만드는 방법

= 주어진 벡터(v_i)들에 대한 직교기저(u_i, orthogonal basis) 또는 정규직교기저(orthonormal basis)를 구하는 과정

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{v}_1, \\ \mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2), \\ \mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), \\ &\vdots \\ \mathbf{u}_k &= \mathbf{v}_k - \sum_{i=1}^{k-1} \mathrm{proj}_{\mathbf{u}_i}(\mathbf{v}_k) \end{aligned}$$



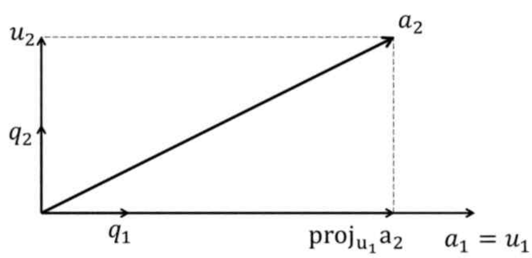


\- $proj_{u_i}(v_j)$: 벡터 v_j를 벡터 u_i에 수직으로 투영한 벡터

\- 이렇게 얻어진 $u_1, u_2, \dots u_k$는 서로 수직(orthogonal)이고 벡터공간 V = $\{v_1, v_2, \dots, v_k\}$에 대한 직교기저가 된다. 이러한 과정을 그람-슈미트 직교화(orthogonalization)라 부른다.

\- 즉, 서로 직교하는 벡터를 구성하기 위해서는 각각이 차지하고 있는 성분 만큼을 빼주어야 한다는 뜻

$$u_2 = a_2 - \underbrace{\mathrm{proj}_{u_1}(a_2)}_{a_2\text{가 갖고 있는 } u_1\text{관련 성분}}$$



★ QR decomposition

: m x n 행렬 A가 서로 linear independent인 열들을 가지고 있을 때 A는 다음과 같이 분해 가능함.

A = QR

- Q = m x n 행렬 Q의 열들은 A의 열공간의 orthonormal basis를 구성함
- R = n x n 행렬 R은 양의 대각원소를 가진 upper triangular invertible matrix임

$$u_1 = v_1$$
$$u_2 = v_2 - \left(\frac{v_2 \cdot u_1}{u_1 \cdot u_1}\right)u_1$$
$$\cdots\cdots$$
$$u_i = v_i - \Sigma_{i=1}^{k-1}\left(\frac{v_i \cdot u_{i-1}}{u_{i-1} \cdot u_{i-1}}\right)u_{i-1}$$

$\Rightarrow$

$$u_1 = v_1$$
$$u_2 = v_2 - (v_2 \cdot U_1)U_1$$
$$\cdots\cdots$$
$$u_i = v_i - \Sigma_{i=0}^{k=i-1}(v_i \cdot U_{i-1})U_{i-1}$$
$$\left(U_i = \frac{u_i}{\|u_i\|} \right)$$

$$v_1 = u_1 = \|u_1\| U_1 = (v_1 \cdot U_1)U_1$$
$$(\because v_1 \cdot U_1 = \|u_1\| U_1 \cdot U_1 = \|u_1\|, \quad \because U_1 \cdot U_1 = 1)$$

$\Rightarrow$

$$v_2 = (v_2 \cdot U_1)U_1 + u_2 = (v_2 \cdot U_1)U_1 + (v_2 \cdot U_2)U_2$$

$$v_3 = (v_3 \cdot U_1)U_1 + (v_3 \cdot U_2)U_2 + u_3$$
$$= (v_3 \cdot U_1)U_1 + (v_3 \cdot U_2)U_2 + (v_3 \cdot U_3)U_3$$

$\Rightarrow$

$$[v_1\ v_2\ v_3] = [U_1\ U_2\ U_3]\begin{bmatrix} v_1 \cdot U_1 & v_2 \cdot U_1 & v_3 \cdot U_1 \\ 0 & v_2 \cdot U_2 & v_3 \cdot U_2 \\ 0 & 0 & v_3 \cdot U_3 \end{bmatrix}$$

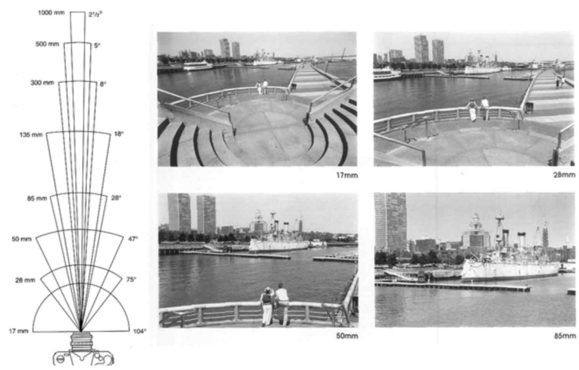
• Ames Room Illusion : anlge과 pallel한 선들의 정보를 잃으므로 착시 효과를 볼 수 있음

• Depth of Field

- len가 focus하는 필름의 부분으로 in focus, 그 외의 부분은 circle of confusion이라는 표현을 씀
- 조리개를 조절하여 aperture 사이즈가 줄어들면, in focus되는 크기가 커짐
- 하지만, light의 양이 줄어들어 exposure 시간을 늘여야 함

• Field of View(FOV)

- focal length와 camera retina의 사이즈에 영향을 받음
- FOV = $\tan^{-1}\left(\frac{d}{2f}\right)$
- 따라서, small f → large FOV : camera close to object
  large f → small FOV : camera far to object



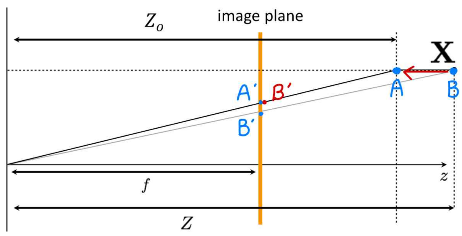
• Orthographic Camera

- Perspective Camera : 사물에 대해 원근감과 공간감을 잘 표현함

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Orthographic Camera : 사물에 대해서 원근감과 공간감 없이 표현함

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & Z_o \end{bmatrix}$$



A와 B가 크기가 같아도 A가 더 가깝기 때문에 더 크게 표현

B를 A의 위치에 이동시키면서 projection 하므로 원근감이 사라짐

Chap 8. Two View Geometry

• Triangulation

Given a set of (noisy) matched points

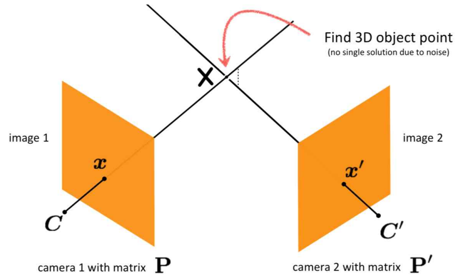
$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

and camera matrices

$$\mathbf{P}, \mathbf{P}'$$

Estimate the 3D point

$$\mathbf{X}$$



$$PX = x$$
$$P^{-1}PX = P^{-1}x$$
$$X = P^{-1}x$$

만약, P의 역행렬이 존재하지 않는다면, pseudo-inverse를 통해서 P의 역행렬을 근사하도록 함

- x와 x′이 교차하지 않아 X를 구할 수 없는 경우가 생기는 이유

① x, x′는 image 좌표이므로 discrete하게 구한 점이므로 error를 포함하고 있음

② camera matrix P와 P′는 estimation하여 구했을 시 error를 포함하고 있음

- 방향이 같이 두 벡터를 Cross Product하면 0이 되는 성질 이용

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \underbrace{\boxed{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}}_{\text{unknown}}$$

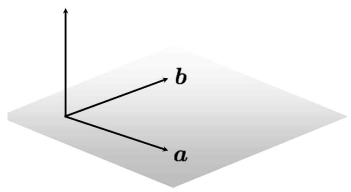
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} \text{---} & \boldsymbol{p}_1^\top & \text{---} \\ \text{---} & \boldsymbol{p}_2^\top & \text{---} \\ \text{---} & \boldsymbol{p}_3^\top & \text{---} \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{X} \\ | \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix}$$

★ Cross Product

takes two vectors and returns a vector perpendicular to both

$$\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$$



$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$$

cross product of two vectors in th
same direction is zero

$$\boldsymbol{a} \times \boldsymbol{a} = 0$$

$$\boldsymbol{c} \cdot \boldsymbol{a} = 0 \qquad \boldsymbol{c} \cdot \boldsymbol{b} = 0$$

Using the fact that the cross product should be zero

$$\mathbf{x} \times \mathbf{P}\boldsymbol{X} = \mathbf{0}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

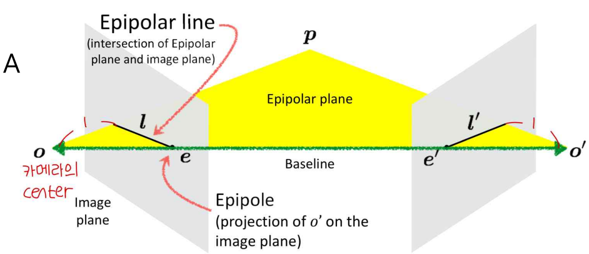
좌, 우 이미지
좌표 대입

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

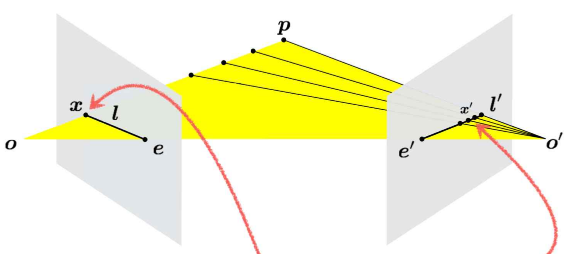
- 마지막 식은 위의 두 식에 dependent하므로 위의 두 식만을 이용함.

• Epipolar Geometry

: 동일한 장면에 대한 영상을 다른 두 지점에서 획득했을 때, 영상 A와 B의 매칭 간의 기하학적 관계를 다루는 것



• Epipolar Constraint



Potential matches for $\boldsymbol{x}$ lie on the epipolar line $\boldsymbol{l}'$

- A의 영상좌표 x로부터 대응되는 B의 영상좌표 x'을 유일하게 결정할 수는 없지만 x'이 지나는 직선인 epipolar line l'은 유일하게 결정할 수 있음

- 한 영상좌표로부터 다른 영상에서의 대응되는 epipolar line을 계산해주는 변환행렬이 Fundamental Matrix, Essential Matrix 임

• Essential Matrix : maps a point to a line (3x3)

- 유도

rigid motion

$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

coplanarity

$$(\boldsymbol{x} - \boldsymbol{t})^\top \boxed{(\boldsymbol{t} \times \boldsymbol{x})} = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})(\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})([\mathbf{t}_\times]\boldsymbol{x}) = 0$$

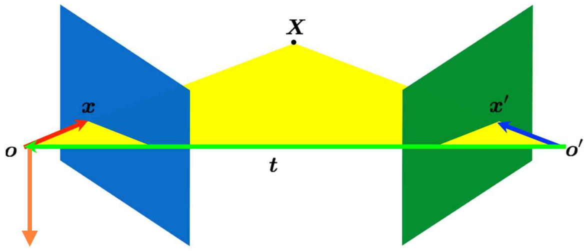
$$\boldsymbol{x}'^\top \boxed{(\mathbf{R}[\mathbf{t}_\times])} \boldsymbol{x} = 0$$

camera 사이의 관계가 E에 encoding

$$\boldsymbol{x}'^\top \mathbf{E} \boldsymbol{x} = 0$$

[t]xR은 t와 R과의 외적을 의미하는게 아니라 먼저 R로 회전을 시킨 후 다음에 t와 외적을 시키는 일련의 변환을 의미함.

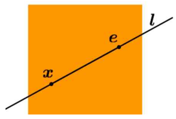
$Ex = [t_\times]Rx = t \times (Rx)$



- 성질

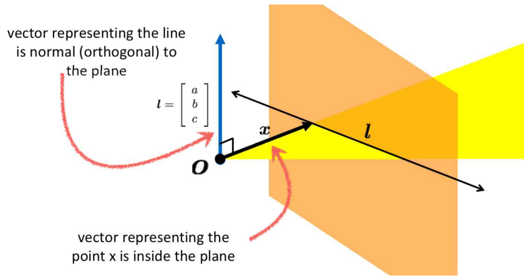
Longuet-Higgins equation	$\boldsymbol{x}'^\top \mathbf{E}\boldsymbol{x} = 0$	
Epipolar lines	$\boldsymbol{x}^\top \boldsymbol{l} = 0$ $\boldsymbol{l}' = \mathbf{E}\boldsymbol{x}$	$\boldsymbol{x}'^\top \boldsymbol{l}' = 0$ $\boldsymbol{l} = \mathbf{E}^T\boldsymbol{x}'$
Epipoles	$\boldsymbol{e}'^\top \mathbf{E} = \mathbf{0}$	$\mathbf{E}\boldsymbol{e} = \mathbf{0}$

$ax + by + c = 0$ in vector form $\boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$



If the point $\boldsymbol{x}$ is on the epipolar line $\boldsymbol{l}$ then

$$\boldsymbol{x}^\top \boldsymbol{l} = 0$$



• Fundamental Matrix

: Essential matrix가 가진 calibrated cameras 가정이 없어진 것으로, intrinsic parameter 요소가 포함된 matrix

$$\hat{\boldsymbol{x}}'^\top \mathbf{E} \hat{\boldsymbol{x}} = 0$$

The Essential matrix operates on image points expressed in **normalized coordinates**
(points have been aligned (normalized) to camera coordinates)

$$\hat{\boldsymbol{x}}' = \mathbf{K}^{-1}\boldsymbol{x}' \qquad \underset{\text{camera point}}{\hat{\boldsymbol{x}}} = \mathbf{K}^{-1} \underset{\text{image point}}{\boldsymbol{x}}$$

Writing out the epipolar constraint in terms of image coordinates

$$\boldsymbol{x}'^\top \mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1} \boldsymbol{x} = 0$$

$$\boldsymbol{x}'^\top (\mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1}) \boldsymbol{x} = 0$$

$$\boldsymbol{x}'^\top \mathbf{F} \boldsymbol{x} = 0$$

- F는 rank = 2인 singular matrix이고 DOF = 7
- E는 rank = 2인 singular matrix이고 DOF = 5

• (Normalized) Eight-Point Algorithm : Fundamental Matirx 구하는 법

① Normalize points : 이미지의 크기가 클 경우, (x,y)의 범위가 커져서 matrix A에서 xx', xy 같은 항들의 중요도가 높아지므로 normalize를 통해서 scale을 맞춰줌

② Construct the M x 9 matrix A

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx_M' & x_My_M' & x_M & y_Mx_M' & y_My_M' & y_M & x_M' & y_M' & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

③ Find the SVD of A

④ Store singular vector of the smallest singular value $F = v_{\hat{i}}$

⑤ Enforce rank 2 constraint on F (F의 마지막 singular value를 0으로 바꿈)

: point(DOF = 0) → line(DOF = 1) mapping이므로 3개였던 constraint를 2개로 relax 해주기 위해서 rank-2 constraint 적용

⑥ Un-normalize F

• 주의할 점

: F와 E의 스케일(scale)까지는 결정할 수 없음. 물체와 카메라가 존재하는 공간 전체를 확대하거나 축소해도 동일한 이미지를 얻기 때문에 이미지만 가지고는 스케일을 알 수 없다는 것을 의미함. 즉, 가까이에 있는 작은 물체를 찍거나 멀리 있는 큰 물체를 찍어도 같은 영상을 얻을 수 있다는 것을 의미함.

★ Singular Matrix

$AX = B \quad \rightarrow \quad X = A^{-1}B$

: 역행렬은 정방행렬에 대해서만 정의되며, 역행렬은 determinant(행렬식) det(A) = 0이면 존재하지 않음.
이때, 역행렬이 존재하지 않는 A를 singular matrix라 함.

★ Determinant

$x' = Ax$

: 행렬 A는 입력좌표 x를 x'으로 변환시켜주는 선형변환(linear transformation)으로 해석 가능
이러한 관점에서 det(A)는 선형변환의 스케일(scale) 성분을 나타내는 값으로 $P' = AP$와 같이 도형 P가 선형변환 A에 의해 P'으로 변환되었을 경우, 다음 수식이 성립한다.

- 면적(P') = $|\det(A)|$ x 면적(P) : 2D의 경우
- 부피(P') = $|\det(A)|$ x 부피(P) : 3D의 경우

이때, det(A)>0이면 도형의 방향(orientation)이 보존되고 det(A)<0이면 도형의 방향이 보존되지 않음. det(A)=0이면, 평면은 직선으로 삼차원 모형은 평면으로 변한다.

Chap 9. Stereo

• Rectified Images

When this relationship holds:

$\boxed{R = I \qquad t = (T, 0, 0)}$

$$\boldsymbol{x}'^T \boldsymbol{E} \boldsymbol{x} = 0, \quad \boldsymbol{E} = [\boldsymbol{t}_\times]\boldsymbol{R}$$

$$\boldsymbol{E} = [\boldsymbol{t}_\times]\boldsymbol{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

- baseline과 각각의 image plane이 평행함
- camera center가 같은 height에 위치함
- focal lengths가 같음
- epipolar line이 image의 x축에 나란함

$$\mathbf{S} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0 \qquad \begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv \end{pmatrix} = 0 \qquad Tv' = Tv$$

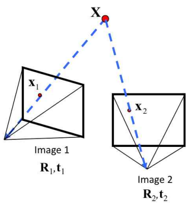
• Rectification : epipole at infinity

① Essential Matrix E 구하기 (8-point algorithm)

② Decompose E into R and T

▪ For a given essential matrix

$$\mathrm{SVD}(\mathbf{E}) = \mathbf{U}\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}\mathbf{V}^T$$

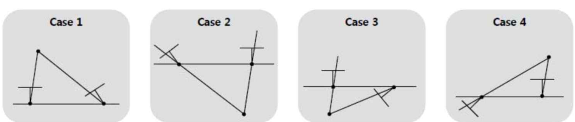


▪ If we assume that $P_1 = [\,I \mid 0\,]$, there are four possible choices for the second camera matrix P_2

$\mathbf{P}_2 = [\mathbf{UWV}^T \mid +\mathbf{u}_3]$
$\mathbf{P}_2 = [\mathbf{UWV}^T \mid -\mathbf{u}_3]$
$\mathbf{P}_2 = [\mathbf{UW}^T\mathbf{V}^T \mid +\mathbf{u}_3]$
$\mathbf{P}_2 = [\mathbf{UW}^T\mathbf{V}^T \mid -\mathbf{u}_3]$

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{u}_3 = \mathbf{U}(\mathbf{0},\mathbf{0},\mathbf{1})^T$: the last column vector of $\mathbf{U}$



$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix}$$

intrinsic parameters   extrinsic parameters

camera matrix에서 intrinsic parameter는 essential metrix에서 고려하지 않아도 됨

4가지 경우의 수 P 중 depth값이 모두 양수인 case1와 같은 상황만 해로 가질 수 있음.

- T = e = u_3
- R = P[:2, :2]

③ R_{rect}를 e(=T)를 통해 구하기

Let $R_{\text{rect}} = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \boldsymbol{r}_3^\top \end{bmatrix}$ Given: epipole **e** (using SVD on E) (translation from **E**)

- singular vector of the smallest singular value $e = v_l = T$
- optical axis = [0, 0, 1]
- $[T_x, T_y, T_z]$ x [0, 0, 1] = $[T_y, -T_x, 0]$

$$\boldsymbol{r}_1 = \boldsymbol{e}_1 = \frac{T}{||T||}$$ epipole coincides with translation vector

$$\boldsymbol{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y & T_x & 0 \end{bmatrix}$$ cross product of **e** and the direction vector of the optical axis

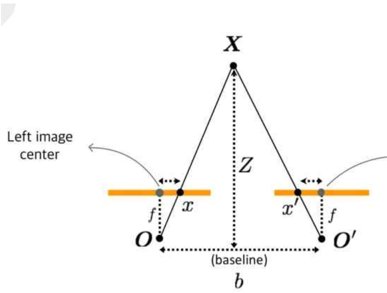
$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$ orthogonal vector

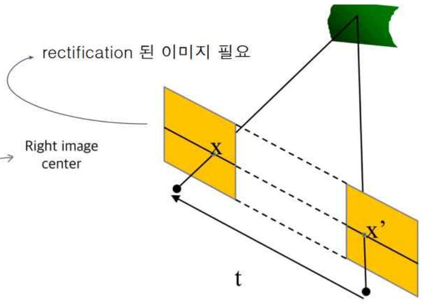
④ $R_1 = R_{rect}$, $R_2 = RR_{rect}$

⑤ 회전시키기 : $[x'_r, y'_r, z'_r] = R_1[x_r, y_r, z_r]$, $[x'_l, y'_l, z'_l] = R_2[x_l, y_l, z_l]$

⑥ Rectified point : $p_r = \frac{f}{z'_r}[x'_r, y'_r, z'_r]$, $p_l = \frac{f}{z'_l}[x'_l, y'_l, z'_l]$

• Disparity and Depth





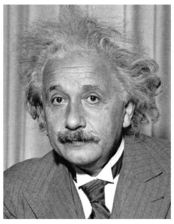
$$disparity\ d = \frac{f \times b}{z}$$

$$depth\ Z = \frac{f \times b}{d}$$

disparity : 좌/우 이미지에서 matching point의 위치 차이

- Rectified된 이미지에서 disparity를 구해야 계산량이 적어짐

• Find the template : 3가지 방법



흰화면 (255) ≥ [eye]

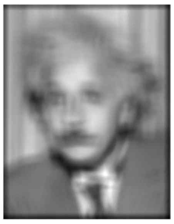
filtering 결과가 눈패치보다 흰바탕에서 값이 더 크게 나옴

filter ≥ 0

output

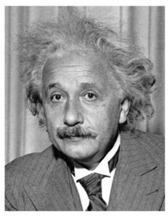
$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image



Solution 1: Filter the image using the template as filter kernel.

What went wrong?
: Increases for higher local intensities.



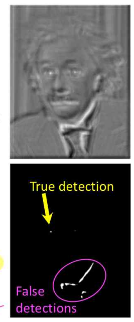
filter, template mean

output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

image

output



thresholding

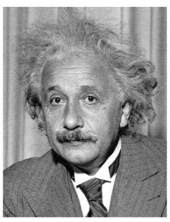
Solution 2: Filter the image using a zero-mean template.

음수 값도 가지므로 빈화면이 [eye]보다 크지 않음

음수인 부분에 0이 곱해지고
야수인 부분에 1이 곱해질 경우

What went wrong?

Not robust to high-contrast areas



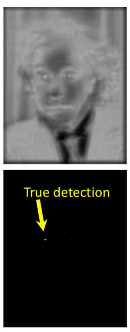
filter

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

image

1-output

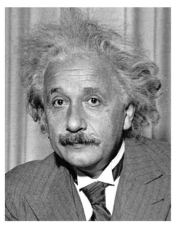


thresholding

Solution 3: Use sum of squared differences (**SSD**).

What could go wrong?

Not robust to local intensity changes



filter, template mean

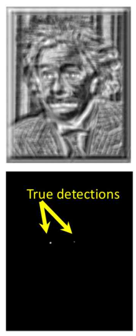
output

$$h[m,n] = \frac{\sum_{k,l}(g[k,l] - \bar{g})(f[m+k, n+l] - \bar{f}_{m,n})}{\sqrt{\left(\sum_{k,l}(g[k,l] - \bar{g})^2 \sum_{k,l}(f[m+k, n+l] - \bar{f}_{m,n})^2\right)}}$$

image

local patch mean

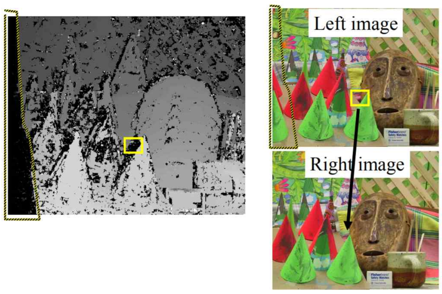
1-output



thresholding

Solution 4: Normalized cross-correlation (**NCC**).

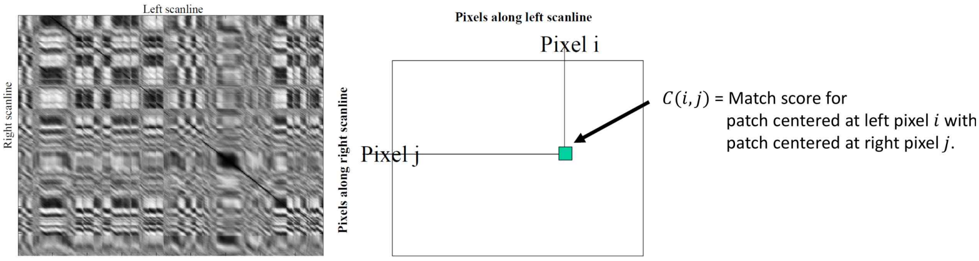
• Correspondence Search
: 한 점에 대해서 epipolar line에서 searching 하는 방법
- 실패하는 상황 : Textureless surfaces, occlusions, repetition, non-lambertian surfaces, specularities



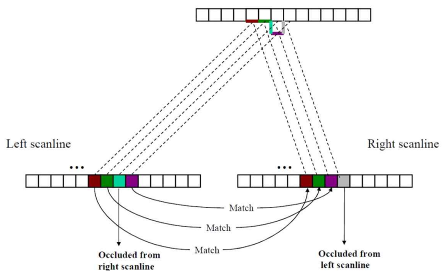
occulsion 예시

• Effect of Window size
- small window : more detail BUT, more noise
- larger window : smoother disparity maps BUT, less detail and fails near boundaries

• 향상된 Stereo Matching 방법
⦿ Disparity Space Image (DSI) : global matching 방법
: epipolar line = scanline의 dissimilarity value를 조사한 후, non-local한 correspondence constraints 영향을 주기 위해서 dynamic programming을 통해 matching point를 찾음



- Correspondence Constraints
① Uniqueness Constraint : matching 점은 하나 또는 0개가 되어야 함.
② Ordering Constraint : occlusion이 일어나지 않은 점에 한해서는 매칭점의 order이 좌/우 이미지에서 동일해야함.

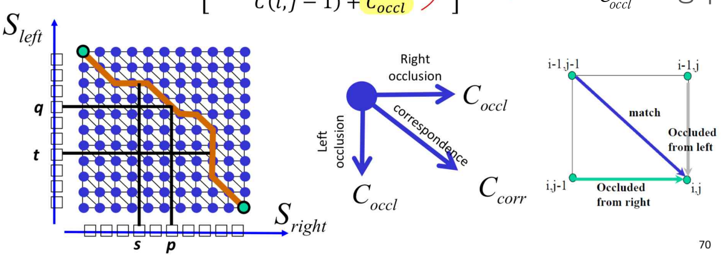


- Dynamic Programming : Shortest paths for scanline stereo

$$C(i,j) = \min \begin{bmatrix} C(i-1,j-1) + \text{dissim}(i,j) \\ C(i-1,j) + C_{occl} \\ C(i,j-1) + C_{occl} \end{bmatrix}$$

사람이 지정

- dissim(i,j) = dissimilarity score
- C_{occl} = 상수



★ Dynamic Programming
: 전체 문제를 작은 문제로 단순화한 다음 점화식으로 만들어 재귀적인 구조를 활용해서 전체 문제를 해결하는 방식

⦿ Block Matching : local matching 방법
: 인접한 pixel간 depth는 smooth하게 변한다는 가정을 두고 disparity 구함
- Energy Function

$$E(d) = E_d(d) + \lambda \boxed{E_s(d)}$$

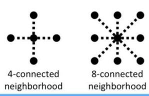
$$E_d(d) = \sum_{(x,y)\in I} C(x,y,d(x,y))$$

data term; SSD distance between windows centered at I(x, y) and J(x+ d(x,y), y)

$$E_s(d) = \sum_{(p,q)\in \mathcal{E}} V(d_p, d_q)$$

smoothness term

$\mathcal{E}$: set of neighboring pixels

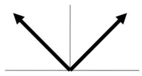


- data term = block matching result
 : want each pixel to find a good match in the other image

- Smoothness term = smoothness function
 : Adjacent pixels should (usally) move about the same amount

$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

"Potts model"