CS 4186 Computer Vision Review

Exam Info

- Time: 18:30 - 20:30 05/May (20:45 for submission)
- Two monitoring device
- Open book, online
- Materials
  - Calculator
  - Books, notes (hard copy)
- Types of questions
  - Short answer question (2-3 sentences)
  - Calculations
- The first page with academic honesty should be submitted together with the answers, as you need to reaffirm the academic honesty policy.

  Academic Honesty

  I pledge that the answers in this examination are my own and that I will not seek or obtain an unfair advantage in producing these answers. Specifically,

  - *I will not plagiarize (copy without citation) from any source;*
  - *I will not communicate or attempt to communicate with any other person during the examination; neither will I give or attempt to give assistance to another student taking the examination; and*
  - *I will use only approved devices (e.g., calculators) and/or approved device models.*
  - *I understand that any act of academic dishonesty can lead to disciplinary action.*

  I pledge to follow the Rules on Academic Honesty and understand that violations may lead to severe penalties.

  Student ID: 55199998

  Name: ZHANG Deheng

1 Image Filtering

1.1 Image

- Image is a grid of intensity values
- Can be represented as a function $f : \mathbb{R}^2 \mapsto \mathbb{R}$
  - $f(x, y)$ is the intensity value at position (x, y)
- A digital image is a discrete (sampled, quantized) version of this function

1.2 Linear Filter

- Image transformation: any functions to change the range of the image function
- Filtering: form a new image whose pixels are a combination of the original pixels (in a specific range)
  - get useful information from images
  - enhance the image (change the visual effect, remove noise)
- Linear Filtering: the target pixel is a linear combination (weighted sum) of its neighbour. The operator is called a kernel (or mask, filter)
  - H represents the kernel of size $(2k+1) \times (2k+1)$, and G represents the output image
  - Cross-correlation:

$$
\begin{aligned}
G &= H \otimes F \\
G[i,j] &= \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]
\end{aligned} \tag{1}
$$

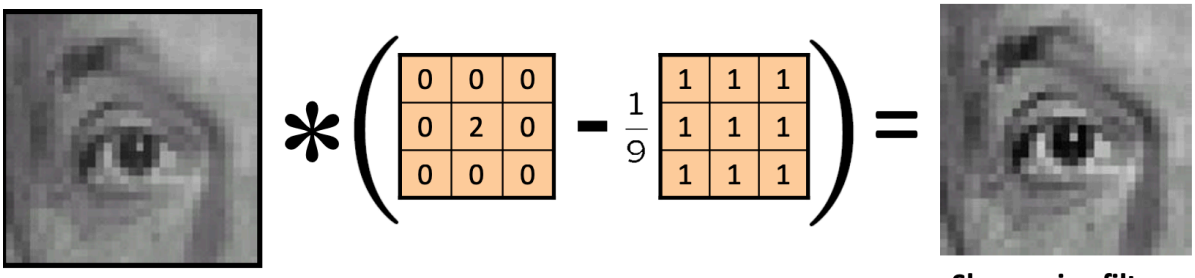
  - Convolution:

$$
\begin{aligned}
G &= H * F \\
G[i,j] &= \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v] \\
s.t. G &= H * F = F * H
\end{aligned} \tag{2}
$$

    - Mean filtering & Box filter: can **blur** the image

$$
\begin{aligned}
n &= (2k+1) \times (2k+1) \\
\forall u, v \in [-k, k] \cap \mathbb{Z}, H[u,v] &= \frac{1}{n}
\end{aligned} \tag{3}
$$

    - Image sharpening: original image adds the difference between it and the blurred image



Original

Sharpening filter

    - Gaussian filter: the filter is defined by 2D Gaussian distribution function

$$
G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{4}
$$

      - The filter sums to 1
      - σ affects the feature's scale
      - convolution with itself is another Gaussian
      - Can also be used to blur the image (mean filter may intrdouce grid-like high-frequency noise to the image), while Guassian filter removes the "high-frequency" components from the image (low-pass filter)

2 Edge Detection

2.1 Image Derivatives

- Edges correspond to extrema (maximum or minimum) of first order derivative of the image intensity function
- Two options
  - Option 1: reonstruct a continuous image f then compute the derivative
  - **Option 2**: take discrete derivative (finite difference)

$$\frac{\delta f}{\delta x}[x, y] \approx F[x+1, y] - F[x, y] \tag{5}$$

$\frac{\partial f}{\partial x}$:

1	-1	

H_x

$\frac{\partial f}{\partial y}$:

	-1	
	1	

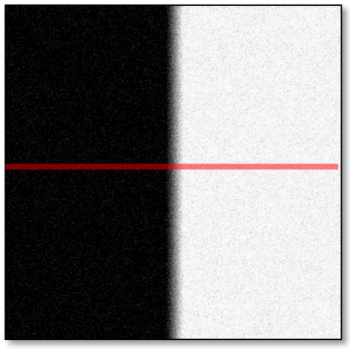
H_y

2.2 Image Gradient

- used in the SIFT detector.
- The gradient is defined as $\nabla f = [I_x, I_y]$
  - **gradient magnitude**: $||\nabla f|| = \sqrt{(I_x)^2 + (I_y)^2}$
  - **gradient direction**: $\theta = tan^{-1}(\frac{I_y}{I_x})$

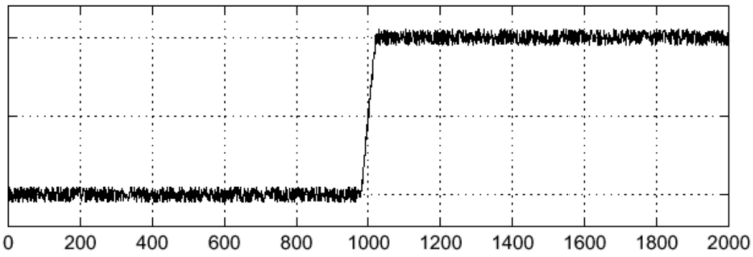
2.3 Sobel (smooth before derivative)

- The edge corresponds to the peak in the first order derivatives
- There might be noise for the image => many peaks

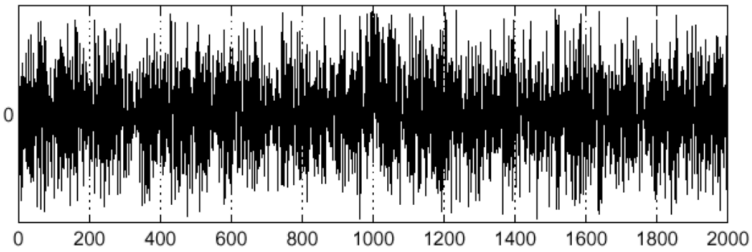


Noisy input image

$f(x)$



$\frac{d}{dx}f(x)$



- Smooth by using Gaussian filter and then apply the associative property of convolution

$$\frac{d}{dx}(f * h) = (\frac{d}{dx}h) * f \tag{6}$$

- Sobel operator: The common approximation of derivative of Gaussian w.r.t x and y

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

s_x

$\frac{1}{8}$

1	2	1
0	0	0
-1	-2	-1

s_y

  - $\frac{1}{8}$ is omitted in the stardard definition
  - Added to get the right gradient magnitude

2.4 Laplacian of Gaussian (LoG)

- The edges correspond to zero-crossing in the second order derivatives
- Smooth by Gaussian:
  - 1D example

$$(\frac{\delta^2}{\delta x^2}h) * f \tag{7}$$

- 2D case: observe the LoG function and construct the filter, then do the convolution

$$\nabla^2 h = \frac{\delta^2 h}{\delta x^2} + \frac{\delta^2 h}{\delta y^2}$$
$$I_{xx} + I_{yy} = (\nabla^2 h) * I \tag{8}$$

2.5 Canny Edge Detector

- Good edge detector
  - good detection: find all real edges, ignoring noise or other artifacts
  - good localisation:
    - edges detected are close to the true edges
    - one point only for each true edge point
- Canny edge detector
  - Filter image with derivative of Gaussian
  - Find the magnitude and orientation
  - Non-maximum suppression: ignore noise
    - get the gradient direction, and compare the edge strength along the direction
    - preserve the largest and suppress the others
  - Linking and thresholding (hysteresis):
    - Define two thresholds: low and high
      - > high: strong edge
      - < low: noise
      - Between: weak edge
    - Use the high threshold to start edge curves and the low threshold to continue them
      - 'Follow' edges starting from strong edge pixels
      - Continue them into weak edges

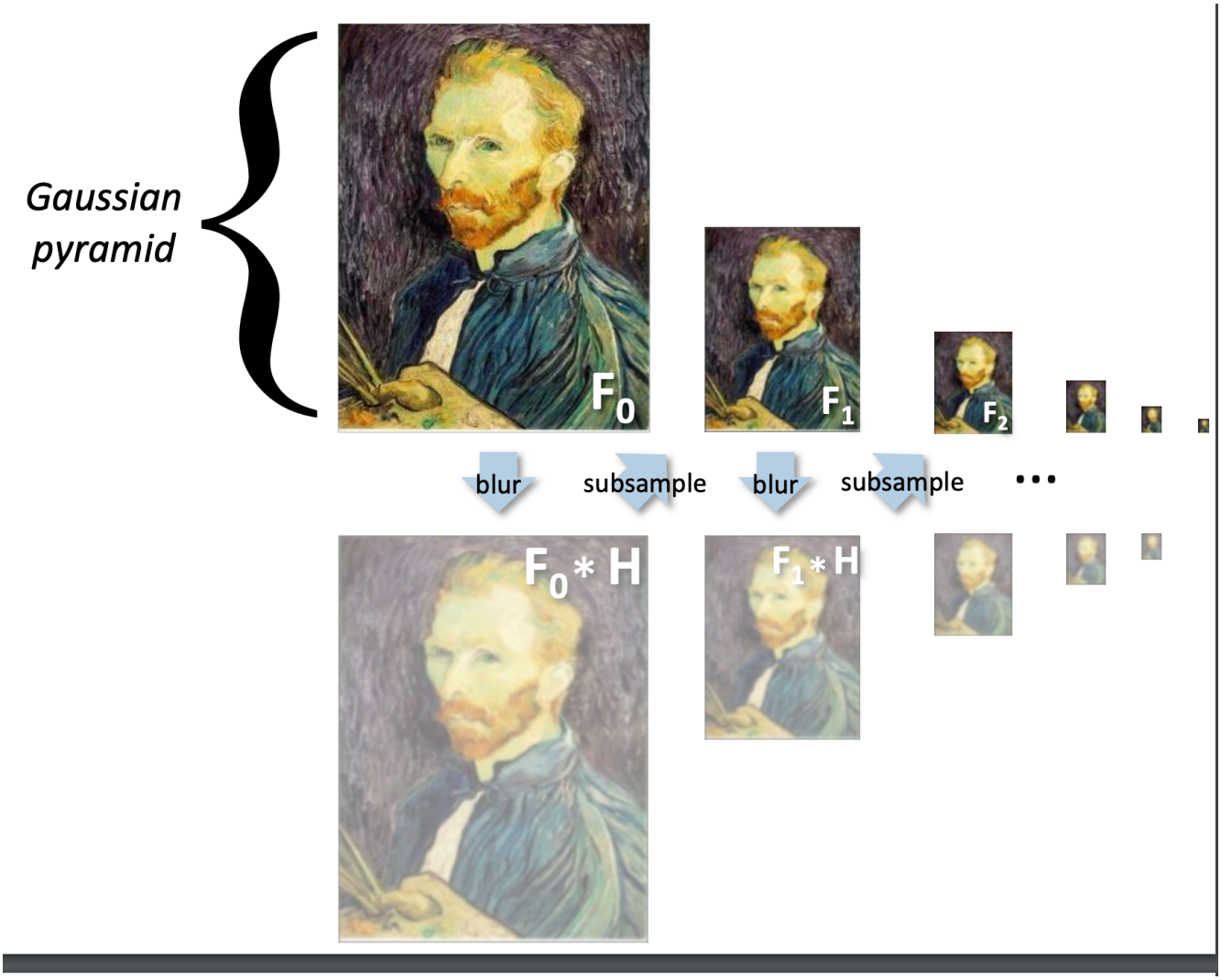
3 Image Sampling

###

- Resampling
  - upsampling: from low resolution to high resolution (guess the missing pixels)
  - Subsampling (downsampling): delelte some pixels
- nearest neibor

3.1 Gaussian Pyramid

- Gaussian pre-filtering: Filter the image by gaussian first, then sub-sample the image
- Gaussian Pyramid: sequence of blur and subsample



3.2 Image Upsampling

- **Nearest neighbour**: copy the pixel from the nearest pixel in the original image
- Average of neighbours
- Gaussian filter

4 Color and Texture

4.1 Color (no calculation in the exam)

- Histogram
  - Gray image histogram: $H[i]$ is the frequency of the grey tone i, $P[i]$ is the percentage of pixels that have gray tone i

$$h(i) = \sum_x \sum_y \mathbb{I}[f(x,y) = i] \tag{9}$$

  - RGB histograms
    - single 3D histogram $H[r,g,b]$ ($256 \times 256 \times 256$)
    - Make 3 histograms and concatenate them (3×256)
    - Create pseudo 8-bits color by using 3 bits of R 3 bits of G and 2 bits of B (1×256)
    - Normalized color space and 2D histogram (record r, g and calculate $b = 1 - r - g$)

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B} \tag{10}$$

- Intersection and match score:

$$\begin{aligned} intersection(h(I), h(M)) &= \sum_{j=1}^{numbins} min\{h(I)[j], h(M)[j]\} \\ match(h(I), h(M)) &= \frac{intersection(h(I), h(M))}{\sum_{j=1}^{numbins} h(M)[j]} \end{aligned} \tag{11}$$

- Property
  - Robust for 2D rotation
  - Not robust for 3D rotation

- Multi-scale spatial color representation
  - Divide an image into $N \times N$ grids at multiple scales
  - Compute color histogram for each grid
  - Concatenate all the histograms across scales and grids as a feature vector

4.2 Edge-based Texture Measures

- gradient magnitude
  - Use an edge detector as the first step in texture analysis.
    - The number of edge pixels in a fixed-size region tells us how busy that region is.
    - The directions of the edges also help characterize the texture
  - Edgeness per unit area: N represents the size of the unit area

$$F_{edgeness} = \frac{|\{p|gradient\_magnitude(p) \geq threshold\}|}{N} \tag{12}$$

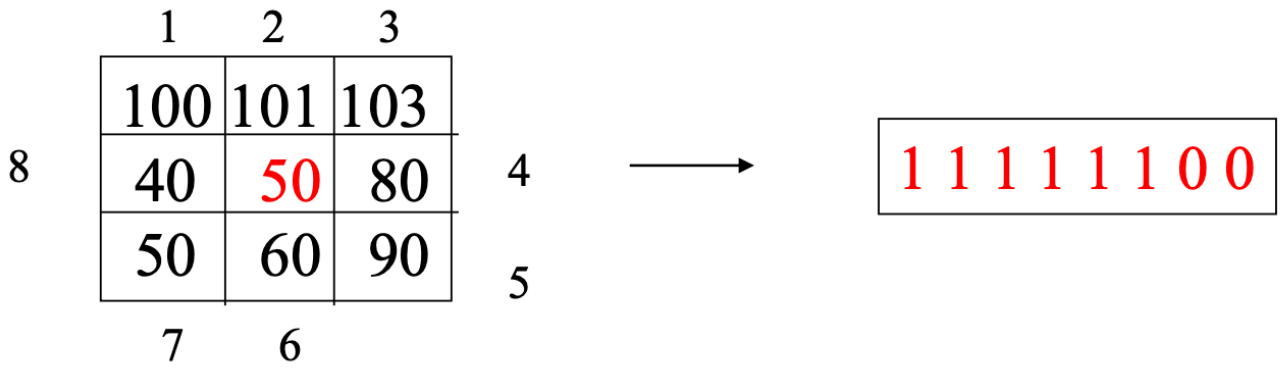
- magnitude & direction histogram (similar to SIFT)

$$F_{magdir} = (H_{magnitude}, H_{direction}) \tag{13}$$

  - these are normalized histograms of gradient magnitudes aned gradient directions
  - e.g. the direction bin can be set in $[0, 2\pi]$ with 16 bins

4.3 Local Binary Pattern Measure

- For each pixel, find the comparison of 3×3 window

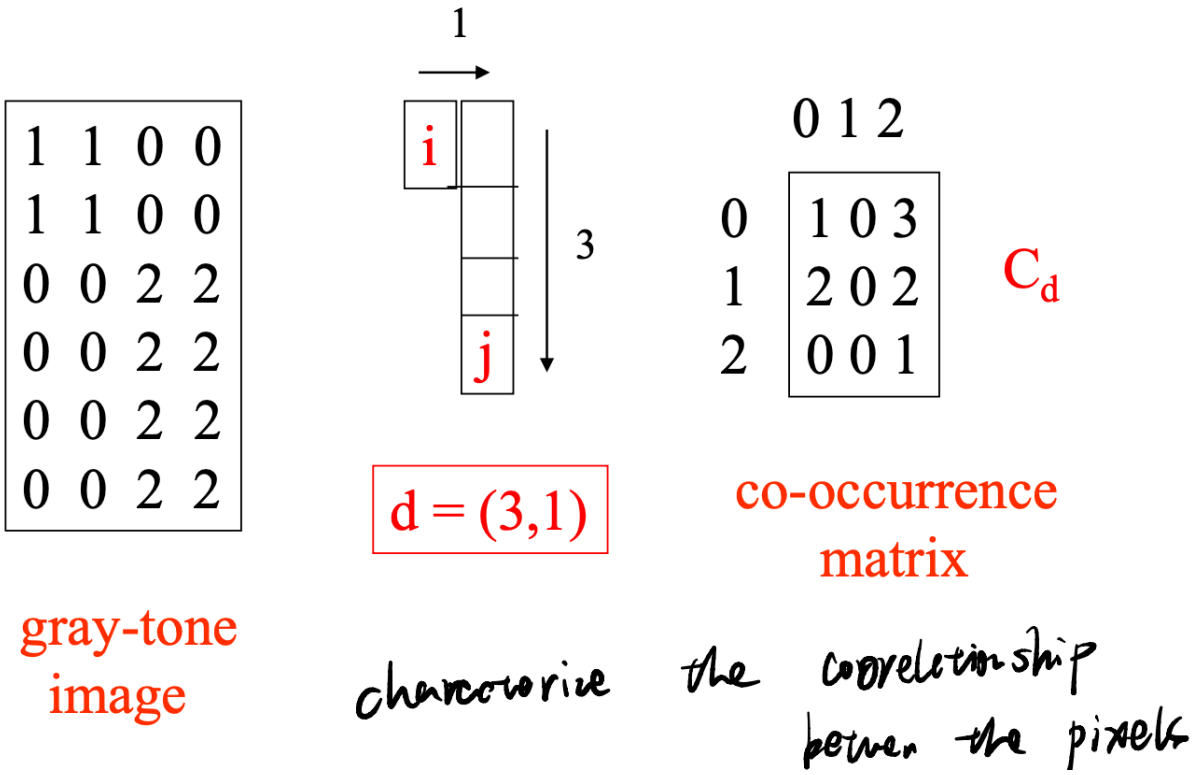


$$LBP_{p,r}(N_c) = \sum_{p=0}^{P-1} \mathbb{I}[N_p = N_c] 2^p \tag{14}$$

- N_c : center pixel
- N_p: neighbour pixel
- r: radius (for 3×3 cell, it is 1)

4.4 Co-occurrence Matrix Features

- Co-occurrence matrix is a 2D array in which
  - rows and columns are labelled by the possible pixel value (256×256)
  - $C_d[i, j]$ represents the value how many times value i co-occurs with value j in a particular spatial relationship represetned by a vector $\mathbf{d} = (d_r, d_c)$
  - We can normalize the matrix C by dividing all values by the sum of all the values to get the normalized co-ocurrence matrix N
- Example:



5 Corners and Blobs

- Interesting features
  - corner: defined as the intersection of two edges
  - blob: regions in the image that differ in properties, such as brightness or color, compared to the surrounding regions
- Application example: panorama stitching
  - Interest point detection - Harris corner detection or LoG blob detection
  - extract features - SIFT
  - match features - distance & ration distance
  - Align images (perspective transform)
- Invariant local features

    - geometric invariance: translation, rotation, scale
    - photometric invariance: brightness, exposure
    - Advantages:
        - Locality – features are local, so robust to occlusion and clutter
        - Quantity – hundreds or thousands in a single image
        - Distinctiveness: – can differentiate a large database of objects
        - Efficiency – real-time performance achievable
- Corner vs flat vs edge: move the window a little bit
    - corner: signifcant change
    - Edge: no change along the edge direction
    - Flat: no change in all direction

5.1 Harris corner detection

- SSD error:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2 \tag{15}$$

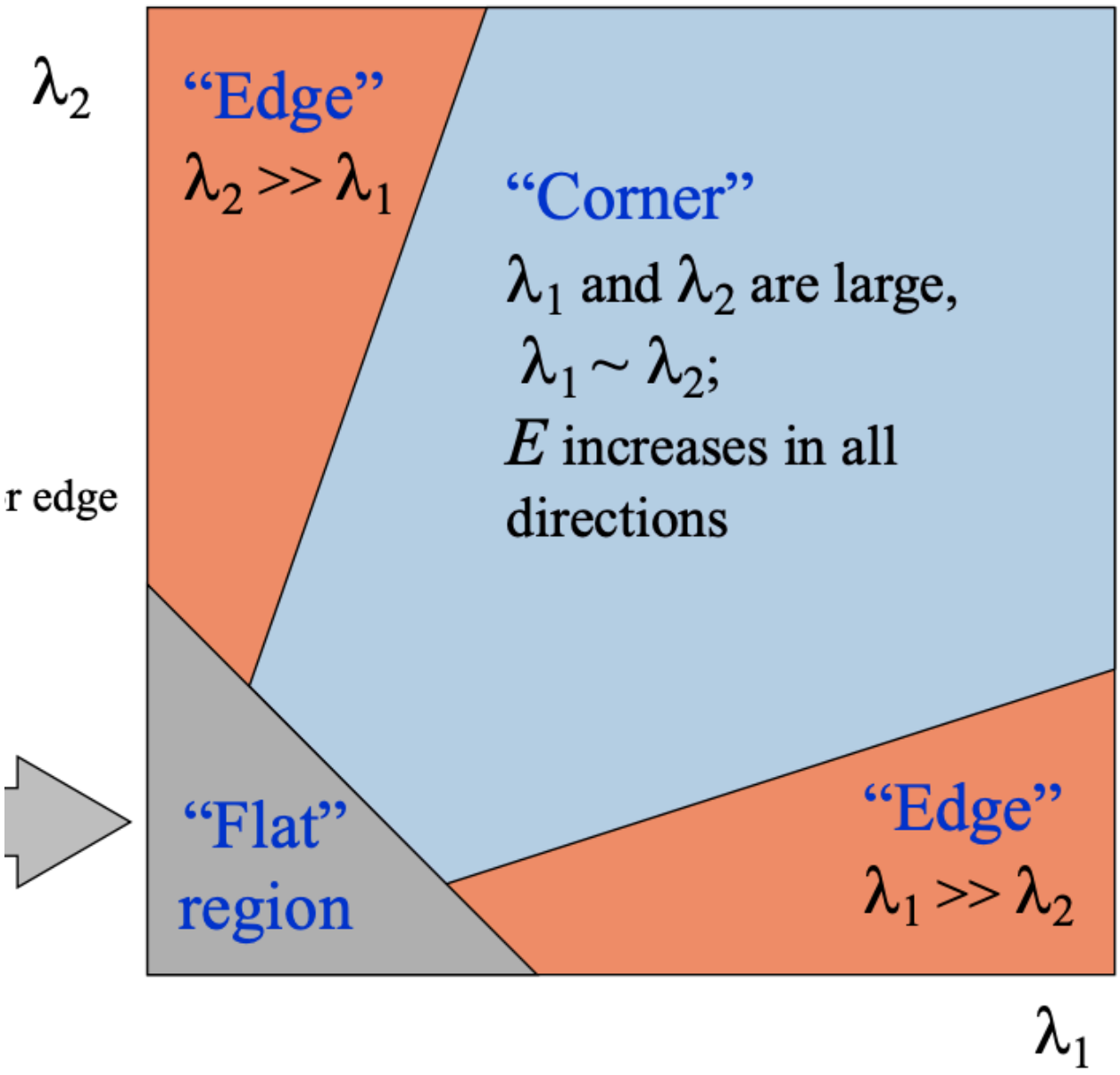
    - shift the window by (u,v)
    - summing up the squared difference (SSD)
    - search the high error patch
- Small motion assumption

$$\begin{aligned} I(x+u, y+v) &\approx I(x,y) + I_x u + I_y v \\ &\approx I(x,y) + [I_x, I_y][u,v]^T \\ E(u,v) &\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2 \\ &\approx Au^2 + 2Buv + Cv^2 = [u,v]\begin{bmatrix} A & B \\ B & C \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} \\ where\ A = \sum_{(x,y)\in W} I_x^2, B = \sum_{(x,y)\in W} I_x I_y, C = \sum_{(x,y)\in W} I_y^2, H = \begin{bmatrix} A & B \\ B & C \end{bmatrix} \end{aligned} \tag{16}$$

    - If the motion of the window is small, we can represent the shifted image as the first order Taylor expassion
- Eigenvalues analysis: H is a real symmetric matrix

$$\begin{aligned} H &= V\Lambda V^T\ s.t.\ V^T V = I \\ SSE &= [u,v]H\begin{bmatrix} u \\ v \end{bmatrix} = [u,v]V\Lambda V^T\begin{bmatrix} u \\ v \end{bmatrix} \\ V' &= \begin{bmatrix} m \\ n \end{bmatrix} = V^T\begin{bmatrix} u \\ v \end{bmatrix} \end{aligned} \tag{17}$$

$$SSE = (V')\Lambda(V')^T = \lambda_1 m^2 + \lambda_2 n^2 \geq min(\lambda_1, \lambda_2)(m^2+n^2) = min(\lambda_1, \lambda_2)(u^2+v^2)$$

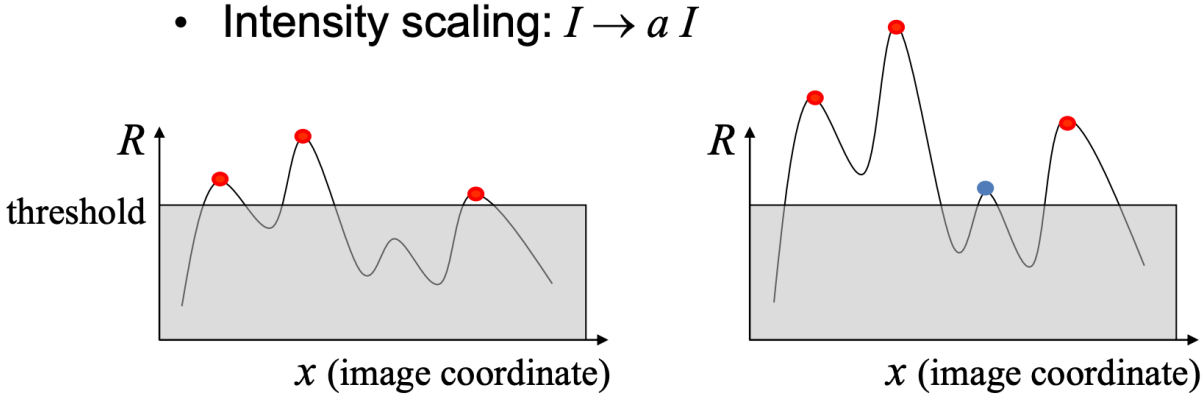


    - larger λ_1, λ_2 => larger $min(\lambda_1, \lambda_2)$ => large SSD
    - Metrics
        - $R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$ (large for corner, negative for edge, small for flat region)
        - $R = \frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2} = \frac{1}{\frac{1}{\lambda_1}+\frac{1}{\lambda_2}}$
        - $R = min(\lambda_1, \lambda_2)$
- Pipeline
    - compute the gradient at each point in the image (need Gaussian)
    - create the H matrix from the entries in the gradient
    - compute the eigenvalues
    - find points with large response ($\lambda_{min} > threshold$)
    - choose those points where λ_{min} is a local maximum as features
    - Hyperparameters: not include (u,v), include sigma for Gaussian, window size, and threshold
- Weighting the derivatives: weight the harris matrix based on the distance from the center pixel

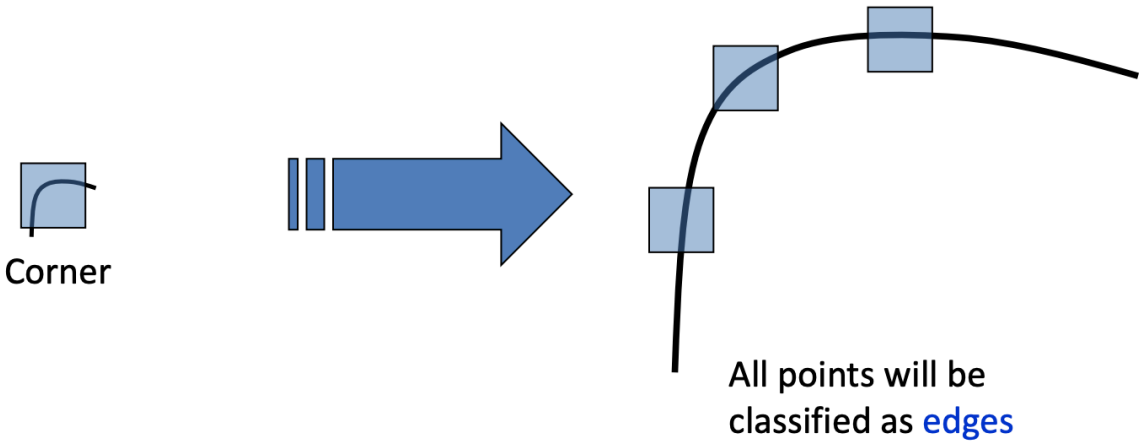
$$H = \sum_{(x,y) \in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- property (invariance?)
  - Translation: okay
  - rotation: okay
  - affine intensity scaling: partially okay

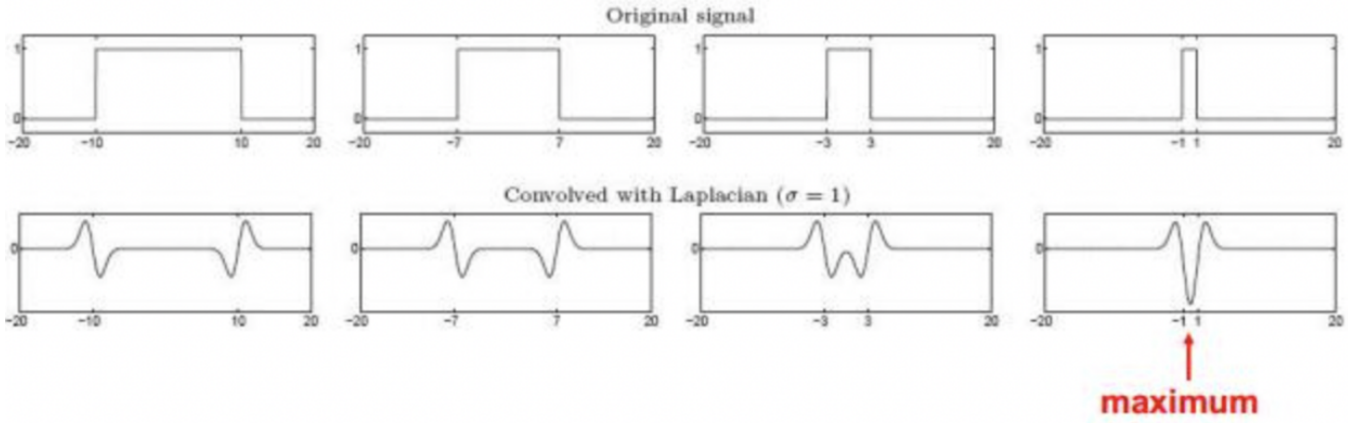
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\, I$



  - Scale: not okay => try in Gaussian Pyramid
    - Find the local maximum of the Harris eigenvalue pairs for different positions and scales
    - Use Gaussian Pyramid, no need to change the window size



5.2 Blob detection



- Find the maximum and minimum of LoG operator in space and scale
- Use different σ in Gaussian filter to get different scale

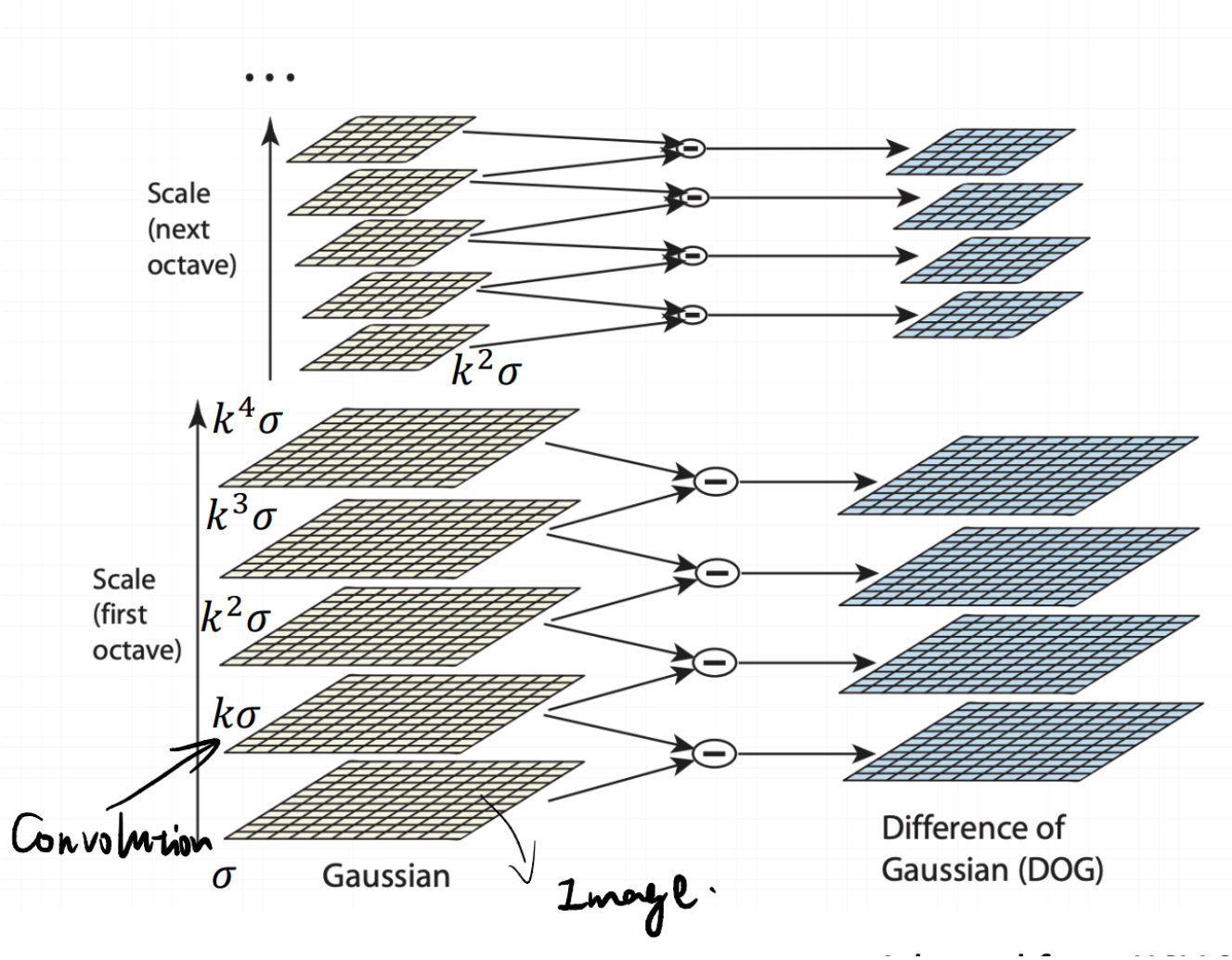
6 Scale Invariance Feature Transform (Important)

Pipeline:

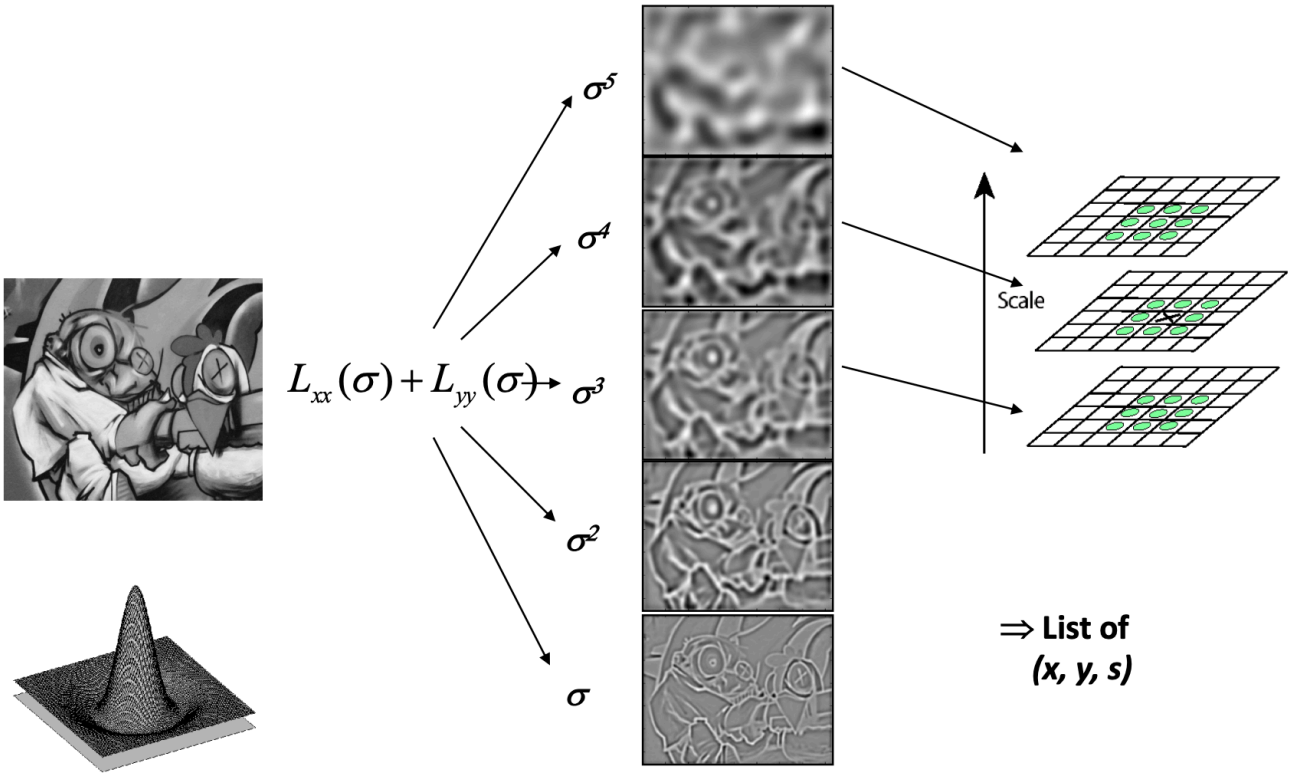
- Scale space peak selection (potential locations): LoG or Harris
- Key point localization: remove some point that are not interesting
- Orientation Assignment: Relative gradient histogram
- Key point descriptor: 128 dimension vector

6.1 Building scale-space & Interest Point Detection

- LoG can be approximated by DoG(difference of Gaussian)



- Find the local extreme value of the neigborhood scale and value (8+9+9 = 26)



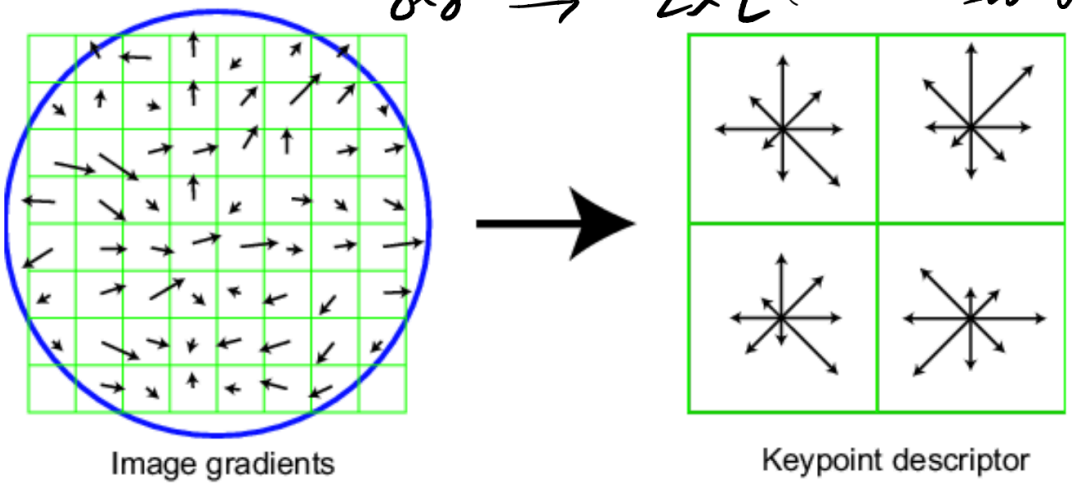
6.2 Orientation Assignment

- An orientation histogram includes **36 bins** (in 360 degrees) is formed for sample points **within a region around the key point**
- The samples added to the histogram is weighted by the **gradient magnitude**
- **dominate direction**: peak in the histogram

6.3 SIFT feature descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram (8 bin) for each cell (relative orientation and magnitude)
- 16 cells * 8 orientations = 128 dimensional descriptor



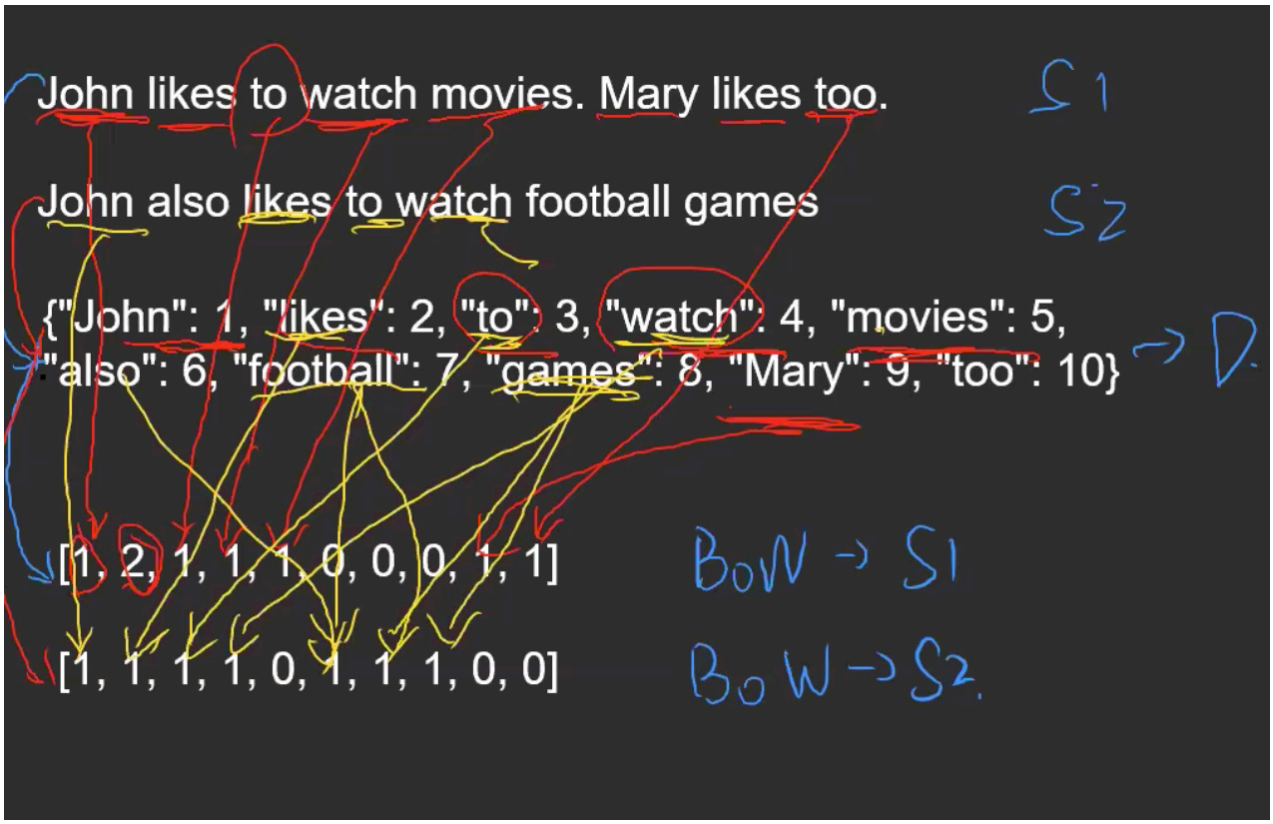
6.4 SIFT distance calculator

- May rotate the patch first to align the dominate direction
- Simple approach: L2 distance $argmin_{f_2}||f_1 - f_2||_2^2$ (may have ambiguous matches)

- Ratio distance: ratio distance $\frac{||f_1-f_2||_2^2}{||f_1-f_2'||_2^2}$
  - large values for ambiguous matches (1 is the largest value)
  - **drop the match if the ratio distance has large value, cannot change the best match**

7 Bag of Words

- Brute force image matching vs BoW
  - Brute force: many pairs of images for large dataset => high computational cost
  - BoW: only consider likely matches by using fast global similarity measures (concept of histogram)



    - a histogram of local feature vectors in an image
    - reduce the computational complexity

7.1 The BoW representation

- Pipeline
  - Extract features
  - Learn “visual vocabulary” (clustering)
  - Quantize features using visual vocabulary
  - Represent images by frequencies of “visual words” (get the histogram for the images)
- Extract features: e.g. use SIFT to extract n 128 dimension vectors
- Visual vocabulary learning: K-means to cluster the feature vectors

$$D(X,M)=\sum_{cluster\ k}\ \sum_{i\ in\ cluster\ k}(x_i-m_k)^2 \tag{18}$$

  - randomly initialize K cluster centers
  - interate until convergence (minimize the total distance)

        - Assign each data point to the nearest center
        - Recompute each cluster centre as the mean of all points assigned to it
- Quantize features using visual vocabulary: map the image to the codebook (dictionary) to get the histogram
    - extract 128-D vectors from the image
    - Compare each vector of the image with the codebook vectors and find the minimum distance
- Spatial Pyramid: divide the image into different region, and compute BoW histogram for each region

7.2 TF-IDF weighting

- Weight: weight of each word could be adjusted (some visual words are more discriminative than others) e.g. the, and, or have lower weight
- TF-IDF (term frequency - inverse document frequency)

$$
\begin{aligned}
IDF(word\ j) = log(\frac{number\ of\ documents}{number\ of\ documents\ in\ which\ j\ appears}) \\
Weight(word\ j) = TF(j) * IDF(j)
\end{aligned} \tag{19}
$$

    - Instead of computing a regular histogram distance, we'll weight each word by it's inverse document frequency (increase the weight for unique words)

7.3 Inverted File

- Sparse histogram: the image may not include most of features
- Mapping from words to the document: record a list of documents (images) for each word (feature)
- Can quickly use the inverted file to compute similarity between a new image and all the images in the database
    - Only consider database images whose bins overlap the query image

8 Transformation and Alignment

8.1 Image Warping

- Image filtering changes the range of the image, while image warping changes the domain of the image.
- Forward Warping: compute the location first and then assign the pixel value

$$
\begin{aligned}
(x', y') = T(x, y) \\
g(x', y') := f(x, y)
\end{aligned} \tag{20}
$$

    - What if pixel lands between two pixels? => add contribution to several pixels, normalize later
    - may have holes
- Inverse Warping (no holes): for each pixel in the target image, find the pixel value in the original image.

$$
\begin{aligned}
(x, y) = T^{-1}(x', y') \\
g(x', y') := f(x, y)
\end{aligned} \tag{21}
$$

    - require taking the inverse of the transform
    - what if original pixel is between two pixels? => resample color value from interpolated (pre-filtered) source image
    - No holes
- Parametric (global) Warping: a global transform (same for each pixel) matrix

$$
\begin{aligned}
p' &= T(p) \\
e.g.\ linear\ transform &: p' = Tp
\end{aligned}
\tag{22}
$$

8.2 All 2D Linear Transformations:

- including **scale, rotation, shear and mirror**
- Can be represented by a 2×2 matrix

8.3 Homogeneous Coordinates

- Represented by 3×3 matrix
- Add one more coordinate to compute the translation, each points in the homogeneous coordinates can be projected to the homogeneous plane ($z = 1$) to get the 2D coordinate.

$$
\begin{aligned}
(x, y) &=> (x, y, 1) \\
(x, y, w) &=> (x/w, y/w, 1)
\end{aligned}
\tag{23}
$$

- Translation

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

8.4 Affine Transformations

- $T^{(3)} = [0, 0, 1]$ (the last dimension is always 1 after transformation)
- Including linear transformations and translations

8.5 Homography:

- $T^{(3)} = [g, h, 1]$ (we can always divide the other eight parameters by $T_3^{(3)}$
- Use the 3D matrix two rotate the homography plane

8.6 Image Alignment

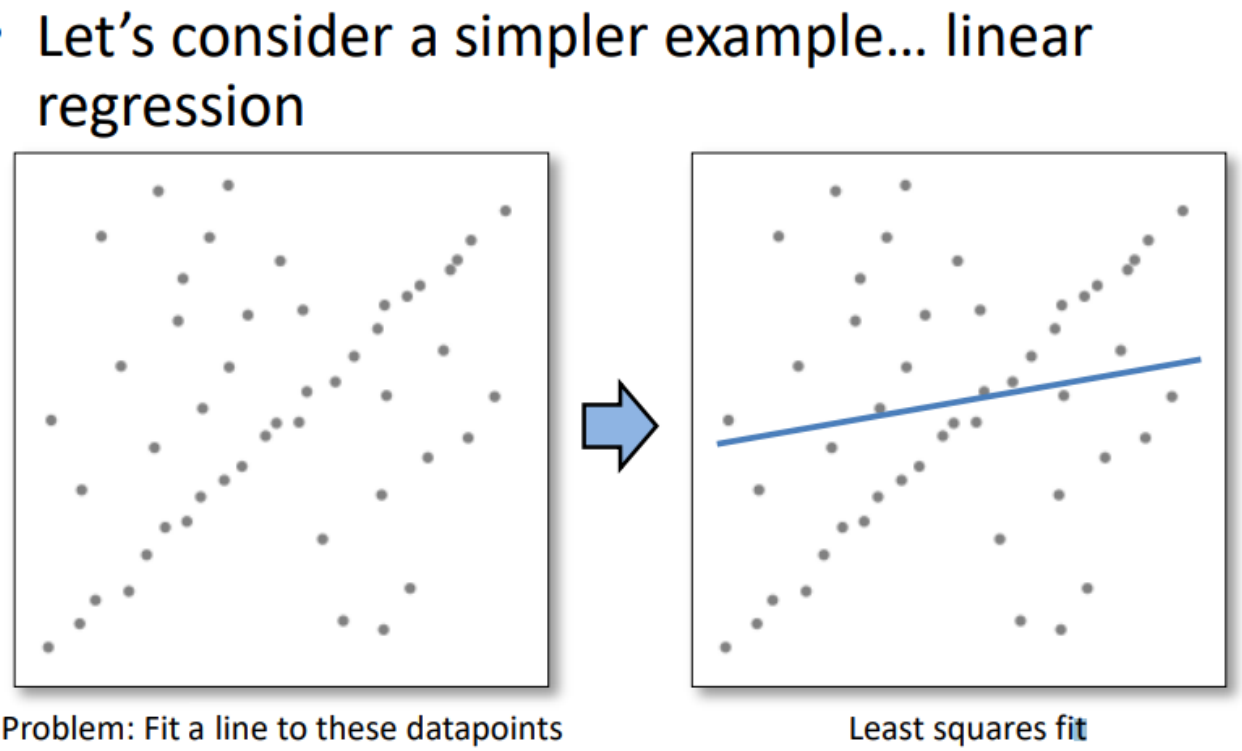
- Simple case: translation match 2 equations per match, 2 unknowns => one match needed

$$
(x_t, y_t) = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i' - x_i), \frac{1}{n} \sum_{i=1}^{n} (y_i' - y_i) \right)
\tag{24}
$$

- Affine match: 2 equations per match, 6 unknowns => 3 matches needed
- Homography match: 2 equations per match, 8 unknowns => 4 matches needed

8.7 RANSAC (no calculation)

- Problem of least square: outliers will affect the accuracy

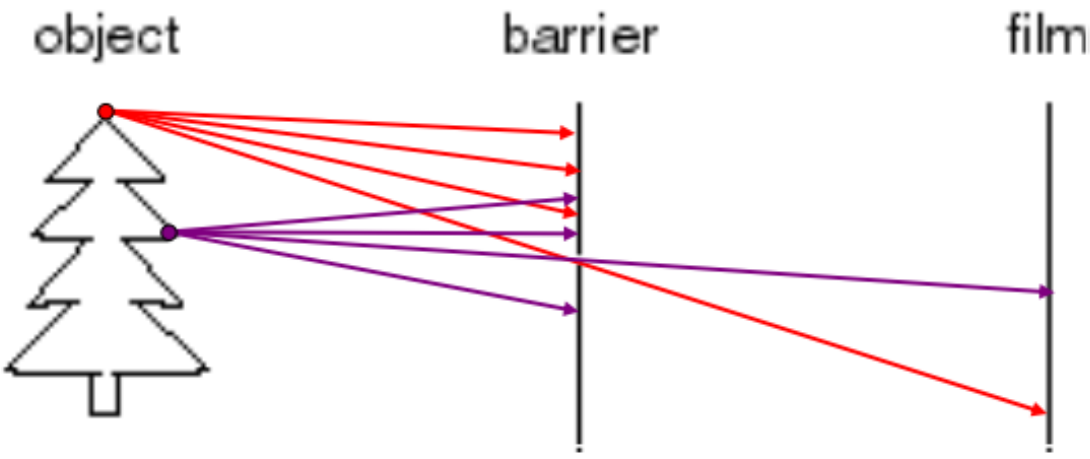


- Method: RANdom SAmple Consensus (RANSAC)
  - Randomly choose s examples
  - fit a model (given a hypothesized line)
  - count the number of inliers => distance with a threshold ϵ
  - repeat n times
  - choose the model with the most number of inliers
  - After finding the vector with the most number of inliers, can also take the average of inliers

9 Camera

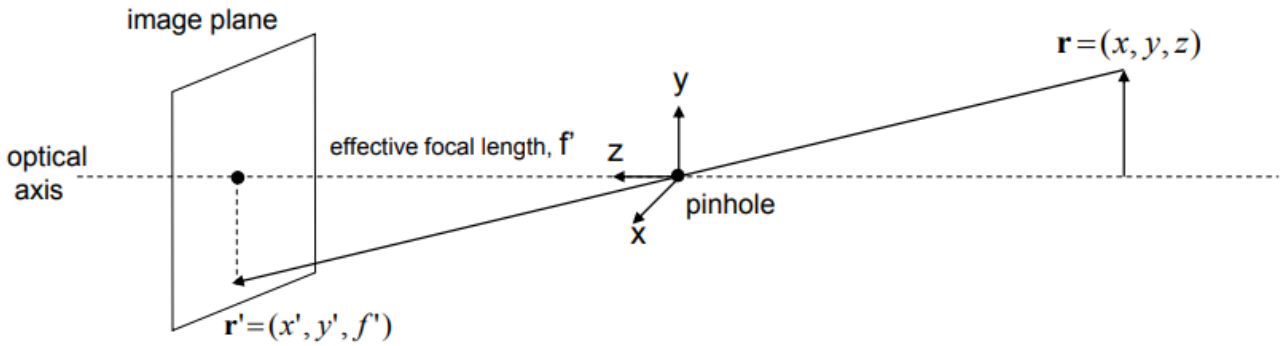
9.1 Pinhole camera

- Add a barrier to block off most of the ray



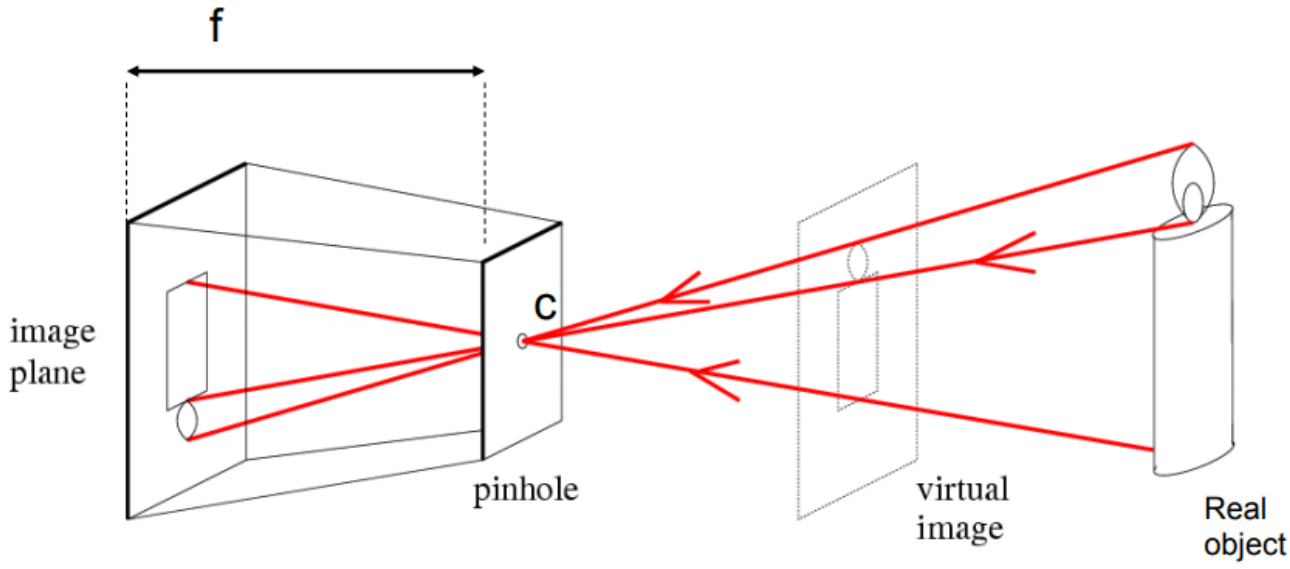
- reduce blurring
- opening: **aperture**

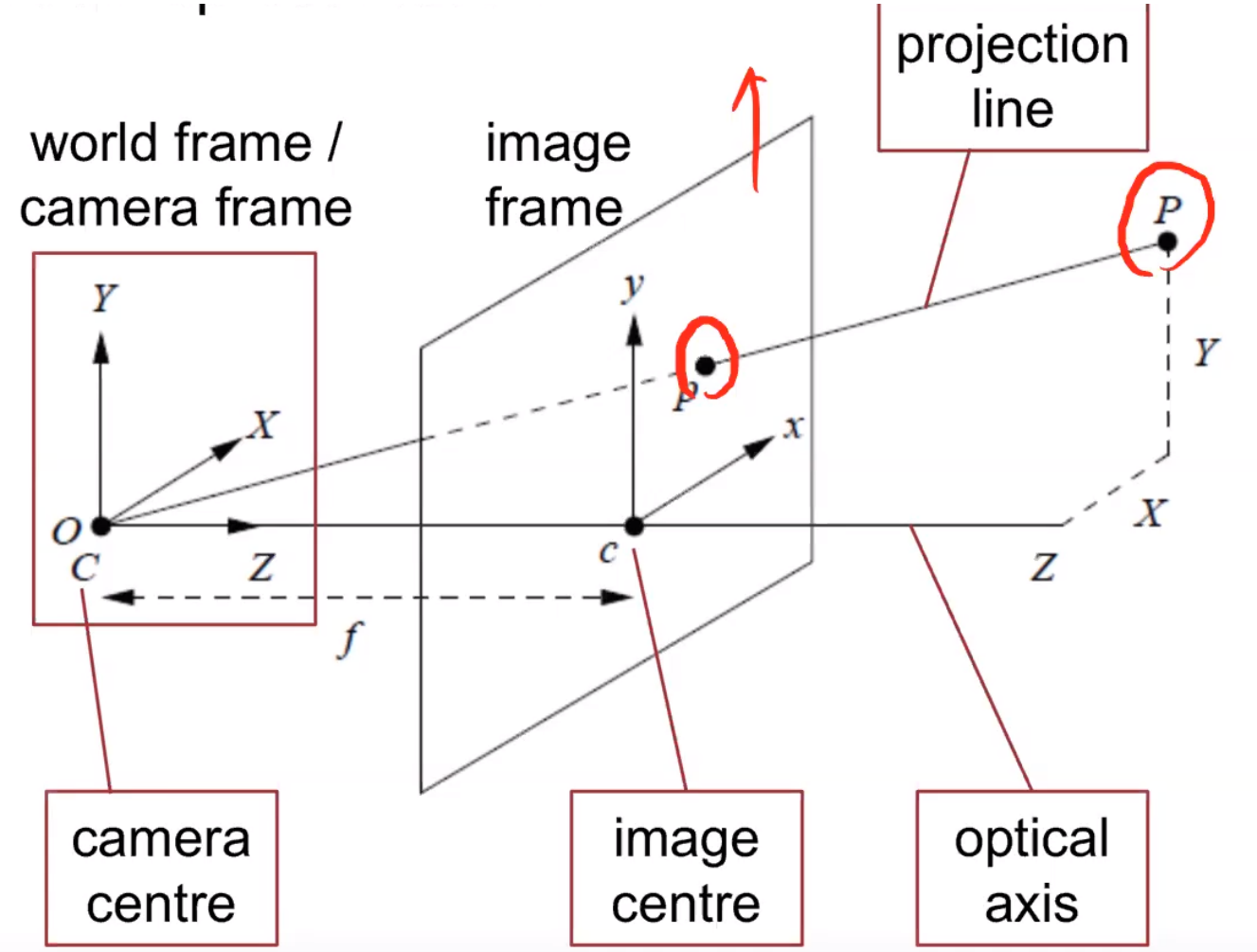
- Perspective Projection



$$\frac{\mathbf{r}'}{f'}=\frac{\mathbf{r}}{z} \quad \Rightarrow \quad \frac{x'}{f'}=\frac{x}{z} \qquad \frac{y'}{f'}=\frac{y}{z}$$

- Pinhole Camera Model





- f: focal length
- c: Optical center of the camera
- Task: given a point of real object, find the corresponding point in the virtual image.

$$
\begin{aligned}
(x', y') &<= (x, y, z) \\
\frac{x}{z} = \frac{x'}{f}, \frac{y}{z} &= \frac{y'}{f} \\
P' = (x', y') &= (f\frac{x}{z}, f\frac{y}{z})
\end{aligned} \tag{25}
$$

9.2 Camera parameters

- world vs. camera coordinate
  - Project a point in world coordinate into the camera coordinate (3D => 3D)
  - calculate the point on the virtual image plane (3D => 2D)
- intrinsic (fixed for each cameras)

$$
\begin{aligned}
P &= (x, y, z)_{camera\ coordinate} \\
P' = (x', y') &= (fk\frac{x}{z} + c_x, fl\frac{y}{z} + c_y) \\
&= (\alpha\frac{x}{z} + c_x, \beta\frac{y}{z} + c_y) \\
P' &= \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = MP \\
P' = MP &= \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} [I \quad 0]P = K[I \quad 0]P
\end{aligned} \tag{26}
$$

  - translation: denote the location of the principle point c as (c_x, c_y)
  - Scale: digital image (pixel) and real world (centimeter) ratio ($\frac{pixel}{cm}$) in two dimensions k, l. For square pixels $k = l$
  - K: Camera Matrix (or calibration matrix) => why use 3×4 matrix? (to match the shape of extrinsic parameters, the extrinsic matrix includes 3D translation, therefore we need to add a homogeneous plane)
- extrinsic (change the position of camera=> change)

$$
P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w \tag{27}
$$

  - Camera position & orientation => rotation + translation matrix (4×4)

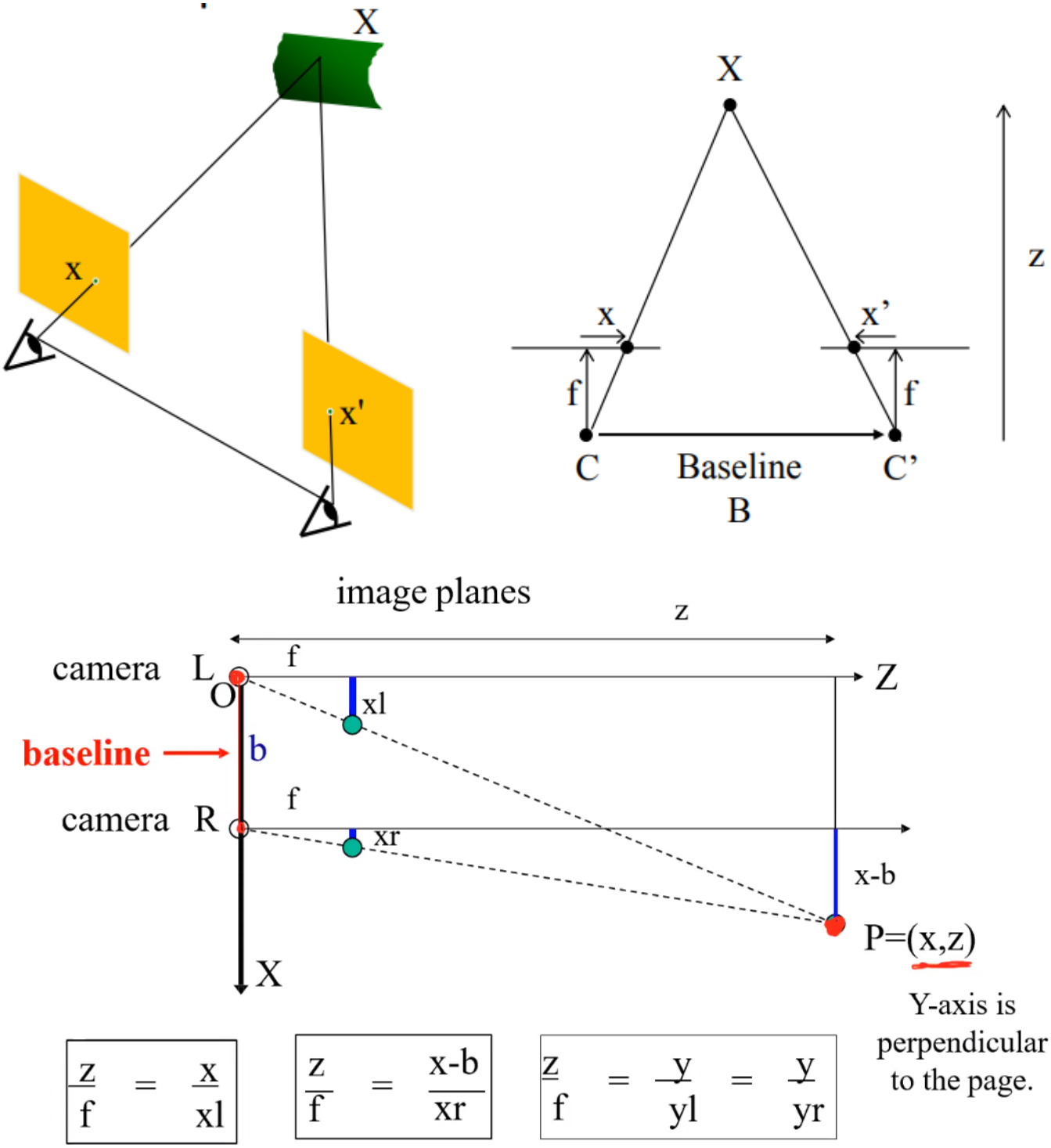
9.3 Modeling projection

$$
\begin{aligned}
P' = MP = K[I \quad 0]P &= K[I \quad 0] \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w \\
&= K[R \quad T]P_w
\end{aligned} \tag{28}
$$

- Use homogeneous coordinates for camera and world coordinates $(x, y, z) => (x, y, z, 1)$
- Estimation of intrinsic and extrinsic parameters: camera calibration

10 Stereo Vision and Structure from Motion

10.1 Depth and Disparity



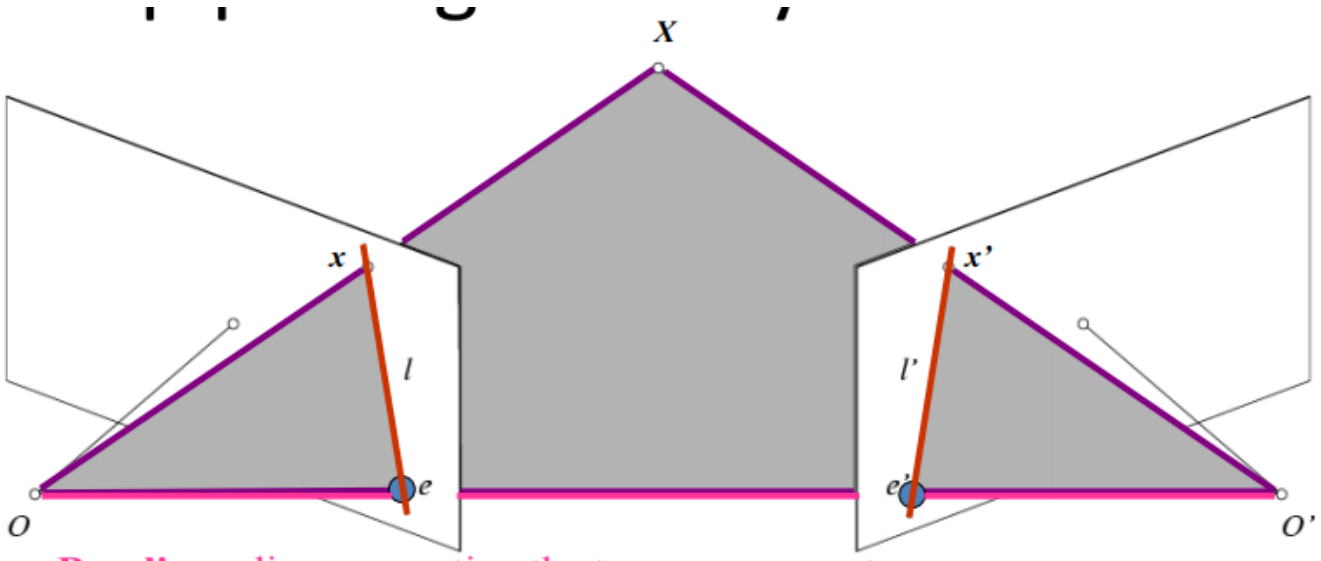
$$\frac{z}{f} = \frac{x}{xl} \qquad \frac{z}{f} = \frac{x\text{-}b}{xr} \qquad \frac{z}{f} = \frac{y}{yl} = \frac{y}{yr}$$

$$x_l = \frac{xf}{z}, x_r = \frac{(x-b)f}{z}$$
$$x_l - x_r = \frac{bf}{z} \tag{29}$$
$$z = \frac{bf}{x_l - x_r}$$

- Depth (z): distance to the camera
- Disparity ($x_l - x_r$): horizontal shift of two cameras (the pixel shift in two image planes)
- Depth is inversely proportional to the disparity
- Goal: recover the depth using the disparity (also should know the baseline and focal length)

- Calibration: Recover the relation (baseline) of the cameras (translation)
- Correspondence: search for the matching point x_r for x_l

10.2 Epipolar Geometry



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

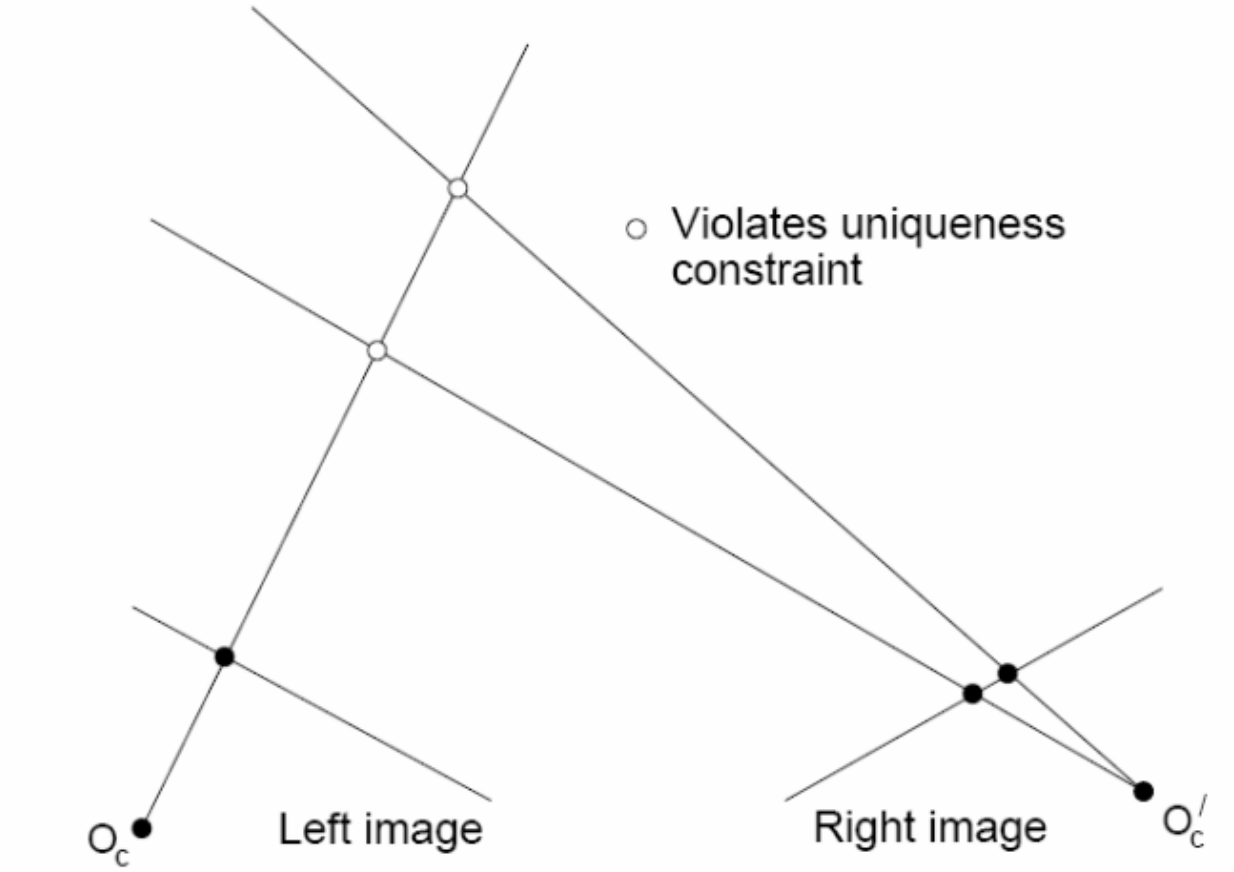
12

- Reduce searching scale to the **Epipolar lines** for correspondence problem

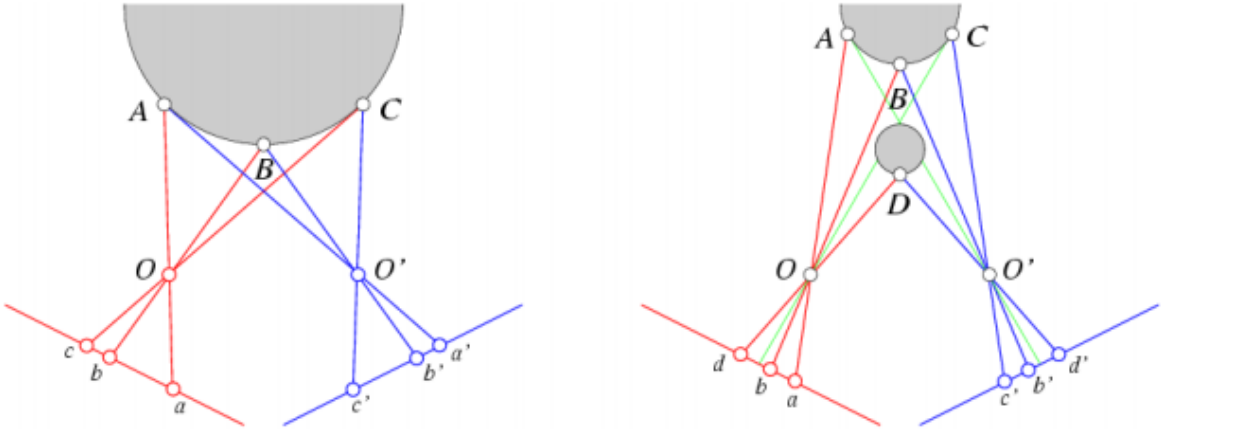
10.3 Stereo Matching

- Calculate the disparity
  - Rectify the two stereo images to transform Epipolar lines into scanlines
  - For each pixel x_l in the left image
    - find corresponding epipolar scanline
    - search and pick the best match x_r (compare the patch in the neighborhood)
    - compute the disparity $x_l - x_r$ and the depth
- Stereo Image Rectification
  - Make the camera in parallel => epipolar lines can be represented as horizontal lines
  - Use the homography
- Matching Cost
  - Slide a window along the right scanline and compare contents of that window with the reference window in the left image
  - Matching cost: SSD, SAD, or normalized cross correlation
  - Window size:

        - Smaller: more detail but more noise
        - Larger: Smoother disparity maps but less detail and fails near boundaries
- Stereo Constraints /Priors
    - **Uniqueness**: For any point in one image, there should be at most one matching point in the other image



    - **Ordering**: Corresponding points should be in the same order in both views (not always true, but for most cases)



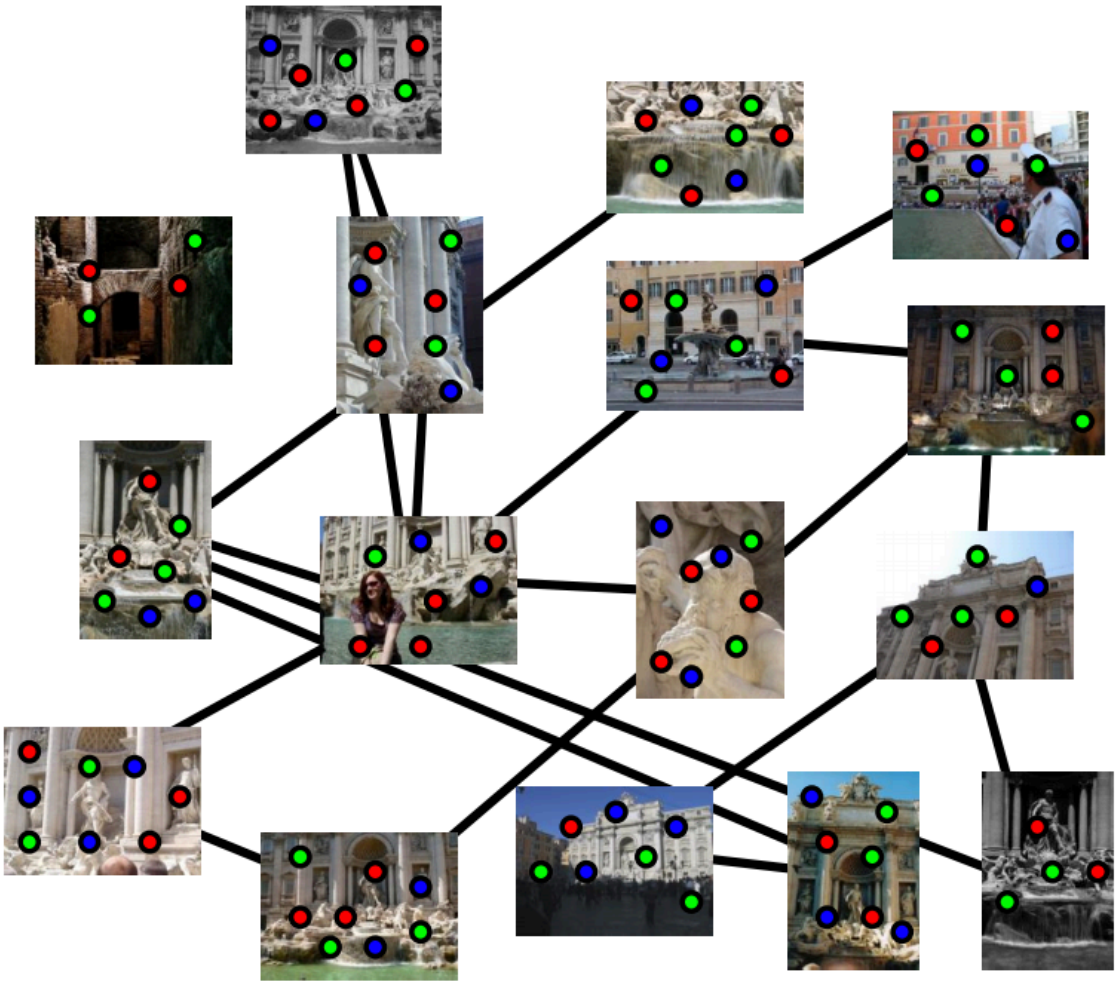
Ordering constraint doesn't hold

    - **Smoothness**: We expect disparity values to change slowly => disparity change for adjacent pixels are the similar
- Pipeline
    - Steps

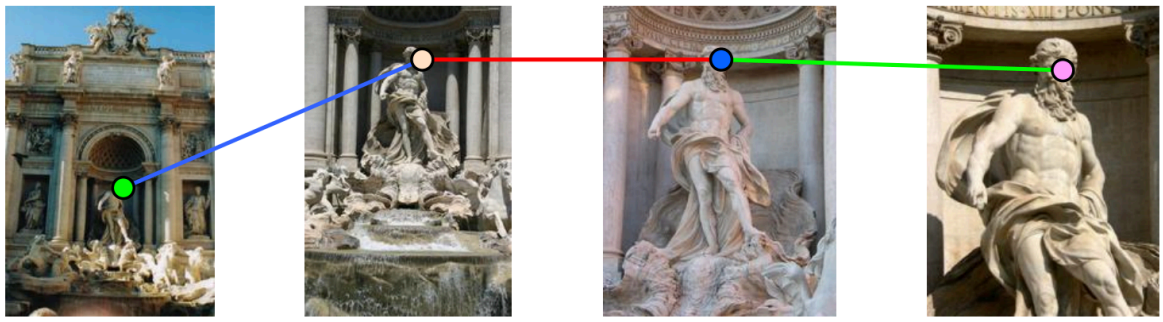
        - calibrate cameras
        - rectify images
        - compute disparity
        - estimate depth
    - Error cases=>use more than two images may improve the performance
        - camera calibration errors
        - poor image resolution
        - occlusions
        - violations of brightness constancy
        - low-contrast image regions

10.4 Structure from motion: problem definition

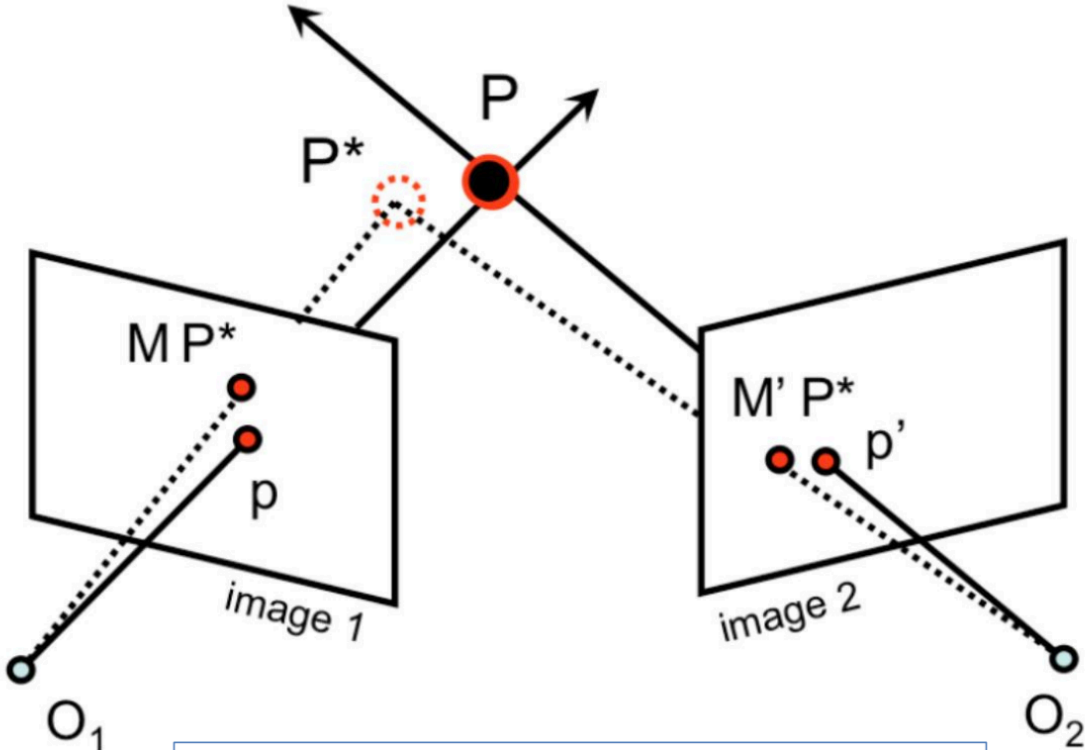
- Problem
    - Images => 3D point cloud
    - Figure out camera position
    - Build a 3D model of the scene
- Applications
    - object recognition
    - Robotics
    - Computer Graphics
    - Image Retrieval
    - Localization
- The Algorithm (i labels point, j labels camera)
    - Input: Image with points in correspondence $p_{i,j} = (u_{i,j}, v_{i,j})$
    - Output
        - Structure: 3D location $\mathbf{x_i}$ for each point p_i
        - Motion: camera parameters R_j, t_j
    - Objective function: minimize reprojection error
    - Steps
        - Feature detection (SIFT) and matching (bewteen each pair of images)



- Correspondence estimation: Link up pairwise matches to form connected components of matches across several images



- Minimize re-projection error: bundle adjustment (find the R T X)



$$\begin{aligned}\mathbf{R},\mathbf{T},\hat{\mathbf{X}} &= argmin_{\mathbf{R},\mathbf{T},\hat{\mathbf{X}}}\{g(\mathbf{R},\mathbf{T},\hat{\mathbf{X}})\} \\ &= argmin_{\mathbf{R},\mathbf{T},\hat{\mathbf{X}}}\{||M\hat{X}-p||^2+||M'\hat{X}-p'||^2\end{aligned} \tag{30}$$

11 Optical Flow

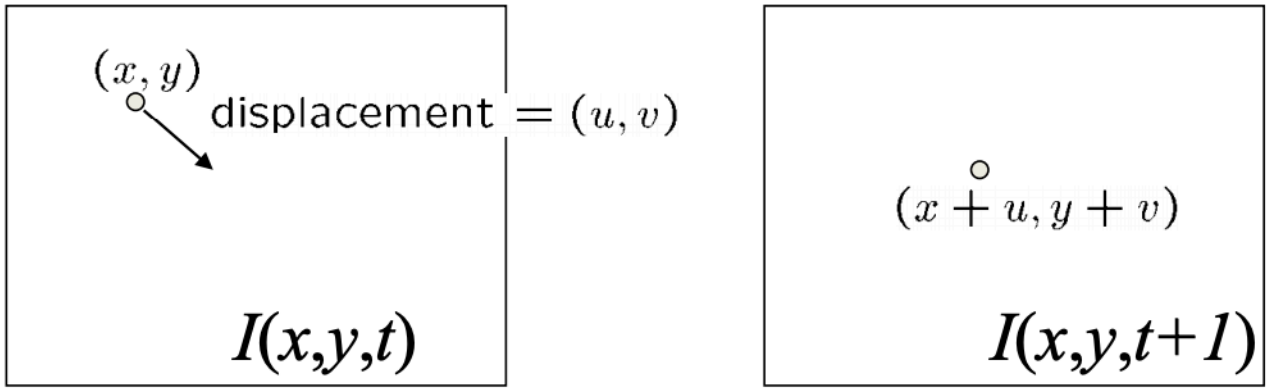
11.1 Video & Optical Flow

- Image vs. video
  - Image: pixels * channels
  - Video: pixels * channel * temporal
- Motion
  - Cause of motion (different combination of light, object, camera condition)
    - Static camera, moving objects (surveillance)
    - Moving camera, static scene (3D capture)
    - Moving camera, moving scene (sports, movie)
    - Static camera, moving objects, moving light (time lapse)
  - Recovering motion:
    - Feature tracking: extract visual features (corners, texture areas) and “track” them over multiple frames
      - Figure out which features can be tracked
      - Efficiently track across frames
      - Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
      - Drift: small errors can accumulate as appearance model is updated
      - Points may appear or disappear: need to be able to add/delete tracked points
    - Optical flow: recover image motion at each **pixel** from patio-temporal image vrightness variations (optical flow)

- Definition: a velocity field in the image which transforms one image into the next image in a sequence
- Ambiguity: two flow may have the same result

11.2 Assumptions in Lucas-Kanade method

- Task: given two subsequent frames, estimate the point translation



- Assumptions
  - **Brightness Constancy**: projection of the same point looks the same in every frame
  - **Small Motion**: points do not move very far (alows first order Taylor expansion)
  - **Spatial Coherence**: points move like their neighbors

11.3 Lucas-Kanade Algorithm (brightness Constancy Equation)

$$
\begin{aligned}
I(x,y,t) &= I(x+u, y+v, t+1) \\
I(x+u, y+v, t+1) &\approx I(x,y,t) + \frac{dI}{dx}u + \frac{dI}{dy}v + \frac{dI}{dt} \\
I_x u + I_y v + I_t &\approx I(x+u, y+v, t+1) - I(x,y,t) = 0 \\
\nabla I[u,v]^T + I_t &= 0 \; or \; I_x u + I_y v = -I_t
\end{aligned}
\tag{31}
$$

- Brightness constancy equation
  - Take the first order Taylor expansion of $I(x+u, y+v, t+1)$ at (x,y,t) to linearize the right side
  - One equation two unknowns per pixel => multiple result $I_x u + I_y v + I_t = 0$ represent a line in $u-v$ plane. This can be shown as the aperture problem visually (multiple motion have the same visual effect).
  - Aperture problem is significant for flat regions and lines, but not for corners. (important information for feature tracking, tells us which pixels are easy to track)
- Spatial coherent assumption: neighbours have the same motion

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$Ad = b => d = (A^T A)^{-1} A^T b \tag{32}$$

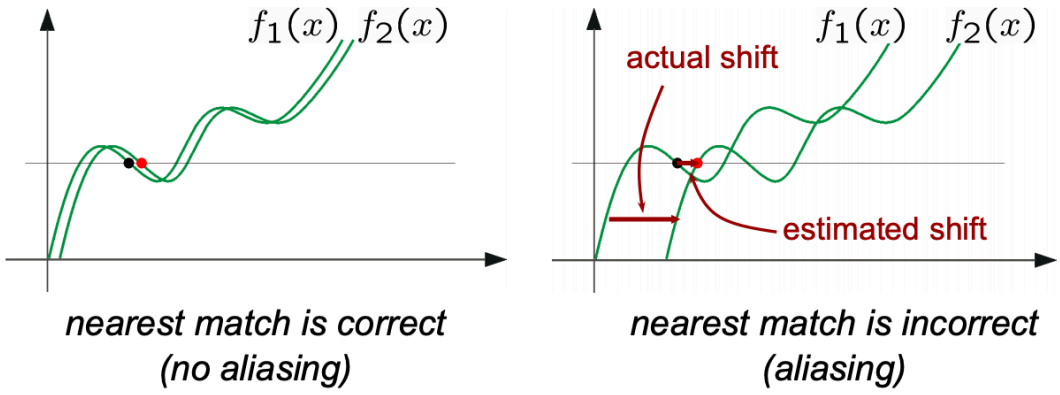
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

- If we use 5×5 window, that gives us 25 equations per pixel

- Errors in Lukas-Kanade
  - $A^T A$ is not easily invertible
  - noise in the image
  - one of the three assumptions is not satisfied
    - variate brightness
    - large motion: Newton's method and coarse-to-fine estimation
      - First order Taylor expansion is not accurate
      - Aliasing: images can have many pixels with the same pixel value

I.e., how do we know which 'correspondence' is correct?

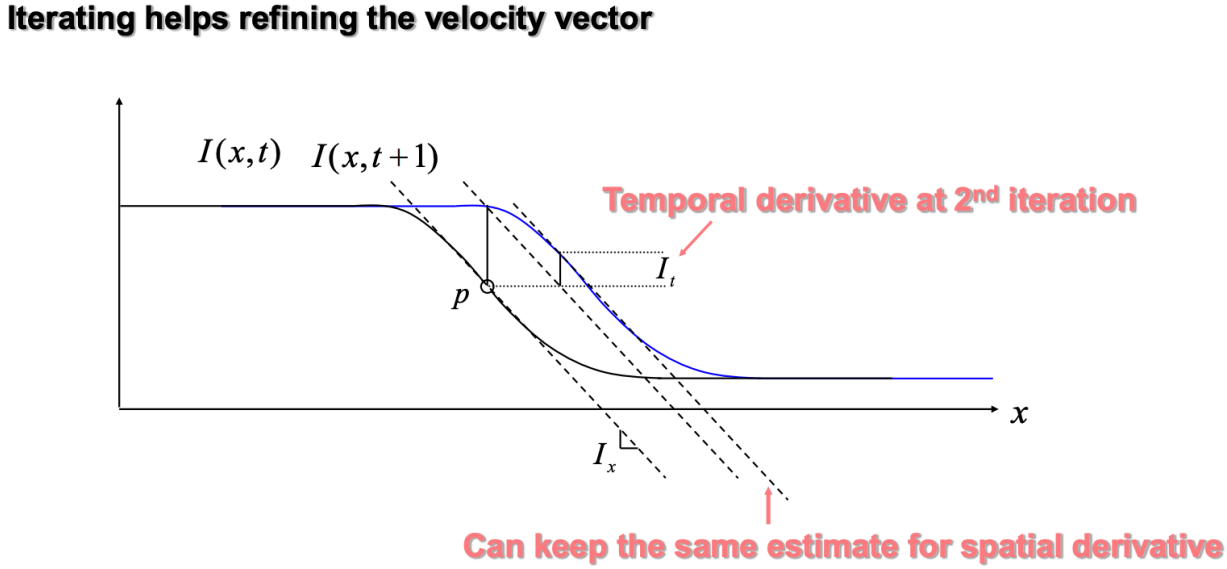


nearest match is correct (no aliasing)

nearest match is incorrect (aliasing)

- diverse trace: should find suitable window size

- Interative Refinement
  - 1D example

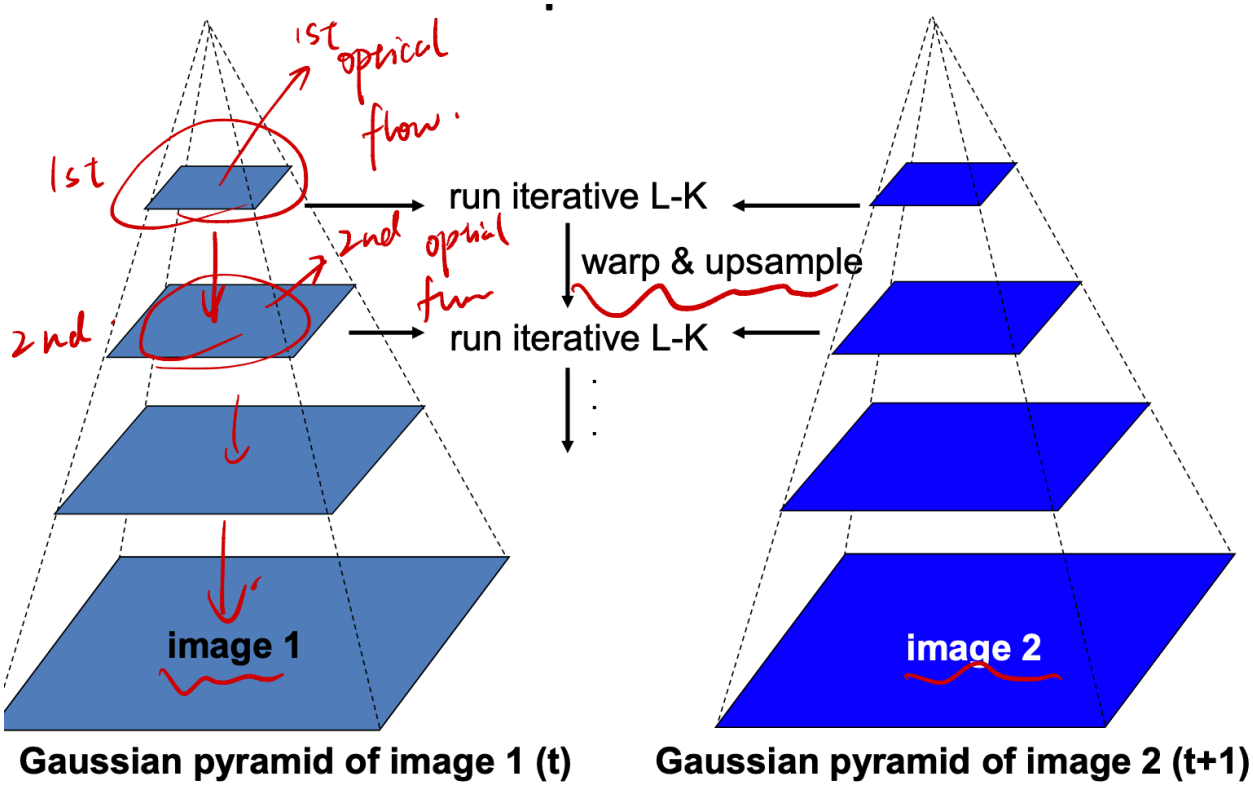


$$\vec{v} \leftarrow \vec{v}_{previous} - \frac{I_t}{I_x}$$

Converges in about 5 iterations 34

$$
\begin{aligned}
I_t \approx \frac{I(x,t+1) - I(x,t)}{t+1-t} = I(x,t+1) - I(x,t) \\
I_x = \frac{dI}{dx} = slope \\
I_x u + I_t = 0 \rightarrow u = -\frac{I_t}{I_x} \\
I'(x,t) := I(x-u,t)
\end{aligned}
\tag{33}
$$

  - Method
    - estimate velocity at each pixel by solving Lucas-Kanade equations
    - Warp $I(t-1)$ towards $I(t)$ using the estimated flow field
    - repeat util convergence
- Coarse-to-fine flow estimation
  - reduce the resolution to modify the motion (low resolution image has lower pixel shift for the same shift in real world)
  - Gaussian Pyramid



- Add a upsample (according to the scale) procedure to the flow field to the next level, and use it as the initial flow guess (e.g. 1.25 pixels => 2.5 pixels => 5 pixels)