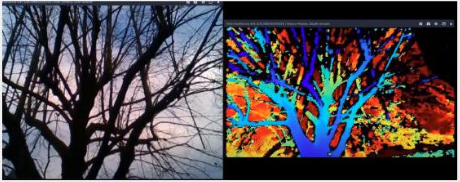
Autonomous Mobile Robotics

Multi-view Geometry

Geesara Kulathunga

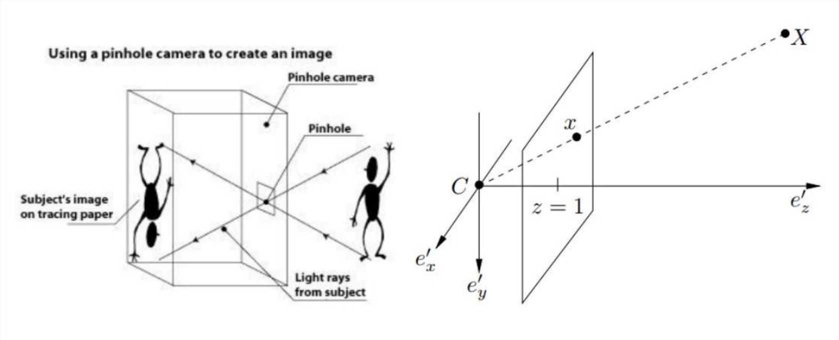


February 13, 2023

Contents

- Monocular Vision
  - Pinhole Camera Model
  - Image Plane, Camera Plane, Projection Matrix
  - Projective transformation
  - Finding Projection Matrix using Direct Linear Transform (DLT)
  - Camera Calibration
- Stereo Vision
  - Simple Stereo, General Stereo
  - Some homogeneous properties
  - Epipolar Geometry
  - Essential matrix, Fundamental matrix
- Depth Estimation

Pinhole camera model



http://www.ctr.maths.lu.se/media/FMA270/2015/alllectures.pdf

- Idea is that light rays enter through **a small hole (the pinhole)** and project an image on **the back of the camera wall**
- If camera coordinate system, defined as $\{e'_x, e'_y, e'_z\}$. The coordinate of **camera center** or **pinhole** of the camera(C) is at $(0, 0, 0)$
- The projection of $\mathbf{X} = (X_w, Y_w, Z_w)$ scene point into the image plane $\mathbf{x}' = (x', y', z')$ while assuming $z' = 1$ has the **normal** e_z lies at the distance 1 from the camera center. e_z can be defined as the viewing direction since the $\mathbf{X} - \mathbf{C}$ is the direction vector of viewing ray

$$\mathbf{C} + s(\mathbf{X} - \mathbf{C}) = \mathbf{sX}, \mathbf{s} \in \mathbb{R}$$

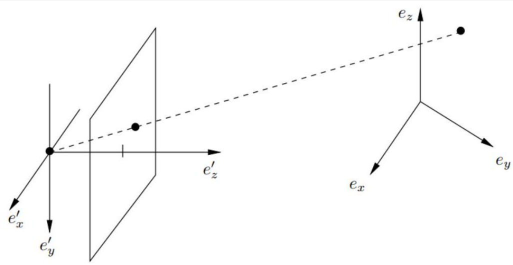
Thus, where will the intersection of this vector be, if $e_z = 1$?

$$\mathbf{x}' = \begin{bmatrix} X_w/Z_w \\ Y_w/Z_w \\ 1 \end{bmatrix}$$

Example 01

Compute the projection of the cube with corners: $(\pm 1, \pm 1, 2)$ and $(\pm 1, \pm 1, 4)$ in image plane?

Global coordinate system and camera coordinate system



In real-world examples, the camera can **undergo a series of rotations and translations**. Hence, it is required to transform the **world coordinate system into a camera coordinate system**.

http://www.ctr.maths.lu.se/media/FMA270/2015/alllectures.pdf

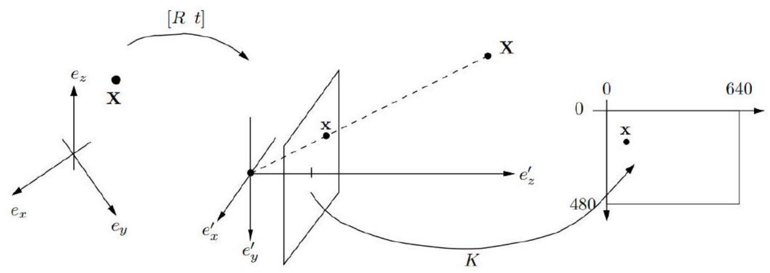
A given point in the global coordinate system can be represented with respect to the camera coordinate system:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = [Rt] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Example 02

Compute the projection of $\mathbf{X} = (0, 0, 1)$ in the cameras coordinate system if R is $\begin{bmatrix} 1 & 0 & -1 \\ 0 & \sqrt{2} & 0 \\ 1 & 0 & 1 \end{bmatrix}$ and t vector equals to $[0, 0, \sqrt{2}]$.
Also, how do you assume for a given point is in the front of the camera or not?

Global coordinate system and camera coordinate system



http://www.ctr.maths.lu.se/media/FMA270/2015/alllectures.pdf

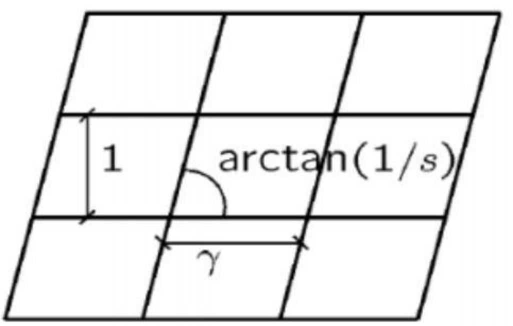
Camera Plane

In the image plane, the **center of the image** is located in, i.e., 0,0,? $(c_x, c_y, 0)$. However, in the camera plane, 0,0 starts from the upper left corner. This transformation is given by the camera matrix, i.e., the inner parameters of the camera. This transformation matrix is denoted as K where it is **invertible**. In general, K is expressed as:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{2}d\text{ translation}} \times \underbrace{\begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{2}d\text{ scaling}} \times \underbrace{\begin{bmatrix} 1 & s/f_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{2d shear}}$$

where f is called focal length, c_x and c_y is denoted the principle point of the camera, γ is the aspect ratio.

The skew parameter (s) corrects non-rectangular pixels and γ is used correct the aspect ratio issue



When the **pixels** are **not square values**, γ will not be equal to one. Otherwise, it will be equal to 1. The final parameter is s which is defined as **skew**. This parameter is used to tilt the pixels

The relationship between a point in the camera and in the world:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \mid t]X = PX$$

where $R \mid t$ is the homogeneous transformation which is composed out of a rotation matrix R, and a translation vector t.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}.$$

All in all, we can define the projective transformation that maps world coordinates points in $\mathbf{R}^3$ to $\mathbf{R}^2$ image coordinate system followed by normalized camera coordinate system.

Projective transformation

The projective transformation that maps world coordinates points in $\mathbf{R}^3$ to $\mathbf{R}^2$ image coordinate system followed by normalized camera coordinate system.

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R \mid t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x X_c / Z_c + c_x \\ f_y Y_c / Z_c + c_y \end{bmatrix}$$

where $x' = X_c / Z_c$ and $y' = Y_c / Z_c$.

Example 03

How we are going to find the λ, K, R and t? Which one of these belongs to intrinsic parameters? As stated before, the task is to find the projection matrix P. Thus, how many unknowns do we have and how many equations do we need to solve in this problem?

Finding P using the direct linear transformation (DLT)

Example 03

Let's say we have N number of points in which correspondence is known between the world and the camera frame.

$$\lambda_i x_i = PX_i, \quad i = 1, ..N$$

In order to find P, can you try to derive an expression for minimum value for N to be satisfied? And prove that N should be equal to or higher than 6.

Finding P using the direct linear transformation (DLT)

Example 03

Let $p_i, i = 1, 2, 3$ be vectors containing the rows of P, that is,

$$P = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix}$$

then, Equ. 9 can be reformulated as follows:

$$\begin{aligned} X_i^\top p_1 - \lambda_i x_i &= 0 \\ X_i^\top p_2 - \lambda_i y_i &= 0 \\ X_i^\top p_3 - \lambda_i &= 0 \end{aligned}$$

Finding P using the direct linear transformation (DLT)

Example 03

Can you convert the previous formulation into a matrix form?

$$\underbrace{\begin{bmatrix} X_1^T & 0 & 0 & -x_1 & 0 & 0 & \cdots \\ 0 & X_1^T & 0 & -y_1 & 0 & 0 & \cdots \\ 0 & 0 & X_1^T & -1 & 0 & 0 & \cdots \\ X_2^T & 0 & 0 & 0 & -x_2 & 0 & \cdots \\ 0 & X_2^T & 0 & 0 & -y_2 & 0 & \cdots \\ 0 & 0 & X_2^T & 0 & -1 & 0 & \cdots \\ X_3^T & 0 & 0 & 0 & 0 & -x_3 & \cdots \\ 0 & X_3^T & 0 & 0 & 0 & -y_3 & \cdots \\ 0 & 0 & X_3^T & 0 & 0 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{bmatrix}}_{v} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

Finding P using the direct linear transformation (DLT)

Example 03

In order to find vector v, we have to find the **null space vector of M**. Basically, need to solve system $Mv = 0$. Can we actually solve this? I would say no! what are you up to?

Decompose P into K, R, and t

- The projection matrix P = K (R, t) with size of 3×4

$$P = (P_{13}, p4)$$

- The projection matrix P = K (R, t) with size of 3×4

$$P = (P_{13}, p4)$$

- Hence, $P_{13} = KR$

Decompose P into K, R, and t

- The projection matrix P = K (R, t) with size of 3×4

$$P = (P_{13}, p4)$$

- Hence, $P_{13} = KR$
- Matrix P_{13} is the product of an upper triangular matrix and a rotation matrix

Decompose P into K, R, and t

- The projection matrix P = K (R, t) with size of 3×4

$$P = (P_{13}, p4)$$

- Hence, $P_{13} = KR$
- Matrix P_{13} is the product of an upper triangular matrix and a rotation matrix
- **QR decomposition** is the ideal choice to decompose P_{13} into KR

- K is a upper tringular matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

How QR factorization works for our case

- K is a upper tringular matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

- R is a rotation matrix $\Rightarrow R^\top R = I$

How QR factorization works for our case

- K is a upper tringular matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

- R is a rotation matrix $\Rightarrow R^\top R = I$
- Extract last row of R from P_{13}?

Extract **f** using last row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ fr_{31} & fr_{32} & fr_{33} \end{pmatrix}$$

Extract **f** using last row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ fr_{31} & fr_{32} & fr_{33} \end{pmatrix}$$
- Let the last row of P_{13} be $\mathbf{p}_3^\top = f(r_{31}, r_{32}, r_{33}) = f\mathbf{r}_3^\top$

Extract **f** using last row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ fr_{31} & fr_{32} & fr_{33} \end{pmatrix}$$
- Let the last row of P_{13} be $\mathbf{p}_3^\top = f(r_{31}, r_{32}, r_{33}) = f\mathbf{r}_3^\top$
- Since $R^\top R = I$, $\mathbf{r}_3^\top \mathbf{r} = 1$

Extract **f** using last row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ fr_{31} & fr_{32} & fr_{33} \end{pmatrix}$$
- Let the last row of P_{13} be $\mathbf{p}_3^\top = f(r_{31}, r_{32}, r_{33}) = f\mathbf{r}_3^\top$
- Since $R^\top R = I$, $\mathbf{r}_3^\top \mathbf{r} = 1$
- $\mathbf{p}_3^\top \mathbf{p}_3 = f^2 \Rightarrow f = \|\mathbf{p}_3\|$

Extract **f** using last row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ fr_{31} & fr_{32} & fr_{33} \end{pmatrix}$$
- Let the last row of P_{13} be $\mathbf{p}_3^\top = f(r_{31}, r_{32}, r_{33}) = f\mathbf{r}_3^\top$
- Since $R^\top R = I$, $\mathbf{r}_3^\top \mathbf{r} = 1$
- $\mathbf{p}_3^\top \mathbf{p}_3 = f^2 \Rightarrow f = \|\mathbf{p}_3\|$
- Hence, how can we recover last row of R?

Extract **f** using last row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ fr_{31} & fr_{32} & fr_{33} \end{pmatrix}$$
- Let the last row of P_{13} be $\mathbf{p}_3^\top = f(r_{31}, r_{32}, r_{33}) = f\mathbf{r}_3^\top$
- Since $R^\top R = I$, $\mathbf{r}_3^\top \mathbf{r} = 1$
- $\mathbf{p}_3^\top \mathbf{p}_3 = f^2 \Rightarrow f = \|\mathbf{p}_3\|$
- Hence, how can we recover last row of R?
- $\mathbf{r}_3 = \mathbf{p} / \|\mathbf{p}_3\|$

Extract middle row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$
$$= d \begin{pmatrix} * & * & * \\ r_{21} & r_{22} & r_{23} \\ * & * & * \end{pmatrix} + e \begin{pmatrix} * & * & * \\ r_{31} & r_{32} & r_{33} \\ * & * & * \end{pmatrix}$$

Extract middle row of R from P_{13}

- $$\begin{aligned} P_{13} = KR &= \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \\ &= d \begin{pmatrix} * & * & * \\ r_{21} & r_{22} & r_{23} \\ * & * & * \end{pmatrix} + e \begin{pmatrix} * & * & * \\ r_{31} & r_{32} & r_{33} \\ * & * & * \end{pmatrix} \end{aligned}$$
- $\mathbf{r}_3 = \mathbf{p} / \|\mathbf{p}_3\|$ is known and have $\mathbf{p}_2 = d\mathbf{r}_2 + e\mathbf{r}_3$

Extract middle row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$
  $$= d \begin{pmatrix} * & * & * \\ r_{21} & r_{22} & r_{23} \\ * & * & * \end{pmatrix} + e \begin{pmatrix} * & * & * \\ r_{31} & r_{32} & r_{33} \\ * & * & * \end{pmatrix}$$
- $\mathbf{r}_3 = \mathbf{p}/\|\mathbf{p}_3\|$ is known and have $\mathbf{p}_2 = d\mathbf{r}_2 + e\mathbf{r}_3$
- $\mathbf{r}_3^\top \mathbf{p}_2 = d\mathbf{r}_3^\top \mathbf{r}_2 + e\mathbf{r}_3^\top \mathbf{r}_3 = e$ using orthogonality constraints

Extract middle row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$
  $$= d \begin{pmatrix} * & * & * \\ r_{21} & r_{22} & r_{23} \\ * & * & * \end{pmatrix} + e \begin{pmatrix} * & * & * \\ r_{31} & r_{32} & r_{33} \\ * & * & * \end{pmatrix}$$
- $\mathbf{r}_3 = \mathbf{p}/\left\|\mathbf{p}_3\right\|$ is known and have $\mathbf{p}_2 = d\mathbf{r}_2 + e\mathbf{r}_3$
- $\mathbf{r}_3^\top \mathbf{p}_2 = d\mathbf{r}_3^\top \mathbf{r}_2 + e\mathbf{r}_3^\top \mathbf{r}_3 = e$ using orthogonality constraints
- $\mathbf{r}_2 = \frac{\mathbf{p}_2 - e\mathbf{r}_3}{d}$, since $\mathbf{r}_2^\top \mathbf{r}_2 = 1, \Rightarrow d = \left\|\mathbf{p}_2 - e\mathbf{r}_3\right\|$

How QR factorization works for our case

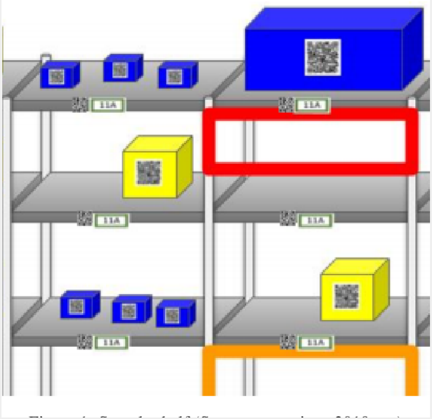
Extract middle row of R from P_{13}

- $$P_{13} = KR = \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$
  $$= d \begin{pmatrix} * & * & * \\ r_{21} & r_{22} & r_{23} \\ * & * & * \end{pmatrix} + e \begin{pmatrix} * & * & * \\ r_{31} & r_{32} & r_{33} \\ * & * & * \end{pmatrix}$$
- $\mathbf{r}_3 = \mathbf{p} / \|\mathbf{p}_3\|$ is known and have $\mathbf{p}_2 = d\mathbf{r}_2 + e\mathbf{r}_3$
- $\mathbf{r}_3^\top \mathbf{p}_2 = d\mathbf{r}_3^\top \mathbf{r}_2 + e\mathbf{r}_3^\top \mathbf{r}_3 = e$ using orthogonality constraints
- $\mathbf{r}_2 = \frac{\mathbf{p}_2 - e\mathbf{r}_3}{d}$, since $\mathbf{r}_2^\top \mathbf{r}_2 = 1, \Rightarrow d = \|\mathbf{p}_2 - e\mathbf{r}_3\|$
- That is, $\mathbf{r}_2 = \frac{\mathbf{p}_2 - e\mathbf{r}_3}{\|\mathbf{p}_2 - e\mathbf{r}_3\|}$

Camera calibration is all about finding project matrix ($P = K[Rt]$).
More information can be found here:
`https://www.mathworks.com/help/vision/camera-calibration.html` or
`http://wiki.ros.org/camera_calibration`.

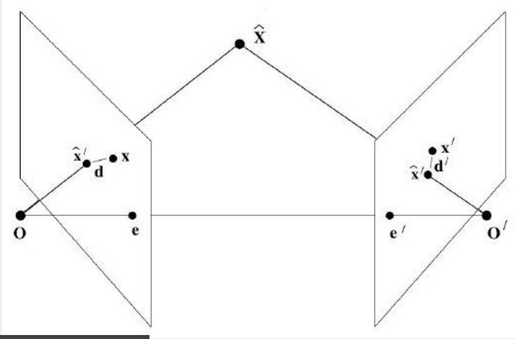
Depth estimation using monocular camera

Can you use monocular camera for depth estimation?

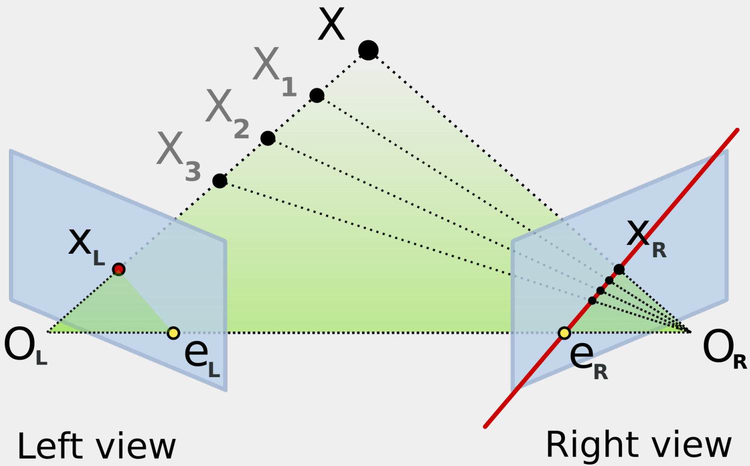


https://www.semanticscholar.org/paper/Warehouse-Management-Using-Real-Time-QR-Code-and-Saha-Udayagiri/4c6c478b7ba8c46dca35dcba5d69648610c2742b

If the **camera matrices are known** (the triangulation problem) Direct Linear Transformation (**DLT**) to find the projection matrix (**P**). On the contrary, if the scene points and camera matrices are not known problems get complicated. The main intuition is to find **some similarities between considered two images** where part of those are overlapping each other. The technique used to solve this problem is called epipolar geometry.

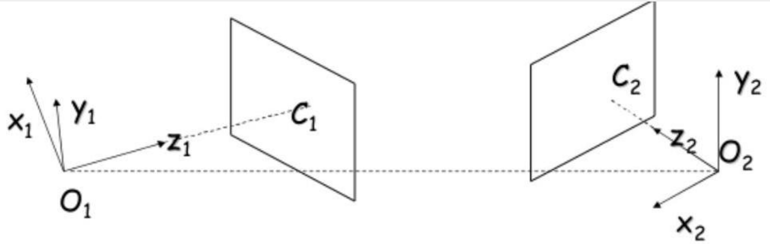


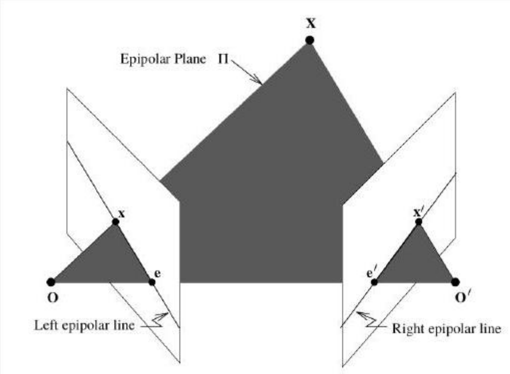
If both projection matrices, i.e., P_1 and P_2, are known, how can we estimate the $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ for a known $\hat{\mathbf{X}}$?



https://en.wikipedia.org/wiki/Epipolar_geometry

In general, there are two types of problems that belong to general stereo: matrix K is known, need to find $[\mathrm{R}t]$ matrix (Essential Matrix) or matrix K also unknown or has different focal lengths (Fundamental Matrix).





Terms, **e** and **e**′, are considered as **epipoles**, **epipolar plane** is defined by points **o**′, **O** and **X**. Besides, assume f and f' are the focal lengths of left and right cameras, respectively

- Point x on a line

$$\mathbf{l}^T\mathbf{x} = \mathbf{x}^T\mathbf{l} = 0, l_1x + l_2y + l_3 = 0$$

- Two points define a line

$$l = X_1 \times X_2$$

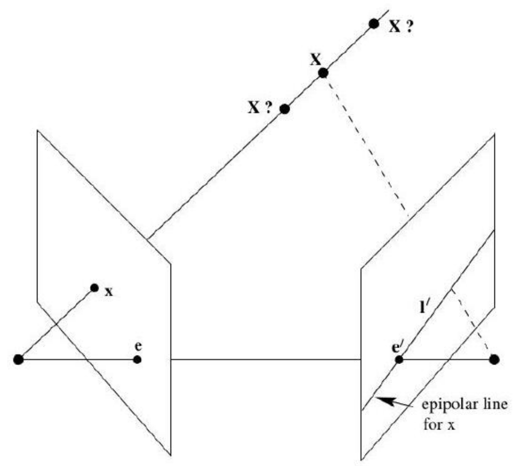
- Intersection of two lines defines a point

$$x = l_1 \times l_2$$

where cross product between two vectors can be written as a matrix multiplication

$$\mathbf{v} \times \mathbf{u} = [\mathbf{v}]_\times \mathbf{u}, \mathbf{v}_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

How can we see the location we see from left camera from the right camera



Since we have two cameras, two projection matrices with respect to left and right cameras have to be identified:

$$\mathbf{x} = \lambda_1 P_1 \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

$$\mathbf{x}' = \lambda_2 P_2 \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

where $P_1 = K_1(I \mid 0)$ and $P_2 = K_2(R \mid t)$, and baseline between the two cameras is denoted by t. Let's start assuming K_1 and K_2 are known. Then if

$$\hat{\mathbf{x}}' = K_2^{-1}\mathbf{x}' = \lambda_2(R \mid t) \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

$$\hat{\mathbf{x}} = K_1^{-1}\mathbf{x} = \lambda_1(I \mid 0) \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

Now let's project **X** on the left and right images

$$\hat{\mathbf{x}} = \lambda_1 (I \mid \mathbf{0}) \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \Rightarrow \mathbf{X} = \lambda_1^{-1} \hat{\mathbf{x}}$$

$$\lambda_2 (R \mid t) \begin{pmatrix} \lambda_1^{-1} \hat{\mathbf{x}} \\ 1 \end{pmatrix} = \lambda_2 \lambda_1^{-1} R \hat{\mathbf{x}} + \lambda_2 t = \lambda_2 \left(\lambda_1^{-1} R \hat{\mathbf{x}} + t \right)$$

And this will be the epipolar line with respect to right camera in our setup. Let's take corresponding points when $\lambda_1 = 1$ and $\lambda_1 = \pm\infty$

$$(R\hat{\mathbf{x}} + t), t$$

Thus, we can define the right epipolar line:

$$\begin{aligned} \mathbf{l}' = t \times (R\hat{\mathbf{x}} + t) = t \times R\hat{\mathbf{x}} + t \times t = t \times R\hat{\mathbf{x}} \\ = [t]_\times R\hat{\mathbf{x}} = E\hat{\mathbf{x}} \end{aligned}$$

This matrix E is called the **Essential matrix**, which map point in the left image to a line in the right image. Thus, we can define the epipolar constraint that $\hat{\mathbf{x}}'$ lies on $\mathbf{l}'$ can be written as

$$\hat{\mathbf{x}}'^T \mathbf{l}' = \hat{\mathbf{x}}'^T E \hat{\mathbf{x}} = 0$$

Some properties of Essential matrix

- The **epipolar line** corresponding to $\hat{\mathbf{x}}'$ is given by

$$\mathbf{l} = E^T \hat{\mathbf{x}}'$$

- The epipole $\mathbf{e}'$ by definition has

$$0 = e'^T \mathbf{l}' = e'^T E \hat{\mathbf{x}}$$

where $\mathbf{e}'^T E = 0$ for all $\mathbf{x}$. Thus, $\mathbf{e}'$ is the **left null space** of E. Similarly, $\text{Ee} = 0$ that is the **right null-space** of E.

- Epipolar constraints (Essential matrix)

$$\begin{pmatrix} \hat{x}' & \hat{y}' & 1 \end{pmatrix} \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = 0$$

- Epipolar constraints (Essential matrix)

$$\begin{pmatrix} \hat{x}' & \hat{y}' & 1 \end{pmatrix} \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = 0$$

- Essential matrix has **rank 2 and is singular**

Essential matrix estimation 8-point algorithm

- Epipolar constraints (Essential matrix)

$$\begin{pmatrix} \hat{x}' & \hat{y}' & 1 \end{pmatrix} \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = 0$$

- Essential matrix has **rank 2 and is singular**
- Essential matrix has **5 degree of freedom** (+3 for rotation, +3 for translation, -1 due to homogeneous coordinates)

- Epipolar constraints (Essential matrix)

$$\begin{pmatrix}\hat{x}' & \hat{y}' & 1\end{pmatrix}\begin{pmatrix}e_{11} & e_{12} & e_{13}\\ e_{21} & e_{22} & e_{23}\\ e_{31} & e_{32} & e_{33}\end{pmatrix}\begin{pmatrix}\hat{x}\\ \hat{y}\\ 1\end{pmatrix} = 0$$

- Essential matrix has **rank 2 and is singular**
- Essential matrix has **5 degree of freedom** (+3 for rotation, +3 for translation, -1 due to homogeneous coordinates)
- Essential matrix can be found as $\Phi^\top \mathbf{e} = 0$

$$\Phi = \begin{bmatrix}\hat{x}'\hat{x} & \hat{x}'\hat{y} & \hat{y}'\hat{x} & \hat{y}'\hat{y}' & \hat{x}' & \hat{y}' & \hat{x} & \hat{y} & 1\end{bmatrix}^\top,$$

$$\mathbf{e} = \begin{bmatrix}e_{11} & e_{12} & e_{13} & e_{21} & e_{22} & e_{23} & e_{31} & e_{32} & e_{33}\end{bmatrix}^\top$$

If we do not have camera parameters as well. In that sense, along with the Essential matrix **can we calculate the correspondence?** For that we have to go from image plane to camera plane and estimate the correspondence, namely **Fundamental Matrix F**, between camera planes.
Let's plug back-in the camera coordinates since we do not know the camera **intrinsic parameters**.

$$\hat{\mathbf{x}}'^{\top} E \hat{\mathbf{x}} = \mathbf{x}'^{\top} K_2^{-\top} E K_1^{-1} \mathbf{x} = \mathbf{x}'^{\top} K_2^{-T} [t]_{\times} R K_1^{-1} \mathbf{x} = \mathbf{x}'^{\top} F \mathbf{x}$$

where F is the fundamental matrix.

Some properties of Fundamental matrix

- F is a 3×3 rank 2 homogeneous matrix

Fundamental matrix

Some properties of Fundamental matrix

- F is a 3×3 rank 2 homogeneous matrix
- $F^T \mathbf{e}' = 0$

Some properties of Fundamental matrix

- F is a 3×3 rank 2 homogeneous matrix
- $F^T \mathbf{e}' = 0$
- 7 degree of freedom (-1 for scaling and -1 for $\det(F) = 0$ that is not a full rank matrix)

Some properties of Fundamental matrix

- F is a 3×3 rank 2 homogeneous matrix
- $F^T \mathbf{e}' = 0$
- 7 degree of freedom (-1 for scaling and -1 for $\det(F) = 0$ that is not a full rank matrix)
- Epipolar lines

$$\mathbf{l}' = F\mathbf{x}, \mathbf{l} = F^T\mathbf{x}'$$

Some properties of Fundamental matrix

- F is a 3×3 rank 2 homogeneous matrix
- $F^T \mathbf{e}' = 0$
- 7 degree of freedom (-1 for scaling and -1 for $\det(F) = 0$ that is not a full rank matrix)
- Epipolar lines
$$\mathbf{l}' = F\mathbf{x}, \mathbf{l} = F^T\mathbf{x}'$$
- Epipoles:
$$F\mathbf{e} = 0, F^T\mathbf{e}' = 0$$

Some properties of Fundamental matrix

- F is a 3×3 rank 2 homogeneous matrix
- $F^T\mathbf{e}' = 0$
- 7 degree of freedom (-1 for scaling and -1 for $\det(F) = 0$ that is not a full rank matrix)
- Epipolar lines

$$\mathbf{l}' = F\mathbf{x}, \mathbf{l} = F^T\mathbf{x}'$$

- Epipoles:

$$F\mathbf{e} = 0, F^T\mathbf{e}' = 0$$

- F matrix consists of **16 number of components**, i.e., 10 for $k_1 k_2$ 3 for R, and 3 for t, hence those are never be decomposed into its components

There are various techniques can be applied to calculate F. **8-point algorithm** is the one of primitive techniques is used to find the matrix F. We have a epipolar constraint $\left(\mathbf{x}'^T F\mathbf{x} = 0\right)$ for each corresponding points in right and left images. Let $\mathbf{x}' \sim \left(x_i', y_i', z_i'\right)$ and $\mathbf{x} \sim (x_i, y_i, z_i)$.

$$\mathbf{x}'^T F\mathbf{x} = 0$$

Thus, if we have n number of correspondences in which each correspondence contributes with one linear constraint of F.

$$\begin{pmatrix} x_1'x_1 & x_1'y_1 \ldots z_1'z_1 \\ x_2'x_2 & x_2'y_2 \ldots z_2'z_2 \\ . & . \\ . & . \\ . & . \\ x_n'x_n & x_n'y_n \ldots z_n'z_n \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ . \\ . \\ . \\ F_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ 0 \end{pmatrix}$$

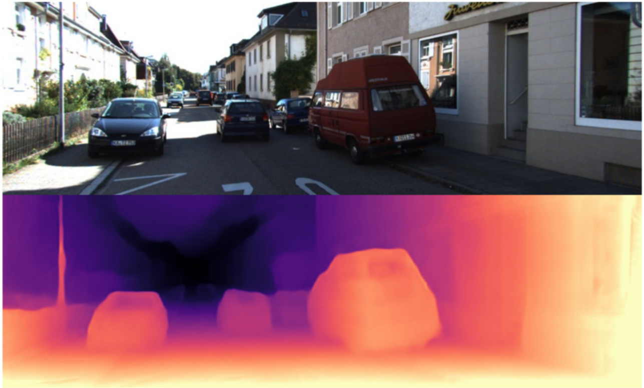
These kinds of linear homogeneous systems can be solved with SVD (**Singular Value Decomposition**). The matrix F has 9 entries. But image correspondence is taken in the image plane, $z_i' = 1$ and $z_i = 1$. Thus, this system has the 8 degrees of freedom. One of the properties of F is $\det(F) = 0$. However, this constraint is not actually true due to the noise of the system. Therefore, it is required to minimize this $\left(\min_{\det(F)=0} |\hat{F} - F|\right)$ in order to find matrix F. Solution to $\hat{F}$ is given by SVD of it. $USV^t = \hat{F}$ where $S = \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)$. Then F can be found by setting the smallest singular value $\sigma_3 = 0$, that is

$$F = U \operatorname{diag}\left(\sigma_1, \sigma_2,\ 0\right) V^\top$$

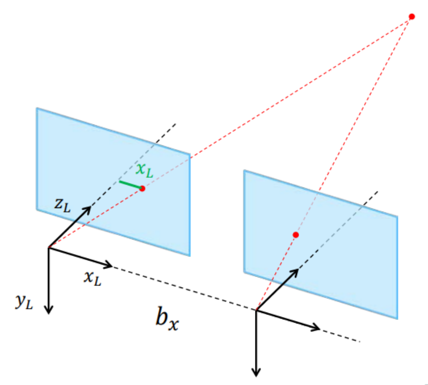
Also, $F\mathbf{e} = 0$. Hence, e is the last column of V. Similarly, $F^T\mathbf{e}' = 0$ which implies $\mathbf{e}'$ is the last column of U.

https://www.youtube.com/watch?v=EokL7E6o1AE

Depth Estimation

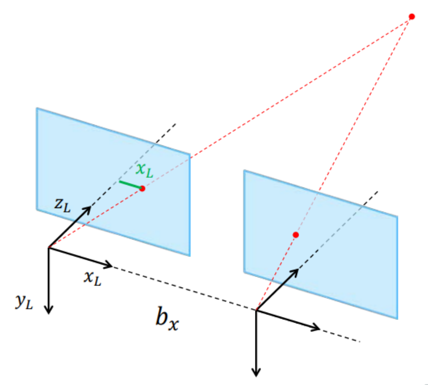


https://github.com/nianticlabs/monodepth2



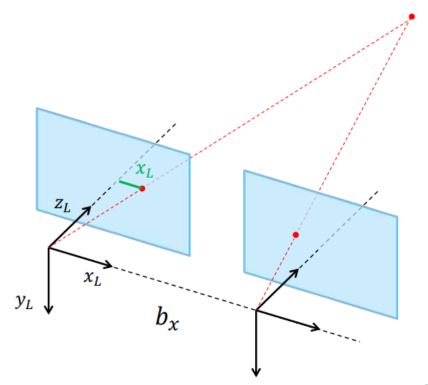
- Consider both left and right camera have the same focal length f

https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger/lecture_6_2_stereo_imaging.pdf



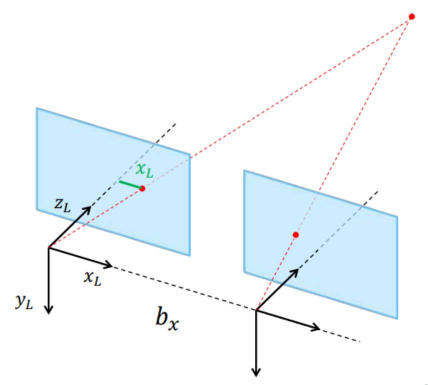
- Consider both left and right camera have the same focal length f
- If the desired position in the world (X,Y,Z) then $\frac{Z}{f} = \frac{X}{x_l}$, $\frac{Z}{f} = \frac{X-b_x}{x_r}$, $\frac{Z}{f} = \frac{Y}{y_l}$, and $\frac{Z}{f} = \frac{Y}{y_r}$

https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger/lecture_6_2_stereo_imaging.pdf



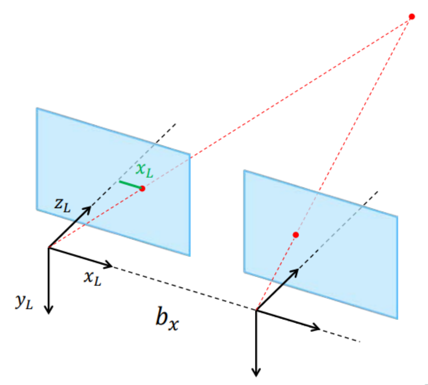
- Consider both left and right camera have the same focal length f
- If the desired position in the world (X,Y,Z) then $\frac{Z}{f} = \frac{X}{x_l}$, $\frac{Z}{f} = \frac{X-b_x}{x_r}$, $\frac{Z}{f} = \frac{Y}{y_l}$, and $\frac{Z}{f} = \frac{Y}{y_r}$
- Depth is estimated as $Z = \frac{f \times b_x}{x_l - x_r}$

https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger/lecture_6_2_stereo_imaging.pdf



- Consider both left and right camera have the same focal length f
- If the desired position in the world (X,Y,Z) then $\frac{Z}{f} = \frac{X}{x_l}$, $\frac{Z}{f} = \frac{X - b_x}{x_r}$, $\frac{Z}{f} = \frac{Y}{y_l}$, and $\frac{Z}{f} = \frac{Y}{y_r}$
- Depth is estimated as $Z = \frac{f \times b_x}{x_l - x_r}$
- $x_l - x_r$ is called disparity

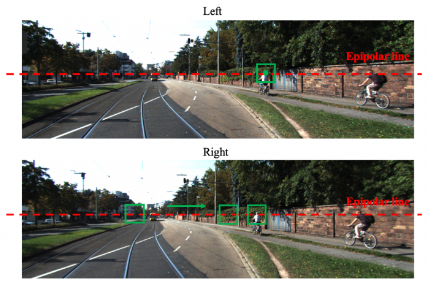
https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger/lecture_6_2_stereo_imaging.pdf

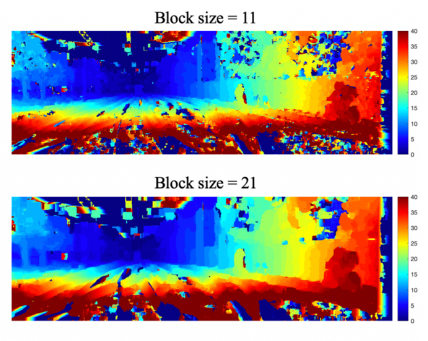


- Consider both left and right camera have the same focal length f
- If the desired position in the world (X,Y,Z) then $\frac{Z}{f} = \frac{X}{x_l}$, $\frac{Z}{f} = \frac{X-b_x}{x_r}$, $\frac{Z}{f} = \frac{Y}{y_l}$, and $\frac{Z}{f} = \frac{Y}{y_r}$
- Depth is estimated as $Z = \frac{f \times b_x}{x_l - x_r}$
- $x_l - x_r$ is called disparity
- Depth is **inversely proportional** to disparity

https://www.uio.no/studier/emner/matnat/its/nedlagte-emner/UNIK4690/v16/forelesninger/lecture_6_2_stereo_imaging.pdf

Similarity Measurements





https://inst.eecs.berkeley.edu/~cs194-26/sp20/upload/files/projFinalProposed/cs194-26-adw/fuyi_yang_finalproj/

Example 3

Let's say you have two feature set x and y as follow, $x = [2000, 2, 3456, 1, 0]$ and $y = [2000, 3, 3400, 3, 0]$. If it is required to get similarity between x and y how you are going to calculate it?

Example 3

Let's say you have two feature set x and y as follow, $x = [2000, 2, 3456, 1, 0]$ and $y = [2000, 3, 3400, 3, 0]$. If it is required to get similarity between x and y how you are going to calculate it?

$$distance = \frac{\Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \bar{x})^2(y_i - \bar{y})^2}}$$

Similarity Measurements

https://openaccess.thecvf.com/content_cvpr_2017/papers/Ufer_Deep_Semantic_Feature_CVPR_2017_paper.pdf

Let two images be $J[x, y]$ and $I[x, y]$ with $(x, y) \in N^{N \times M}$

- Template matching: **linear** and is **not invariant to rotation**
  - Sum Square Difference

$$S_{sq} = \sum_{(n,m) \in N^{M \times N}} \big(J[n, m] - I[n, m]\big)^2$$

or in the normalized form

$$\frac{S_{sq}}{\sqrt{\sum J[n, m]^2 \times \sum I[n, m]^2}}$$

Let two images be $J[x, y]$ and $I[x, y]$ with $(x, y) \in N^{N \times M}$

- Template matching: **linear** and is **not invariant to rotation**
  - Sum Square Difference

$$S_{sq} = \sum_{(n,m) \in N^{M \times N}} \big(J[n, m] - I[n, m]\big)^2$$

or in the normalized form

$$\frac{S_{sq}}{\sqrt{\sum J[n, m]^2 \times \sum I[n, m]^2}}$$

  - Cross-Correlation

$$C_{crr} = \sum_{(n,m) \in N^{M \times N}} \big(J[n, m] \times I[n, m]\big)^2$$

or in the normalized form

$$\frac{C_{crr}}{\sqrt{\sum J[n, m]^2 \times \sum I[n, m]^2}}$$

- Feature detectors/descriptors: various ways to **detect points** that are considered as **features**
  - SIFT and SURF
    Scale invariant, SURF is a open version of SIFT

- Feature detectors/descriptors: various ways to **detect points** that are considered as **features**
  - SIFT and SURF
    Scale invariant, SURF is a open version of SIFT
  - BRIEF, BRISK and FAST
    Binary descriptors and comparatively fast

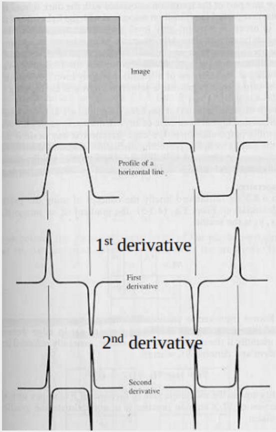
- Feature detectors/descriptors: various ways to **detect points** that are considered as **features**
  - SIFT and SURF
    Scale invariant, SURF is a open version of SIFT
  - BRIEF, BRISK and FAST
    Binary descriptors and comparatively fast
  - Histogram of Oriented Gradients (HoG)
    Rotation invariant

- **Significant local changes** of intensity in an image is called as **edges**

- **Significant local changes** of intensity in an image is called as **edges**
- **Geometric changes** such as **object** or **surface boundaries** or **non-geometric changes** such as **specularity**, shadows and inter-reflection are the main causes for intensity changes

- **Significant local changes** of intensity in an image is called as **edges**
- **Geometric changes** such as **object** or **surface boundaries** or **non-geometric changes** such as **specularity**, shadows and inter-reflection are the main causes for intensity changes
- There are two types of techniques are being used for edge detection either using **derivative** and/or the **gradient**.

Edge Detection Based on Derivative

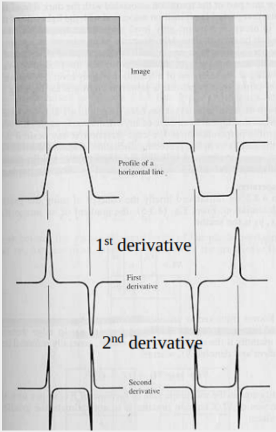


- Computing the first derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The first derivative is used to detect local maxima or minima

Edge Detection Based on Derivative



- Computing the first derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The first derivative is used to detect local maxima or minima

- Computing the second derivative

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h}$$
$$\simeq f'(x+h) - f'(x)$$

The second derivative is used to detect zero-crossing points

Edge Detection Based on Gradient

The gradient vector can be defined as,

$$\bigtriangledown f = [\frac{\partial f}{\partial x} \frac{\partial f}{\partial y}]^T$$

where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial x}$ can be approximated for finite difference as

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y), \quad \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y)$$

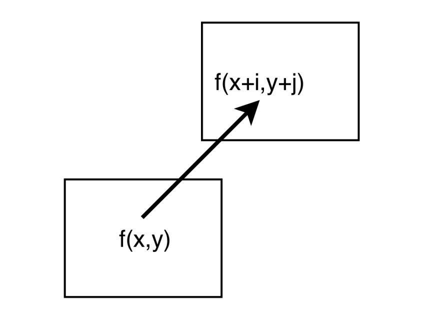
The **magnitude** can be calculated as,

$$|\bigtriangledown f| = \sqrt{((\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2)} = \sqrt{M_x^2 + M_y^2} \tag{1}$$

the **direction** of the vector can be derived as,

$$dir(\bigtriangledown f) = \tan^{-1}(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}) \tag{2}$$

The intensity changes over an image for a given direction is defined by the sum of squared root difference (SSD).



$$Dis(i,j) = \Sigma_{x,y}(f(x+i,y+j) - f(x,y))^2 \quad (3)$$

- Distance $Dis(i,j) = \Sigma_{x,y}(f(x+i, y+j) - f(x,y))^2$

- Distance $Dis(i,j) = \Sigma_{x,y}(f(x+i,y+j) - f(x,y))^2$
- With the assumption i and j are small, by using the Taylor theorem:

$$f(x+i,y+j) \simeq f(x,y) + \frac{\partial f}{\partial x} \cdot i + \frac{\partial f}{\partial y} \cdot j = f_x \cdot i + f_y \cdot j + f(x,y)$$

- Distance $Dis(i,j) = \Sigma_{x,y}(f(x+i,y+j) - f(x,y))^2$
- With the assumption i and j are small, by using the Taylor theorem:

$$f(x+i,y+j) \simeq f(x,y) + \frac{\partial f}{\partial x} \cdot i + \frac{\partial f}{\partial y} \cdot j = f_x \cdot i + f_y \cdot j + f(x,y)$$

- Thus, $(f(x+i,y+j) - f(x,y))^2$ can be quantified as,

$$(f(x+i,y+j) - f(x,y))^2 = \begin{bmatrix} i & j \end{bmatrix} \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \tag{4}$$

- Distance $Dis(i,j) = \Sigma_{x,y}(f(x+i,y+j) - f(x,y))^2$
- With the assumption i and j are small, by using the Taylor theorem:

$$f(x+i,y+j) \simeq f(x,y) + \frac{\partial f}{\partial x} \cdot i + \frac{\partial f}{\partial y} \cdot j = f_x \cdot i + f_y \cdot j + f(x,y)$$

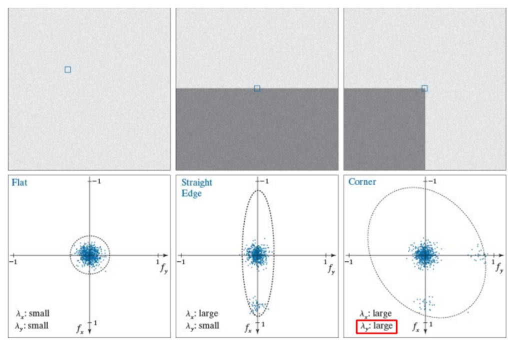
- Thus, $(f(x+i,y+j) - f(x,y))^2$ can be quantified as,

$$(f(x+i,y+j) - f(x,y))^2 = \begin{bmatrix} i & j \end{bmatrix} \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \tag{4}$$

- Therefor,

$$Dis(i,j) = \begin{bmatrix} i & j \end{bmatrix} \begin{bmatrix} \Sigma f_x^2 & \Sigma f_x f_y \\ \Sigma f_x f_y & \Sigma f_y^2 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i & j \end{bmatrix} C \begin{bmatrix} i \\ j \end{bmatrix} \tag{5}$$

Based on **intensity change**, it is required to decide whether it is an **edge** or **corner** or **flat surface**. Dis(i,j) represents the function of an ellipse.



https://cseweb.ucsd.edu/classes/wi21/cse152A-a/lec6.pdf

1. Do the eigenvalues represent the edges?
2. What happens when both eigenvalues are very small?
3. If it contains corners what can you say about corresponding eigenvalues and its directions?

The Laplace Operator

The Laplace operator is defined as two gradient vector operators.

$$\Delta f = \bigtriangledown \cdot \bigtriangledown f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & . & . & . & \frac{\partial f}{\partial x_n} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ . \\ . \\ . \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \Sigma_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{6}$$

When n equals 2,

$$\bigtriangledown^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

In discrete form,

$$\frac{\partial^2 f}{\partial x^2} = 2f(x, y) - f(x+1, y) - f(x-1, y)$$

$$\frac{\partial^2 f}{\partial y^2} = 2f(x, y) - f(x, y+1) - f(x, y-1)$$

Laplacian of Gaussian (LoG)

The **Laplace operation** does not know whether it detects **edges or noise**.

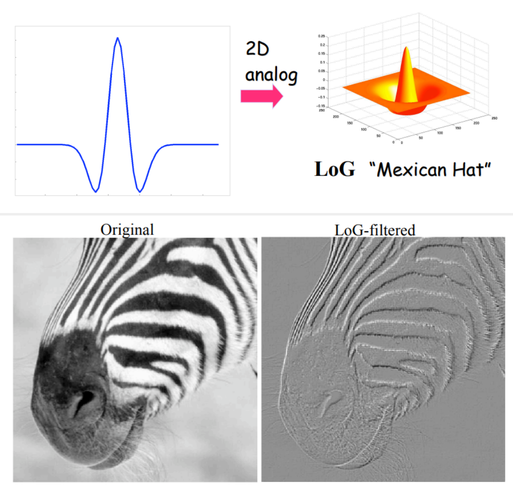
$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

$$G'_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \cdot -\frac{1}{2\sigma^2}(2x + 2y) \exp -\frac{x^2 + y^2}{2\sigma^2}$$

$$G''_\sigma(x, y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{\sigma^2}\right) exp \frac{-(x^2 + y^2)}{2\sigma^2}$$

for a given σ. The LoG operator takes the second derivative. Therefore, if image is basically uniform, the LoG will give zero. Wherever a change occurs, the LoG will give a positive response on the darker side and a negative response on the lighter side. At a sharp edge between two regions, the response will be **zero away from the edge, positive just to one side, negative just to the other side, zero at some point in between on the edge itself**

Laplacian of Gaussian (LoG)

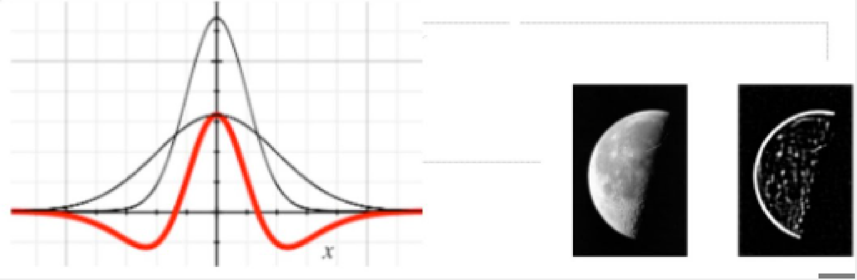


https://www.cse.psu.edu/~rtc12/CSE486/lecture11.pdf

Difference of Gaussian (DoG)

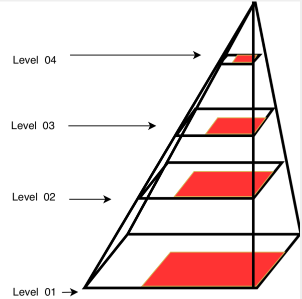
DoG is a kind of an approximation for the LoG which is defined as,

$$DoG = G_{\sigma_1}(x, y) - G_{\sigma_2}(x, y) \tag{7}$$



https://www.youtube.com/watch?v=6kvKzONBQK4

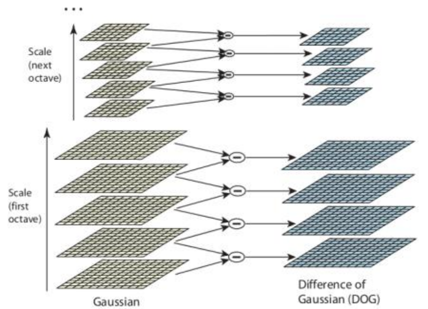
The Gaussian pyramids are made of **convoluting set of Gaussian kernels** with an original image where each layer is constructed by downsampling previous layered image whereas the Laplacian is constructed as a difference between original and after applying a low pass filter (Gaussian filter).



Level 01 is the filtered original image and followed layers from 2 to 4 are stacked after applying a selected filter and scale down with pre-defined factor e.g., 2, 4

Scale Invariant Feature Transform (SIFT)

SIFT uses **DoG** instead of **LoG** because its little costly. The first octave can be made of n number of filtered images with difference σ values. As stated in the original paper, it is sufficient to use 5 images at the initial stage



https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html

Scale Invariant Feature Transform (SIFT)

In order to detect the extreme (minimum or maximum), each pixel compares with its eight neighbours and 9 pixels in the next and previous scales. If it is a local extreme it will be considered as a **potential key point**.