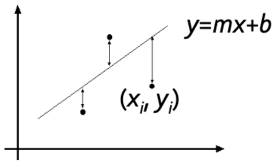
2-1 Fitting: Least squares, RANSAC, Hough Transform

Least squares line fitting
- **not rotation-invariant**
- fails completely for **vertical lines**

- **Data: $(x_1, y_1), \ldots, (x_n, y_n)$**
- **Line equation: $y_i = m x_i + b$**
- **Find (m, b) to minimize**

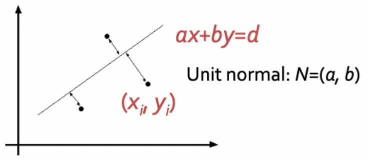
$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$



Total last squares

- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$
- Find (a, b, d) to minimize the sum of squared *perpendicular* distances

$$\boxed{E = \sum_{i=1}^{n} (ax_i + by_i - d)^2}$$

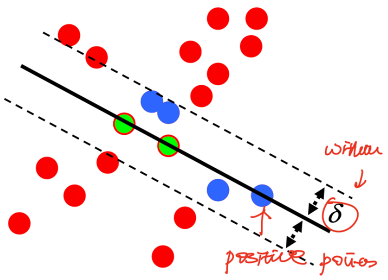


- Problem: squared error heavily penalized **outliers**

Robust estimators
- a nonlinear optimization problem

RANSAC

Algorithm:
1. **Sample** the **number of points** required to fit the model
2. **Solve for model parameters** using samples
3. **Score** by the **fraction of inliers** within a preset **threshold** of the model



Parameters:
- S: # of trials
- k: # of sampled points
- δ: distance threshold

Affine Transformation estimation with RANSAC

1. Randomly sample k data
2. Estimate the affine transformation T by solving Mx = b
3. Score by computing the number of inliers satisfying $|Tp - p'|^2 < \delta^2$ from all matches

Repeat 1 – 3 steps S times (T1 … Ts)

4. Select the best affine transformation TB
5. Re-estimate the affine transformation by solving Mx = b with **TB's inliers**

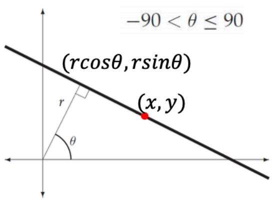
RANSAC Pros and Cons

Pros (3):
- **Robust** to **outliers**
- **Larger number** of objective function **parameters**
- **Optimization parameters** are easier to choose

Cons (4):
- **computational** burden
- Not good for **multiple fits**
- lots of **parameters** to **tune**
- Does not work well for **low inlier ratios**

Application (2):
- image stitching
- Estimating fundamental matrix

Hough Transform: fitting multiple lines



$$\frac{y - rsin\theta}{x - rcos\theta} = -\tan(90 - \theta) = -\frac{cos\theta}{sin\theta}$$

$$\Rightarrow \boxed{xcos\theta + ysin\theta = r}$$

- discretize θ
- Accumulator array contains how many times each value of (r, θ) appears in table

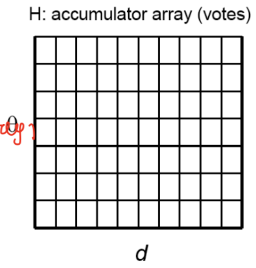
Dealing with Noise

- Choose a **good discretization**
  - **Too course: large votes obtain** when too many different lines correspond to a single point
  - **Too fine: miss lines since some points are not exactly collinear**
- Increment neighboring bins (**smoothing in accumulator array**)
- Get rid of **irrelevant features**

Hough Transform Algorithm

```
Initialize accumulator H to all zeros
for each feature point (x,y) in the image
   for θ = 0 to 180
      ρ = x cos θ + y sin θ
      H(θ, ρ) = H(θ, ρ) + 1
   end
end
Find the value(s) of (θ, ρ) where H(θ,ρ) is
a local maximum
The detected line in the image is given by
ρ = x cos θ + y sin θ
```

⇒ histogram (accumulator array)



$H(\theta, \rho)$ should be local maximum

Hough Transform Pros and Cons

Pros (3):

- robust to **outliers**
- **Efficient**
- Provide **multiple good fits**

Cons (5):

- sensitive to **noise**
- **Bin size trade-off**
- Difficult to find **sweet point**
- Not suitable for **more than a few parameters**
- Grid size grows **exponentially**

Application:

- line fitting
- Object recognition

Question:

Suppose there are 3 points in x-y coordinate, (2,0), (1,1), (0,2).
Using polar representation, $x\cos\theta + y\sin\theta = r$, apply Hough Transform with $\theta = -45°, 0°, 45°, 90°$, and perform the line fitting. (Let $\sqrt{2} = 1.4$)

(x, y)	$-45°$	$0°$	$45°$	$90°$
(2, 0)	1.4	2	1.4	0
(1, 1)	0	1	1.4	1
(2, 1)	0.7	2	2.1	1
(1, 3)	−1.4	1	2.8	3
(2, 3)	−0.7	2	3.5	3
(4, 3)	0.7	4	4.9	3
(3, 4)	−0.7	3	4.9	4

r

$x\cos\theta + y\sin\theta = r$

(x,y)	−45°	0°	45°	90°
(2,0)	1.4	2	1.4	0
(1,1)	0	1	1.4	1
(0,2)	−1.4	0	1.4	2

$2\sin\theta = r$

↓ accumulated array

θ \ r	−1.4	0	1	1.4	2
−45°	1	1		1	
0°		1	1		1
45°				$\underline{\underline{3}}$ → accumulation	
90°		1	1		1

$x \cdot \cos 45° + y \cdot \sin 45° = \sqrt{2}$

$\frac{\sqrt{2}}{2}(x+y) = \sqrt{2}$

$x + y = 2$

$y = 2 - x$

Image Segmentation

Optimum Global Thresholding (Bimodal signal)
Key Idea: exhaustively search for the threshold that minimizes the within–class variance, maximizes the between–class variance

Adaptive Thresholding using Moving Averages

$$g(i,j) = \begin{cases} 1 & if\ I(i,j) > T(i,j) \\ 0 & otherwise \end{cases}$$

$$T(i,j) = b \times m(i,j)$$

threshold determine pixel by pixel

Using the mean intensity $m(i,j)$

$$m(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(i+s, j+t)$$

filtering

$(2a+1)(2b+1)$ kernel size

$\frac{1}{(2a+1)(2b+1)} \cdot I \Rightarrow$ mean filtering

Question:

What is the value (1 or 0) of a grey pixel after
1) Global thresholding (with threshold = 8) and
2) Adaptive thresholding with 3x3 kernel with the parameter b = 0.8

4	2	8	10	6
8	6	6	2	6
0	2	7	4	2
8	9	5	9	8
0	2	0	2	0

(1) 0
(2) T = $0.8 \cdot m(i,j)$ = 4.44, 7 > 4,44 —> 1

K–mean Clustering
1. Randomly **initialize** the cluster centers
2. Given cluster centers, **determine points in each cluster** (for each point, find the closest cluster, put point into that cluster)
3. Given points in each cluster, **solve for a new cluster center**
4. If **cluster center changed** repeat step 2

K–means Pros and Cons
Pros (2):
– simple, fast to **compute**
– converges to **local minimum** of within–cluster squared error
Cons (5):
– k?
– Sensitive to **initial centers**
– Sensitive to **outliers**
– Detects **spherical clusters**
– Assume **mean** can be computed

Feature space on Image Segmentation

Grouping pixels based on:

- color similarity
- Intensity + position
- Texture similarity

Question 2 (a):

a) This question is about **fitting**.

[8 marks]

i) The figure below shows the pseudo code for estimating an affine transformation with RANSAC. Fill out the blanks in the code.

(5 marks)

```
Input: A set of N matched points MP = {(p_0,p'_0),(p_1,p'_1) ..., (p_{N-1},p'_{N-1})}
Output: Affine transform T_F

S: the number of trials
count_mat: S×1 vector
IN: a set of inliers

Initialize count_mat to 0
for i = 0 ~ S-1
   Randomly select k matched points from MP (Usually, k=3,4)
   Estimate an affine transformation T_i with [    (1)    ]
   for j = 0 ~ N-1
      if [  (2)  ]
         count_mat[i]++

Choose the best affine transformation T ← T_K where [    (3)    ]

IN = NULL
for j = 0 ~ N-1
   if [  (4)  ]
    IN ← IN ∪ {(p_j,p'_j)}

Re-estimate an affine transformation T_F with IN
```

ii) Explain the *Hough transform algorithm* with illustrations.

(3 marks)

(1) ① ~~MP = b~~ k matched points

② $|T_i \cdot P_j - P_j'|^2 < \delta^2$

③ $K = \arg\max_i \text{count\_mat}[i]$

④ $|TP_j - P_j'|^2 < \delta^2$

(2) Hough transform is a transform method from (x,y) domain to (r,θ) domain, in which case lines in (x,y) will be points in (r,θ) domain.

① For a line: $x\cos\theta + y\sin\theta = r$

② Discretization. for θ, for each θ and (x,y) point, calculate r.

③ accumulate array contains how many times each value of (r,θ) appears in table.

Algorithm:

Initialize accumulator with all 0.

for each point (x,y) in image.

  for θ = 0 to 180

    $\rho = x\cos\theta + y\sin\theta$

    $H(\theta,\rho) = H(\theta,\rho) + 1$

  end

end

Find the values of (θ,ρ) that $H(\theta,\rho)$ is the local maximum.

Then the corresponding line is found: $x\cos\theta + y\sin\theta = \rho$.

Question 2 (b):

b) This question is about **grouping**.

[8 marks]

i) The figure below shows the pseudo code for *K-means algorithm*. Fill out the blanks.

(3 marks)

1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put $\boxed{(1)}$
3. Given points in each cluster, solve for c_i
   - Set c_i to be the $\boxed{(2)}$
4. If c_i have changed, repeat $\boxed{(3)}$

ii) State three advantages and two drawbacks of *Mean-shift algorithm*.

(5 marks)

(1) ① point p into the cluster with center c_i.

② mean of points in cluster i

③ step 2.

(2) Advantages:

① simple algorithm

② fast and efficient to compute

③ converges to the local minimum of within-cluster squared error

Disadvantages:

① it is not good if initial centers are not chosen

② sensitive to outliers

③ Assuming means can be computed.

④ How to choose K (# of centers)

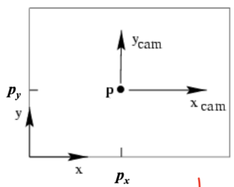
2-3 Calibration

Camera Calibration:
Figure out transformation from world coordinate system to image coordinate system

Normalized coordinate system (camera):
Camera center is at the **origin**, the **principal axis** is the **z-axis**; x and y of the image plane are **parallel** to x and y axis of the world

Principal point: point where **principal axis intersects the image plane**

Principal point offset



principal point: (p_x, p_y)

$$\underbrace{\underbrace{\begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}}_{\substack{\text{calibration matrix} \\ K}} \underbrace{\left[\begin{array}{ccc|c} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{array}\right]}_{\substack{\text{projection matrix} \\ [I\,|\,0]}}}_{P = K[I\,|\,0]} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

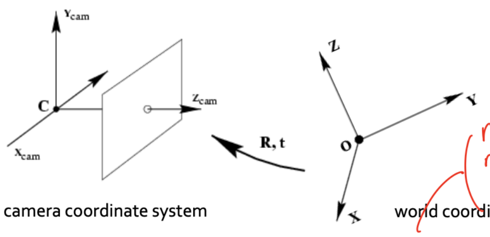
(augmented matrix)

Pixel coordinates

m_x pixels per meter in horizontal direction,
m_y pixels per meter in vertical direction

$$K = \underbrace{\begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix}}_{\text{pixels/m}} \underbrace{\begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}}_{\text{m}} = \underbrace{\begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}}_{\text{pixels}}$$

Camera Rotation and translation



camera coordinate system world coordinate system

3x3

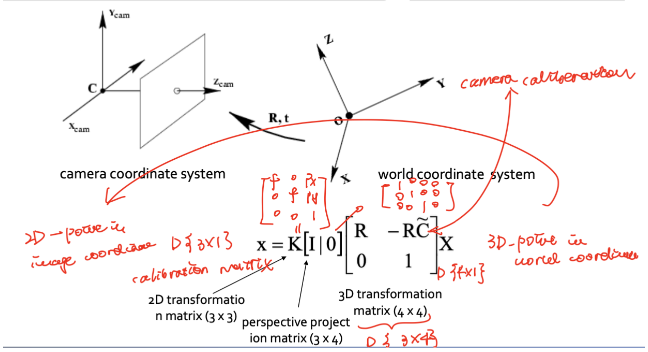
$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

$$\tilde{X}_{cam} = R(\tilde{X} - \tilde{C})$$

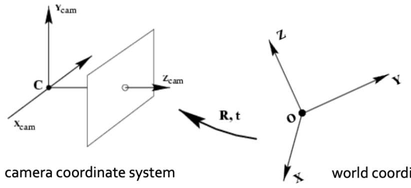
$$\begin{pmatrix} \tilde{X}_{cam} \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} \overset{3\times3}{R} & -R\tilde{C} \\ 0 & 1 \end{bmatrix}}_{\text{3D transformation matrix } (4 \times 4)} \begin{pmatrix} \tilde{X} \\ 1 \end{pmatrix}$$

$$\underbrace{\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}}_{3\times1}$$

Whole Process



$$x = K[I \mid 0]\begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X$$



img ↔ cam   cam ↔ world

$\tilde{C} \Rightarrow$ coordinates of camera center

$$x = K[R \mid t]X \qquad t = -R\tilde{C}$$

$R \Rightarrow$ 3D rotation ma

$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix}\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Cameras parameters

Intrinsic parameters:

- principal point coordinate (px, py)
- Focal length f
- Pixel magnification factors (mx, my)

Extrinsic parameters:

- rotation and translation relative to world coordinate system

Camera Calibration in vanishing points

- for **2/3 finite vanishing points**: can solve **focal length, principal point**
- for **1 finite vanishing point**, **cannot** solve focal length, **principal point is the third vanishing point**

Calibration and Rotation from vanishing points

Let us align the world coordinate system with three orthogonal vanishing directions in the scene:

$$\mathbf{e}_1 = \begin{bmatrix}1\\0\\0\end{bmatrix},\quad \mathbf{e}_2 = \begin{bmatrix}0\\1\\0\end{bmatrix},\quad \mathbf{e}_3 = \begin{bmatrix}0\\0\\1\end{bmatrix} \qquad \lambda_i \mathbf{v}_i = \mathbf{KRe}_i$$

$$\mathbf{e}_i = \lambda_i \mathbf{R}^T \mathbf{K}^{-1} \mathbf{v}_i$$

Orthogonality constraint: $\mathbf{e}_i^T \mathbf{e}_j = 0$

$$\mathbf{v}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{v}_j = 0$$

focal length: f
principle point

Rotation disappears, each pair of vanishing points gives constraint on focal length and principal point

Constraints on vanishing points: $\lambda_i \mathbf{v}_i = \mathbf{KRe}_i$

After solving for the calibration matrix: $\lambda_i \mathbf{K}^{-1} \mathbf{v}_i = \mathbf{Re}_i$

Notice: $\mathbf{Re}_1 = [\,\mathbf{r}_1 \;\; \mathbf{r}_2 \;\; \mathbf{r}_3\,]\begin{bmatrix}1\\0\\0\end{bmatrix} = \mathbf{r}_1$

$Re_i = r_i$

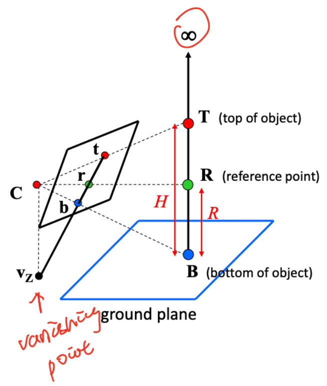
Thus, $\mathbf{r}_i = \lambda_i \mathbf{K}^{-1} \mathbf{v}_i$

Get λ_i by using the constraint $\|\mathbf{r}_i\|^2 = 1$.

$r_1 = \begin{bmatrix} \cos\theta \\ -\sin\theta \\ 0 \end{bmatrix} \quad r_2 = \begin{bmatrix} \sin\theta \\ \cos\theta \\ 0 \end{bmatrix} \quad \theta = 1$

$\|r_1\|^2 = \|r_2\|^2 = \|r_3\|^2 = 1$

Measuring Height



$$\frac{\|\mathbf{T}-\mathbf{B}\|\,\|\infty-\mathbf{R}\|}{\|\mathbf{R}-\mathbf{B}\|\,\|\infty-\mathbf{T}\|} = \frac{H}{R}$$

scene cross ratio

$$\frac{\|\mathbf{t}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{r}\|}{\|\mathbf{r}-\mathbf{b}\|\,\|\mathbf{v}_Z-\mathbf{t}\|} = \frac{H}{R}$$

image cross ratio

P
$\tilde{P}$
P

Question 2 (c):

c) This question is about **calibration**.

[3 marks]

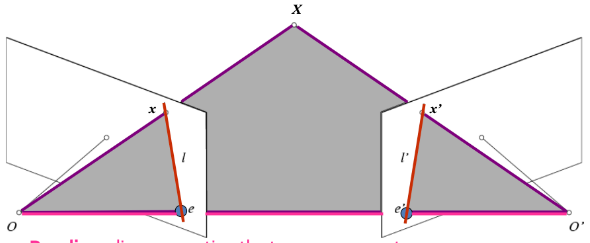
State **1)** the definition of camera calibration with **2)** the illustration of image, camera, and world coordinates.

(3 marks)

Camera calibration is the transformation from world coordinate system to the image coordinate system. For the first step it needs to transform from world coordinate system to the camera coordinate system by rotation and translation. Then, we can use calibration matrix to transform the camera coordinate system to the image coordinate system.

2–5: Stereo

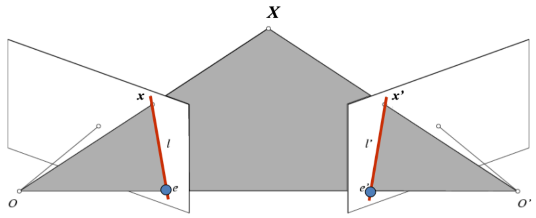
Epipolar geometry



- **Baseline** – line connecting the two camera centers
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipoles**
  = intersections of baseline with image planes
  = projections of the other camera center
  = vanishing points of the motion direction
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

— Potential matches for x have to lie on the corresponding epipolar line l'
— Potential matches for x' have to lie on the corresponding epipolar line l
Baseline | Epipolar Plane | Epipoles | Epipolar Lines

Epipolar constraint: Calibrated Case

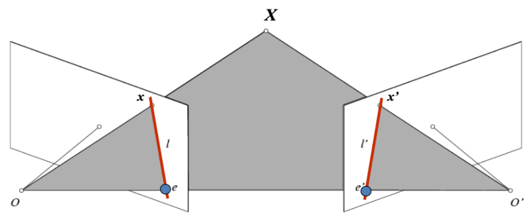


$$\boxed{x'^T E x = 0}$$

- $\boldsymbol{Ex}$ is the epipolar line associated with $\boldsymbol{x}$ ($l' = \boldsymbol{Ex}$)
- $\boldsymbol{E}^T\boldsymbol{x}'$ is the epipolar line associated with $\boldsymbol{x}'$ ($l = \boldsymbol{E}^T\boldsymbol{x}'$)
- $\boldsymbol{Ee} = 0$ and $E^T\boldsymbol{e}' = 0$
- $\boldsymbol{E}$ is singular (rank two)
- $\boldsymbol{E}$ has five degrees of freedom

$E = t \times R$, Essential matrix

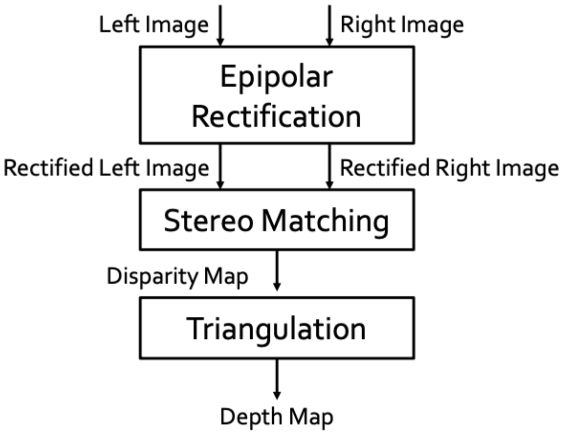
Epipolar constraint: Uncalibrated Case



$$\hat{x}'^T E \hat{x} = 0 \Rightarrow x'^T F x = 0 \quad \text{with} \quad F = K'^{-T} E K^{-1}$$

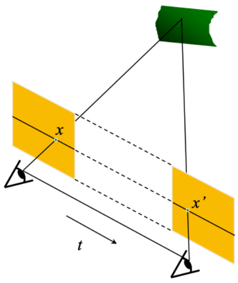
- $\boldsymbol{Fx}$ is the epipolar line associated with $\boldsymbol{x}$ ($\boldsymbol{l}' = \boldsymbol{Fx}$)
- $\boldsymbol{F}^T\boldsymbol{x}'$ is the epipolar line associated with $\boldsymbol{x}'$ ($\boldsymbol{l} = \boldsymbol{F}^T\boldsymbol{x}'$)
- $\boldsymbol{Fe} = 0$ and $\boldsymbol{F}^T\boldsymbol{e}' = 0$
- $\boldsymbol{F}$ is singular (rank two)
- $\boldsymbol{F}$ has *seven* degrees of freedom

Computational Stereo Pipeline



Epipolar rectification
Find corresponding epipolar line in the right image

Simple Case: Parallel Images



Epipolar constraint:

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

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

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

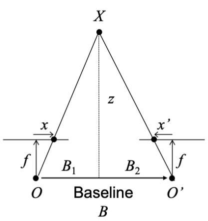
$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0 \qquad \begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -T \\ Tv \end{pmatrix} = 0 \qquad Tv' = Tv$$

Stereo Image Rectification

1. Rotate the **right camera** by **R** (compute E to get R)
2. Rotate the left camera so that the **epipole is at infinity** (Rrect)
3. Rotate the right camera so that the **epipole is at infinity** (Rrect)
4. Adjust the **scale**

Stereo Matching
Examine all pixels on the epipolar line and pick the best match

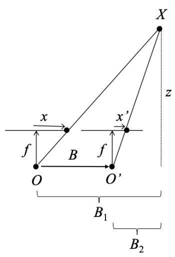
Depth from Disparity



$$\frac{x}{f} = \frac{B_1}{z} \qquad \frac{-x'}{f} = \frac{B_2}{z}$$

$$\frac{x - x'}{f} = \frac{B_1 + B_2}{z}$$

$$\boxed{disparity = x - x' = \frac{B \cdot f}{z}}$$



$$\frac{x}{f} = \frac{B_1}{z} \qquad \frac{x'}{f} = \frac{B_2}{z}$$

$$\frac{x - x'}{f} = \frac{B_1 - B_2}{z}$$

$$\boxed{disparity = x - x' = \frac{B \cdot f}{z}}$$

Stereo Matching

Window–based matching

In a formal way,

– the disparity d_x of a pixel $\boldsymbol{x}$ in the left image is computed as

$$\boxed{d_{\boldsymbol{x}} = \underset{0 \le d \le d_{max}}{\operatorname{argmin}} \sum_{\boldsymbol{q} \in W_{\boldsymbol{x}}} c(\boldsymbol{q}, \boldsymbol{q} - \boldsymbol{d})}$$

– Where Let's split this equation into three steps!

- argmin returns the value at which the function takes a minimum
- d_{max} is a parameter defining the maximum disparity (search range).
- $W_{\boldsymbol{x}}$ is the set of all pixels inside the window centred on $\boldsymbol{x}$
- $c(\boldsymbol{q}, \boldsymbol{q} - \boldsymbol{d})$ is a function that computes the colour difference between a pixel $\boldsymbol{q}$ of the left image and a pixel $\boldsymbol{q} - \boldsymbol{d}$ of the right image (e.g. Sum of absolute difference in RGB values)

Effect of window size

– Smaller window: more detail, more noise

– Larger window: smoother disparity maps, less detail

Problem of untextured regions

– low **texture**

– **Aperture** problem

– **Repetitive** pattern

Problem of **Foreground fattening**

– foreground objects are clearly **enlarged** when using **large kernels**

Adaptive support weight — solution for foreground fattening

Assumption:

1. two point are likely to lie on the **same disparity** if they are **similar in color**
2. Only **pixels** that lie on the **same disparity** contribute to the aggregated matching costs

$$w(\boldsymbol{x}, \boldsymbol{q}) = \boldsymbol{exp}\left(-\left(\frac{\Delta c_{xq}}{\sigma_c} + \frac{\Delta s_{xq}}{\sigma_s}\right)\right)$$

Δc_{xq}: colour distance between **x** and **q**

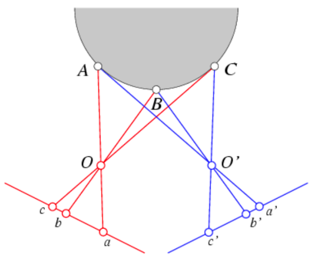
Δs_{xq}: spatial distance between **x** and **q**

Non–local constraints

– Uniqueness: for any point in one image, **at most one matching point** in another image

– Ordering: corresponding points should be in the **same order in both views**

– Smoothness: **disparity values to change slowly**



Question 2 (d):

d) This question is about **stereo matching**.

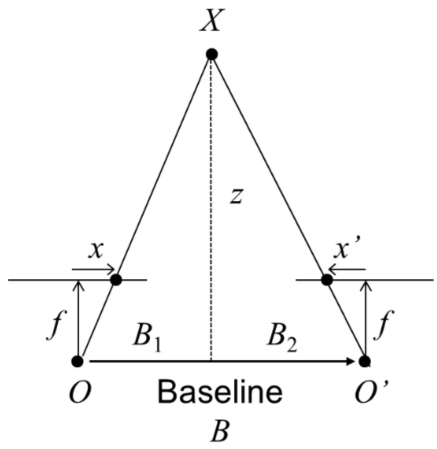
[6 marks]

i) Explain the uniqueness constraint used for improving local stereo matching performance. Provide your description with an illustration.

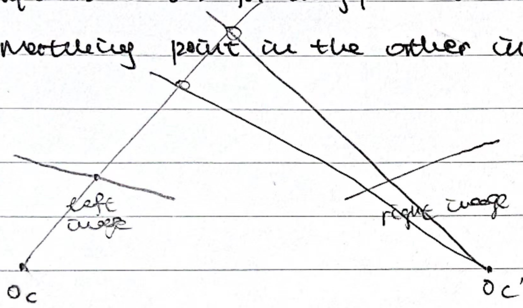
(3 marks)

ii) In the figure below, derive the relationship between disparity, $x - x'$ and depth z.

(3 marks)



(1) uniqueness means for any point in one image, there should be at most one matching point in the other image.



→ violate uniqueness constraint.

(2) $\frac{x}{f} = \frac{B_1}{z}$ $\frac{-x'}{f} = \frac{B_2}{z}$

$\frac{x-x'}{f} = \frac{B_1+B_2}{z}$

disparity $= x - x' = \frac{B_1+B_2}{z} \cdot f = \frac{B}{z} \cdot f$