Advanced Computer Vision (Spring 2023)

Instructor: Chiou-Shann Fuh

Created by David Wang

Last updated on June 06, 2023

Textbook:

Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Nature, 2022.

Contents

(Note that the following section number is based on the above textbook)

Part B: Optimization and Deep Learning

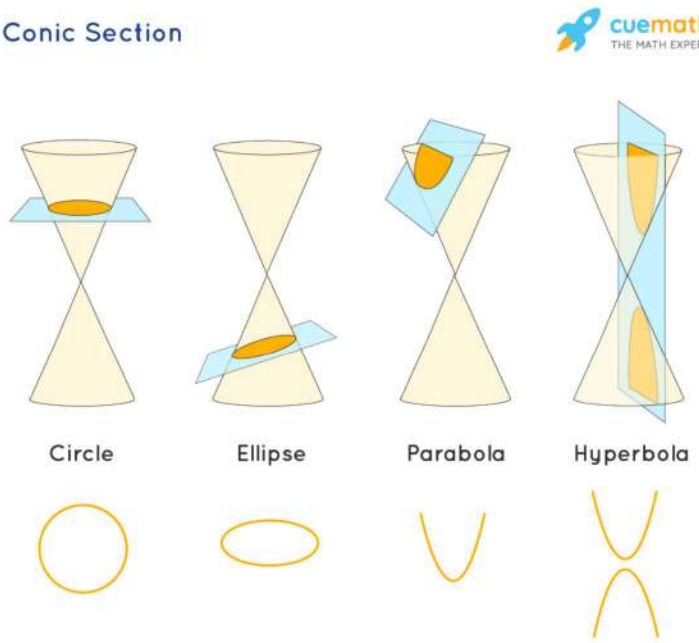
Part C: World Geometry and 3D

Chapter 2 Image formation

2.1 Geometric primitives and transformations

2.1.1 2D transformations

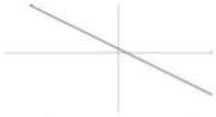
2D conics: ellipse (including circle), parabola, hyperbola



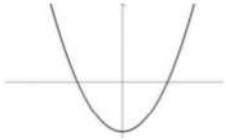
4 basic 2D Curves: line, ellipse (including circle), parabola, hyperbola

A 2D curve review

Lines: $ax + by = c$



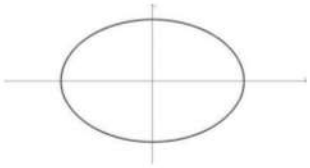
Parabolas: $ax^2 + by = c$ or
$ax + by^2 = c$



Ellipse: $ax^2 + by^2 = c$ (if $a,b,c > 0$)

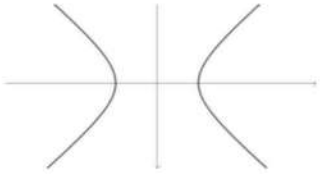
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

(*Note:* If a = b, then it's a circle)



Hyperbola: $ax^2 - by^2 = c$ or
$-ax^2 + by^2 = c$ (if $a,b,c > 0$)

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$



7 basic 3D shapes:

Cylinders, Cones, Ellipsoids (including spheres), Paraboloids*2, Hyperboloids*2

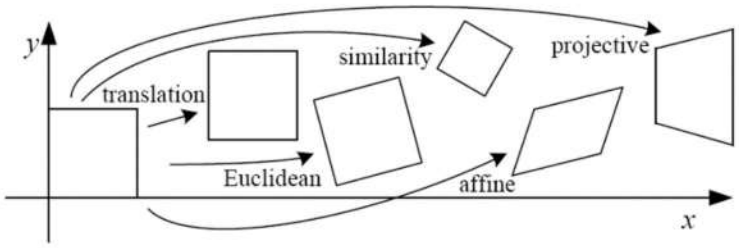


Figure 2.4 Basic set of 2D planar transformations.

Preserves	Translation	Euclidean	Similarity	Affine	Projective
Lengths	O	O	X	X	X
Angles	O	O	O	X	X
Parallelism	O	O	O	O	X

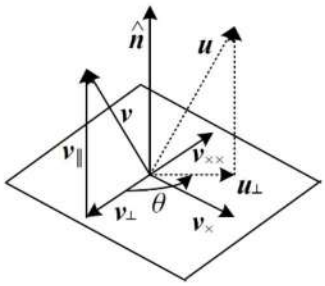
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]_{2\times3}$	2	orientation	
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	3	lengths	
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]_{2\times3}$	4	angles	
affine	$\left[\, \boldsymbol{A} \,\right]_{2\times3}$	6	parallelism	
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{3\times3}$	8	straight lines	

2.1.2 3D transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]_{3\times4}$	3	orientation	
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]_{3\times4}$	6	lengths	
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]_{3\times4}$	7	angles	
affine	$\left[\, \boldsymbol{A} \,\right]_{3\times4}$	12	parallelism	
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{4\times4}$	15	straight lines	

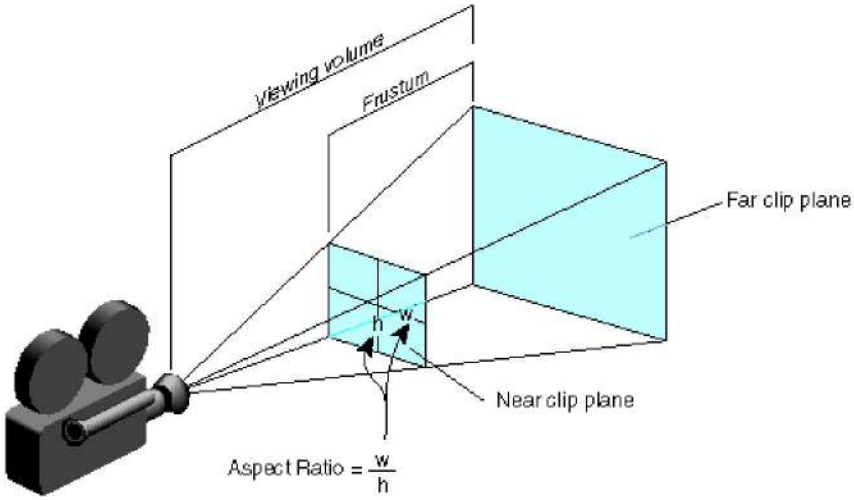
2.1.3 3D rotations

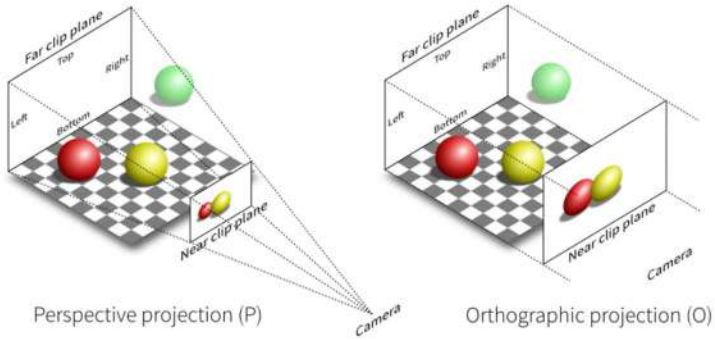
- Project the vector $\boldsymbol{v}$ onto the axis $\hat{\boldsymbol{n}}$:
  - $\boldsymbol{v}_{\parallel} = \hat{\boldsymbol{n}}(\hat{\boldsymbol{n}} \cdot \boldsymbol{v}) = (\hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T)\boldsymbol{v}$
- Compute the perpendicular residual of $\boldsymbol{v}$ from $\hat{\boldsymbol{n}}$:
  - $\boldsymbol{v}_{\perp} = \boldsymbol{v} - \boldsymbol{v}_{\parallel} = (\boldsymbol{I} - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T)\boldsymbol{v}$

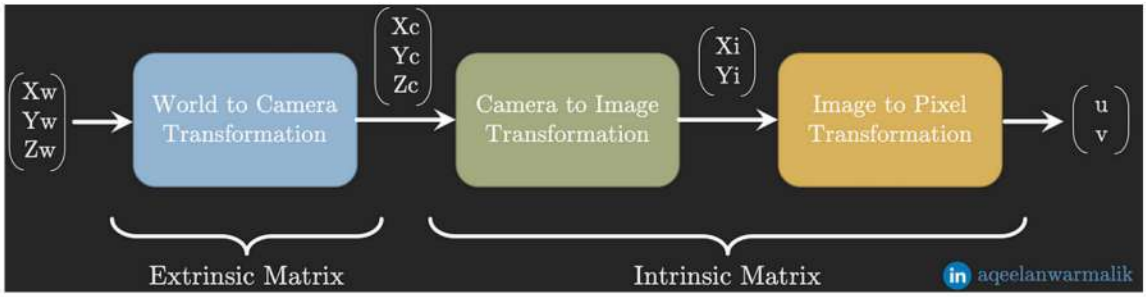


2.1.4 3D to 2D projections

- In the **perspective** view, objects which are far away are smaller than those nearby.
- In the **orthographic** view, all objects appear at the same scale.







- The commonly used coordinate systems in Computer Vision are
  - World coordinate system (3D)
  - Camera coordinate system (3D)
  - Image coordinate system (2D)
  - Pixel coordinate system (2D)
- The **extrinsic** matrix is a transformation matrix from the world coordinate system to the camera coordinate system, while the **intrinsic** matrix is a transformation matrix that converts points from the camera coordinate system to the pixel coordinate system.

$$\begin{pmatrix} X_i \\ Y_i \\ Z_C \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{pmatrix}$$

, and

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 1/\rho_u & 0 & c_x \\ 0 & 1/\rho_v & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_C \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f/\rho_u & 0 & c_x & 0 \\ 0 & f/\rho_v & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & -\cot(\theta) & 0 & 0 \\ 0 & 1/\sin(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{pmatrix}_{(4\times4)} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha & -\alpha\cot(\theta) & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{Camera Intrinsic Matrix}} \underbrace{\begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{pmatrix}_{(4\times4)}}_{\text{Camera Extrinsic Matrix}} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}$$

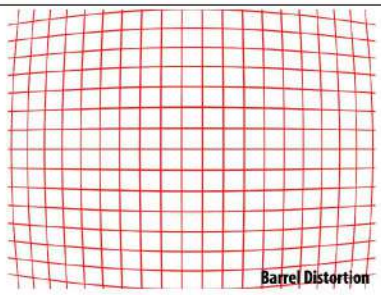
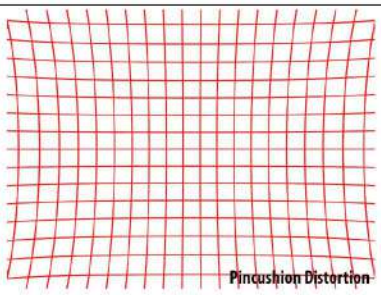
Mapping of a pinhole camera

P = K R [I | -C]

P = K [R | t] where t = -RC

2.1.5 Lens distortions

- In photography, (optical) **distortion** is generally referred to an optical aberration that deforms and bends physically straight lines and makes them appear curvy in image.

 Barrel distortion	 Pincushion distortion

- Simplest radial distortion model

$$\hat{x} = x(1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$\hat{y} = y(1 + \kappa_1 r^2 + \kappa_2 r^4)$$

where $r^2 = x^2 + y^2$ and κ_1, κ_2 are called the radial distortion parameters.

11.1 Geometric intrinsic calibration

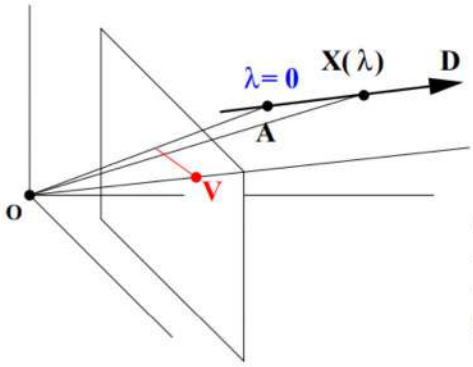
11.1.1 Vanishing points

- Camera projection matrix P
  - P is the matrix that maps the 3D world points in homogeneous coordinates into the (2D) homogeneous image coordinates.
  - $\tilde{x} = P\tilde{\mathrm{X}}$, where $\tilde{x} = [u, v, 1]^T$ and $\tilde{\mathrm{X}} = [x, y, z, 1]^T$.
- A (3D) point at infinity is represented as $x_\infty = [x_1, x_2, 0]^T$ or $\widetilde{\mathrm{X}_\infty} = [x_1, x_2, 0, 1]^T$.
  - A (3D) point at infinity = the point that parallel lines intersect at infinity
- Points at infinity form lines at infinity
  - Each pair of parallel lines intersects at a point at infinity $\{x_{\infty 1}, x_{\infty 2}, \dots, x_{\infty n}\}$. Then the line l_∞ that passes through these points at infinity satisfies ${l_\infty}^T x_{\infty i} = 0$, where $l_\infty = [0, 0, 1]^T$ and $i = 1, 2, \dots, n$.
- Consider a matrix H and a point at infinity $x=x_\infty$ in x'=Hx.
  - If H is a projective transformation (8 dof) $\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$, then x' is NOT a point at infinity unless x_3' is zero.
  - If H is an affine transformation (6 dof) $\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$, then $x_3 = 0$ and x' is a point at infinity.
- Vanishing Point:
  - Consider a camera projective matrix P and a point at infinity x_∞
  - In homogeneous coordinates, x_∞ is represented as $\widetilde{\mathrm{X}_\infty} = [x, y, 0, 1]^T$.
  - $\tilde{x} = P\tilde{\mathrm{X}}$, where $\tilde{\mathrm{X}} = \widetilde{\mathrm{X}_\infty}$, and $\tilde{x}$ is a point in the image plane.
  - Then, $\tilde{x} = v$ is a **vanishing point**.
  - $\tilde{x}$ is NOT a point at infinity since P is projective.
  - The **vanishing point** is the perspective image of the point where a bundle of parallel lines seems to converge.
- $v = Kd$
  - $d = [d_1, d_2, d_3]^T$ is the direction of a set of 3D parallel lines in the camera reference system.

All parallel lines meet at the same vanishing point 2.4

A line of 3D points isrepresented as

$$\mathbf{X}(\lambda) = \mathbf{A} + \lambda\mathbf{D}$$



Using $\mathbf{x} = f\mathbf{X}/Z$ the vanishing point of its image is

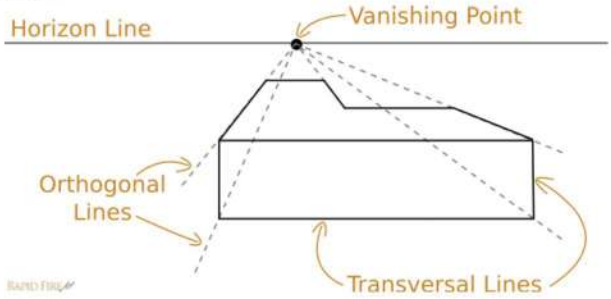
$$\mathbf{v} = \lim_{\lambda \to \pm\infty} \mathbf{x}(\lambda) = f\frac{\mathbf{A} + \lambda\mathbf{D}}{A_Z + \lambda D_Z}$$

$$= f\frac{\mathbf{D}}{D_Z} = f\begin{pmatrix} D_X/D_Z \\ D_Y/D_Z \\ 1 \end{pmatrix}$$

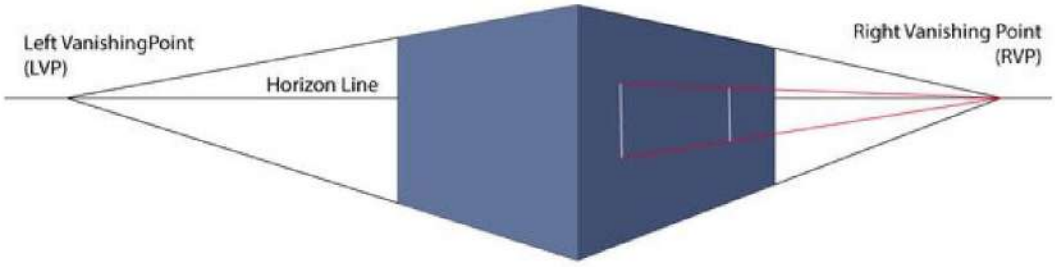
- v depends only on the direction D, not on A.
- Parallel lines have the same vanishing point.

- $l_{horizon} = P^{-T} l_\infty$
  - Each set of parallel lines intersects at a point at infinity, and l_∞ is the line that passes through such set of points at infinity, where l_∞ is **orthogonal lines**.
  - The projective transformation of l_∞ to the image plane is no longer a line at infinity.
  - $l_{horizon}$ = **horizon line = vanishing line = eye level line**
- $l_{horizon} = K^{-T} n$
  - n is the normal of a plane in a 3D space

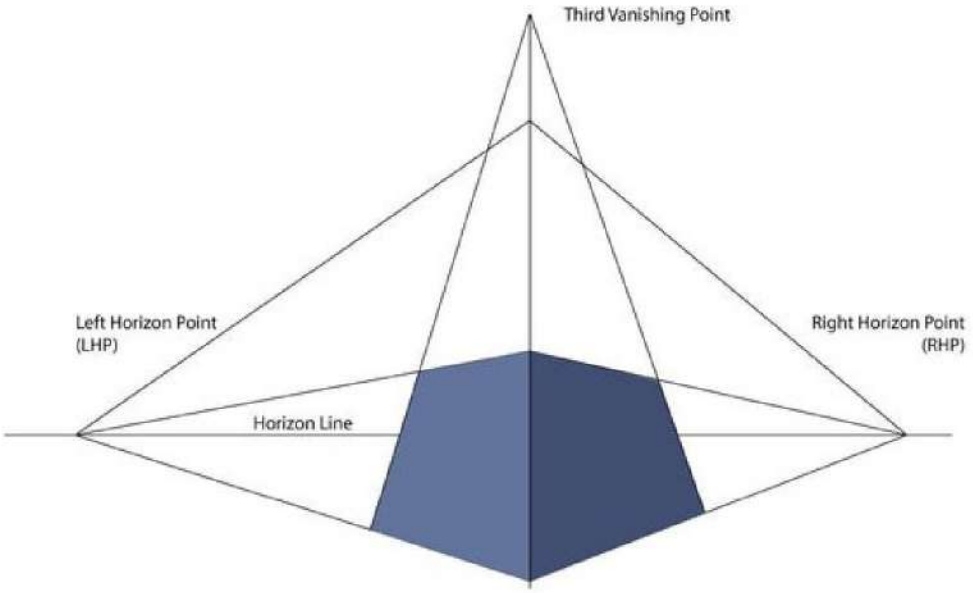
- One-Point Perspective

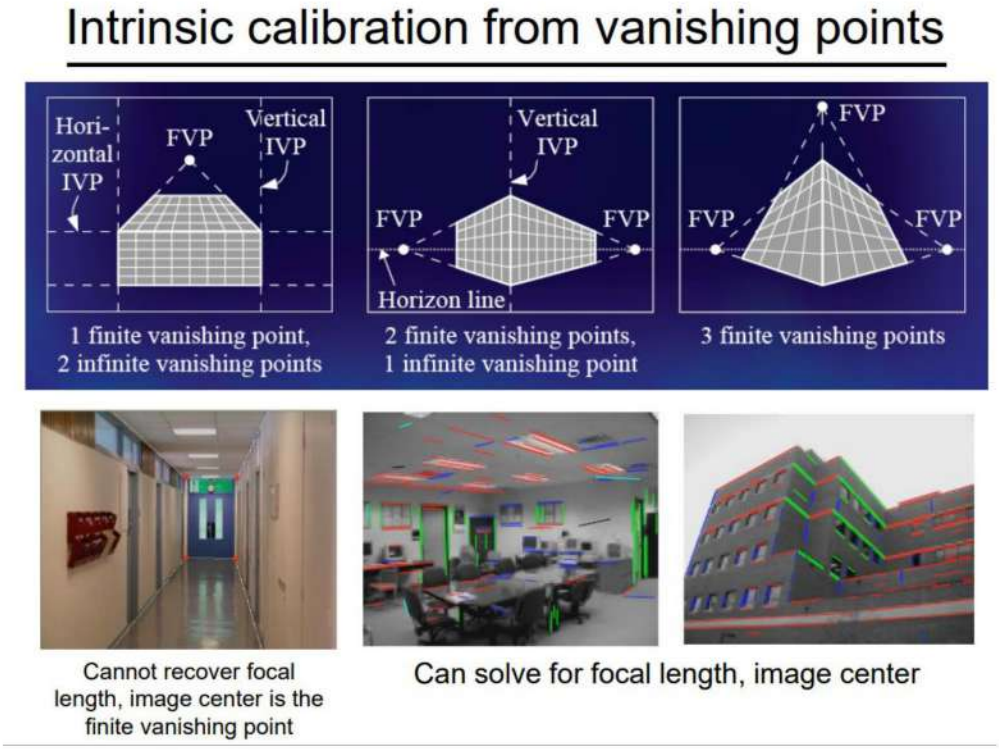


- Two-Point Perspective



➢ Three-Point Perspective





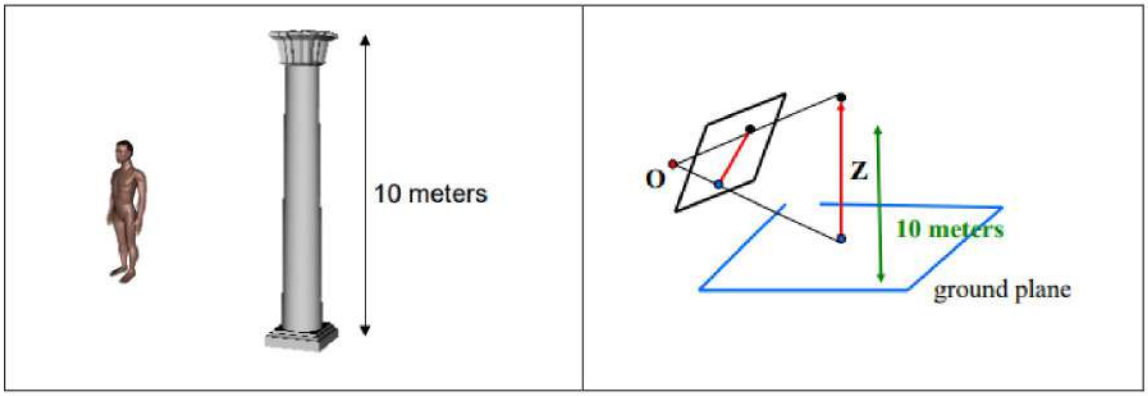
➢ Calibration from vanishing points

- Solve for K (focal length, principal point) using 3 orthogonal vanishing points
- Get rotation directly from vanishing points once K is known
- Advantages
  - No need to estimate 2D-3D correspondences
  - Could be completely automatic

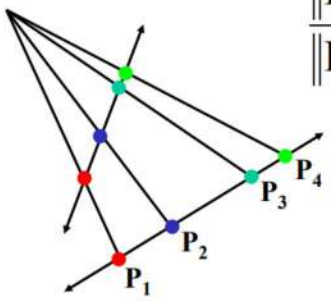
- Disadvantages
  - Only applies to certain kinds of scenes
  - Inaccuracies in computation of vanishing points
  - Problems due to infinite vanishing points

11.1.2 Application: Single view metrology

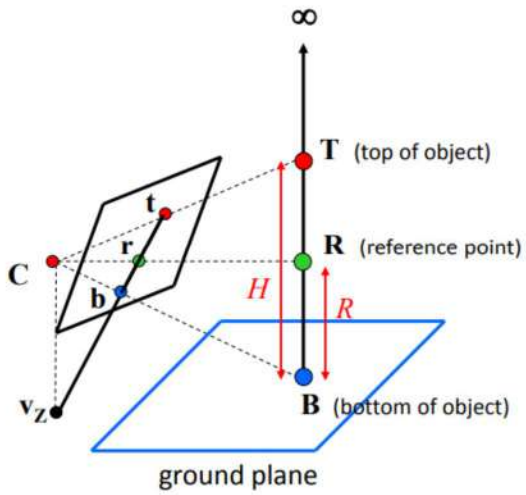
➢ How tall is the man in this image?



➢ The cross-ratio of four points:



$$\frac{\|\mathbf{P}_3-\mathbf{P}_1\|\,\|\mathbf{P}_4-\mathbf{P}_2\|}{\|\mathbf{P}_3-\mathbf{P}_2\|\,\|\mathbf{P}_4-\mathbf{P}_1\|}$$

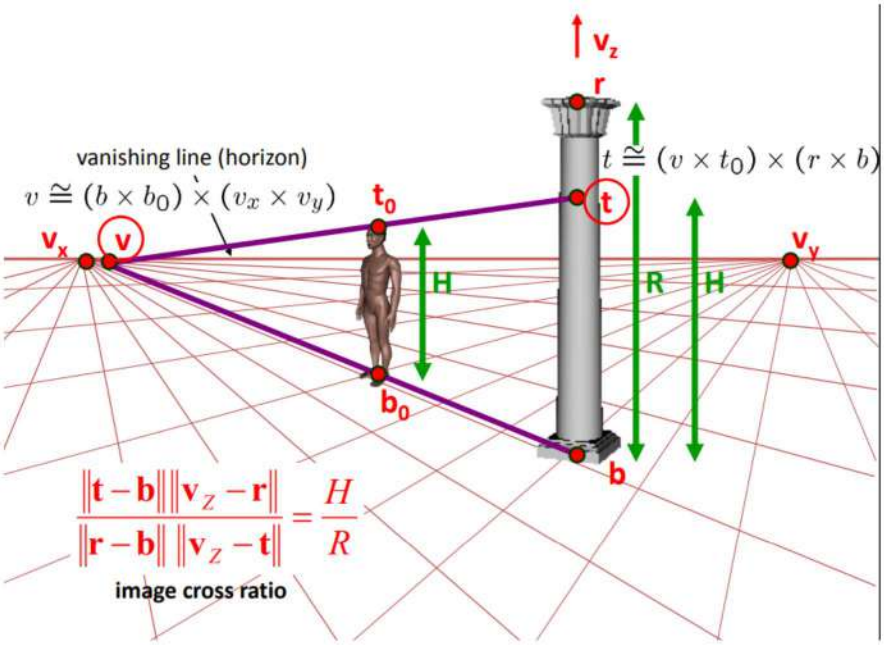


$$\frac{\|\mathbf{T}-\mathbf{B}\|\,\|\infty-\mathbf{R}\|}{\|\mathbf{R}-\mathbf{B}\|\,\|\infty-\mathbf{T}\|}=\frac{H}{R}$$

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

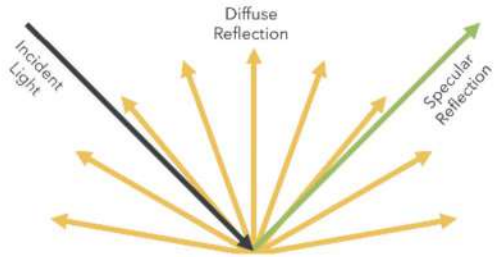


2.2 Photometric image formation

2.2.2 Reflectance

- Specular Reflection (鏡面反射) → for smooth surfaces
- Diffuse Reflection (漫射) → for rough surfaces
- Phong reflection = ambient + diffuse + specular

- **Diffuse reflections** are reflections where an incident light ray is reflected in many angles.
- **Specular reflections** are reflections where an incident light ray is reflected in one angle.
- In computer graphics, **Lambertian reflection** is often used as a model for **diffuse reflection**; the Lambertian Reflection Model models a perfect diffuse surface that scatters incident illumination equally in all directions.

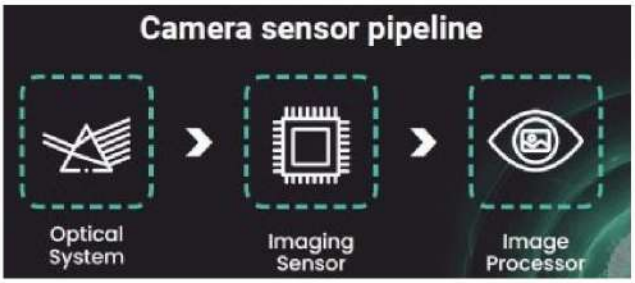


- The above lighting model is based on the **Lambert's cosine law**.
- Two properties of Lambertian reflection:
  (1) The surface's luminance is **isotropic**.
  (2) The **perfectly reflecting diffuser (PRD)**, a surface that is perfectly Lambertian, has a 100% reflectance (所有入射光全部反射).

- The luminance on the Lambertian Surface is **isotropic**:
  Light falling on Lambertian surface is scattered such that the **apparent brightness** of the surface to an observer is the same **regardless of the observer's angle of view**. Besides, the luminous intensity obeys **Lambert's cosine law**.

2.3 The digital camera

- CCD: Charge-Coupled Device
- CMOS: Complementary Metal-Oxide-Semiconductor
- A/D: Analog-to-Digital Converter
- JPEG: Joint Photographic Experts Group
- ISO: International Standards Organization
- image sensing pipeline
  = camera sensor pipeline
  = optical system → imaging sensor → image signal processor (ISP)

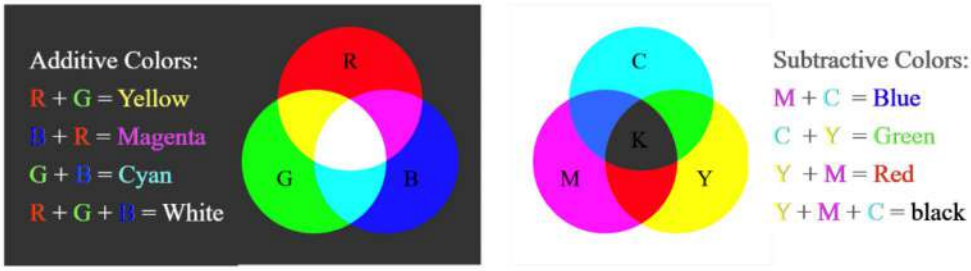


Camera Sensor Pipeline		
optical system	imaging sensor	image signal processor (ISP)
(input) scene radiance → optics → aperture (光圈) → shutter (快門) → (output) optical irradiance	(input) optical irradiance → CCD/CMOS sensor with color filter array (CFA) → pixel vignetting → analog gain (ISO) → A/D converter → (output) (12-bit) raw image (mosaiced, linear)	(input) raw image → demosaicing → noise reduction/sharpening → white balance → gamma correction → JPEG compression → (output) (8-bit) RGB image (JPEG, nonlinear)

2.3.1 Sampling and aliasing

$f_{samping} \geq 2f_{\max}$ to avoid aliasing

2.3.2 Color



(a) Additive colors: RGB (a) Subtractive colors: CMYK

Figure 2.27 Primary and secondary colors: (a) additive colors red, green, and blue can be mixed to produce cyan, magenta, yellow, and white; (b) subtractive colors cyan, magenta, and yellow can be mixed to produce red, green, blue, and black.

- magenta 洋紅
- cyan 青色
- A **color filter array (CFA)** is a mosaic of color filters (generally red, green and blue) that overlays the pixels comprising the sensor.
- A **demosaicing** (= **color interpolation**) algorithm is a digital image process used to reconstruct a full color image from the incomplete color channels output from the **color filter array (CFA)**.

Chapter 3 Image processing

3.1 Histogram equalization

- Histogram equalization: to use histogram to enhance the contrast
- How to implement histogram equalization?

  Given an input gray-scale image,

  (1) Histogram
  (2) PDF: Probability Density Function
  (3) CDF: Cumulative Distribution Function
  (4) Round Value in CDF
  (5) Look-up table
- How to implement histogram equalization?

  Given an input gray-scale image,

  (1) Total number of pixels = ?
  (2) PDF: (Histogram) plot of (# of pixels) – (gray-scale value)
  (3) CDF: sum of cumulative pixels
  (4) Normalization = CDF / total number of pixels
  (5) New = Normalization * 255

- To eliminate noises → **AHE: adaptive histogram equalization**
- Adaptive histogram equalization works by **dividing** an image into an M x N grid and then applying histogram equalization **locally** to each grid.

3.2 Linear filtering

Assume an image I and a kernel h.

- (Cross-) Correlation filtering: $g = h \otimes I$

$$g(i,j) = \sum_{u=-k,v=-k}^{k} h(u,v)I(i+u,j+v)$$

- Convolution filtering: $g = h * I$

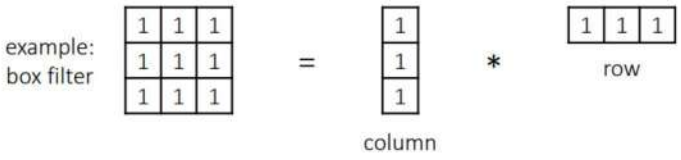
(The kernel h is rotated 180 degrees in the convolution operation)

$$g(i,j) = \sum_{u=-k,v=-k}^{k} h(u,v)I(i-u,j-v)$$

- The only difference between Convolution and (Cross-) Correlation is that in Cross-Correlation there is no mirroring in function g.

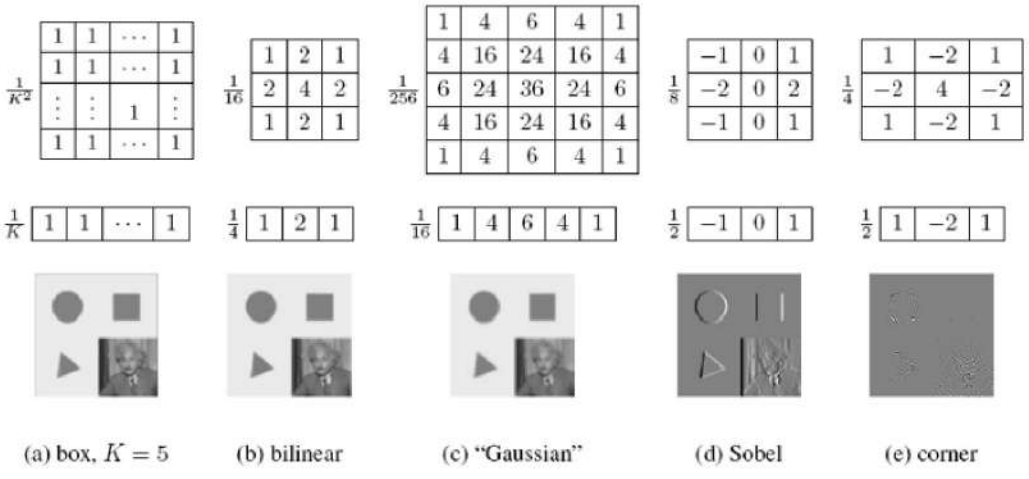
3.2.1 Separable filtering

- A 2D filter is separable if it can be written as the product of a "column" and a "row".



3.2.2 Examples of linear filtering

- Examples of separable filters



(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

3.2.3 Band-pass and steerable filters

- A **Laplacian filter** is an edge detector used to compute the **second derivatives** of an image.
- Laplacian operator

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

- The following array is an example of a 3x3 kernel for a Laplacian filter.

Positive Laplacian	Negative Laplacian
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

- Laplacian of Gaussian (LoG) = Gaussian blur → Laplacian operator
- Common 'derivative' filters (application: edge detection)
  Sobel / Scharr / Prewitt / Roberts filter

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

3.3 More neighborhood operators

3.3.1 Non-linear filtering

- E.g., median filtering

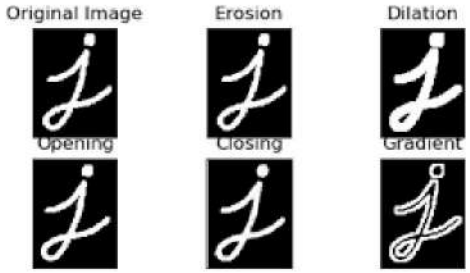
3.3.2 Bilateral filtering

- A **bilateral filter** is a non-linear, **edge-preserving**, and noise-reducing **smoothing** filter for images.

3.3.3 Binary image processing

- **Morphology**: morphology is a broad set of image processing operations that process images based on shapes.
- A **structuring element** is a matrix consisting of only 0's and 1's that can have any arbitrary shape and size.
- **Fit**: When all the pixels in the structuring element cover the pixels of the object, we call it Fit.
- **Hit**: When at least one of the pixels in the structuring element cover the pixels of the object, we call it Hit.
- **Miss**: When no pixel in the structuring element cover the pixels of the object, we call it miss.

- binary morphological operators:

Assume a binary image I and the structuring element S. (a) **Erosion**: $I \ominus S$ (b) **Dilation**: $I \oplus S$ (c) **Opening**: $(I \ominus S) \oplus S$ (d) **Closing**: $(I \oplus S) \ominus S$	

3.4 Fourier transforms

- Euler's formula

$$Ae^{j\theta} = A\cos\theta + jA\sin\theta$$

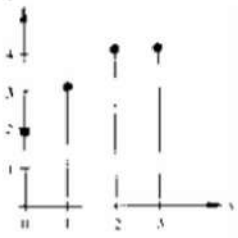
- Discrete 1-D Fourier Transform

$$\mathrm{F(k)} = \frac{1}{N}\sum_{n=0}^{N-1} f(n)e^{-\frac{j2\pi}{N}kn}$$

Where

- $f(n)$ is the input signal, n = 0, 1, ..., N-1
- $F(k)$ is the Frequency spectrum, which is periodically extended with period N.

- Example: 1-D Fourier Transform

	Given: ➢ N = 4 ➢ f(0) = 2 if k = 0. ➢ f(1) = 3 if k = 1. ➢ f(2) = 4 if k = 2. ➢ f(3) = 4 if k = 3.

- F(k=0) = (1/4) $\sum$ f(n)exp(0) = (1/4) * [f(0) + f(1) + f(2) + f(3)] = 13/4
- F(k=1) = (1/4) $\sum$ f(n)exp(-j2 π n/4)
  = (1/4) * [f(0) + f(1)exp(-j π /2) + f(2) exp(-j π) + f(3) exp(-j3 π /2)]
  = (1/4) * [2 + 3(0-j) + 4(-1) + 4(0+j))]
  = -0.5 + j0.25
- F(k=2) = (-1/4)(1+j0)
- F(k=3) = (-1/4)(2+j)
- Find |F(0)|, |F(1)|, |F(2)|, |F(3)|
  E.g., |F(1)| = sqrt(0.5 * 0.5 + 0.25 * 0.25) = sqrt(5) / 4
- Frequency spectrum: |F(k)|– k Plot (where k = 0, 1, 2, 3)

3.5 Pyramids and wavelets

3.5.1 Interpolation

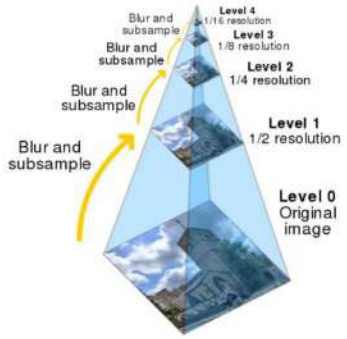
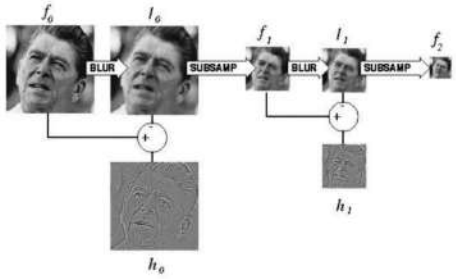
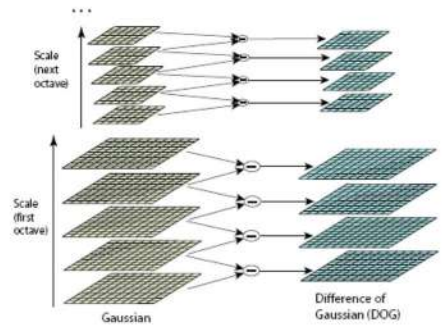
- To upsample an image to a higher resolution

3.5.2 Decimation

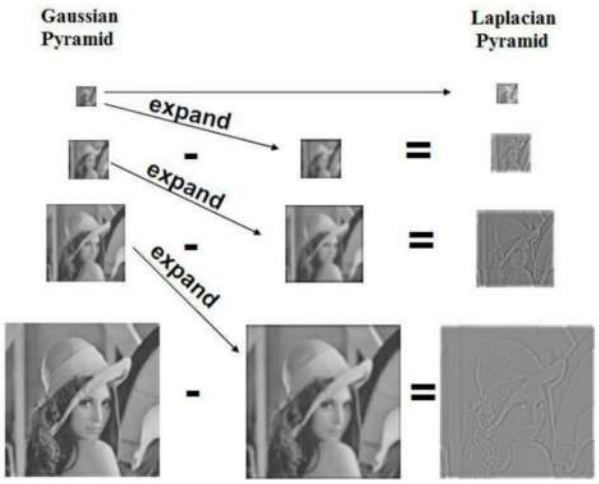
- To downsample an image to a lower resolution

3.5.3 Multi-resolution representations

- **Image pyramids** (or **image scale space**) is a method to handle images in different scales, e.g.,
  - Gaussian scale space (**Gaussian pyramid**)
  - Laplacian of gaussian (**LoG**) scale space
  - Difference of gaussian (**DoG**) scale space

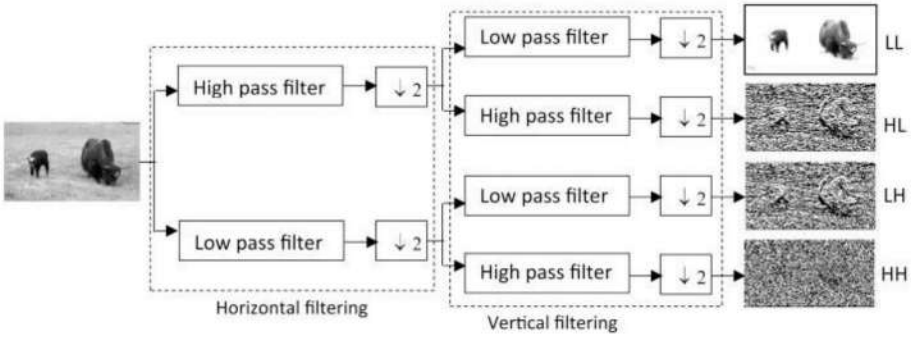
Gaussian pyramid	LoG	DoG
		

- **Gaussian pyramids** use the **Gaussian average** to scale down an image from high resolution to lower resolution.
- A **Laplacian pyramid** is very similar to a Gaussian pyramid but saves the difference image of the blurred versions between each levels.
- In a Laplacian pyramid, $\mathbf{L_i = G_i - expand(G_{i+1})}$

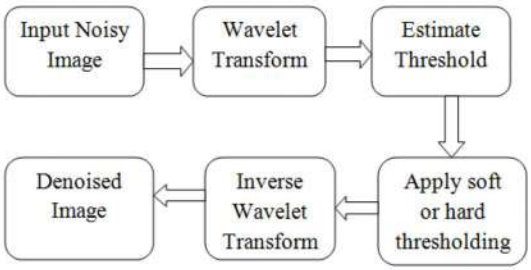


3.5.4 Wavelets

- **Wavelets** represent the scale of features in an image, as well as their position.
- Decomposition of an image 2-D **discrete wavelet transform** (2-D DWT)

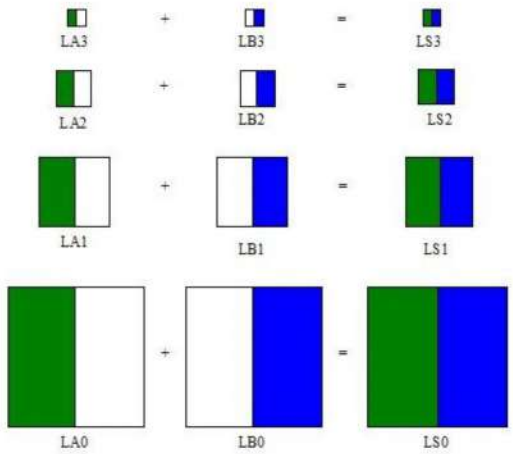
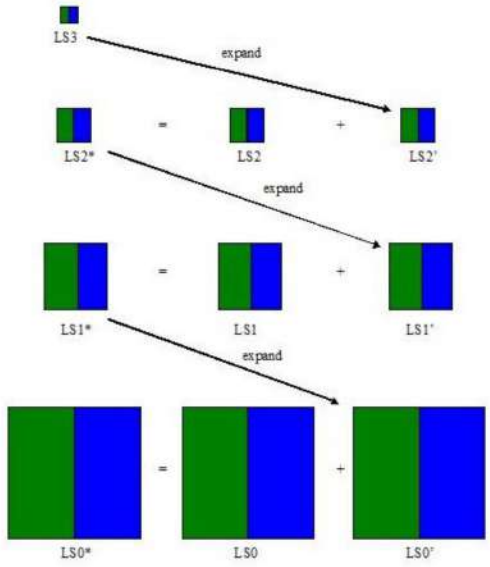


- **low_low (LL)** is the approximation of the input 2D signal at a larger scale.
- low_high, high_low, and high_high correspond to the detail information along the column, row, and diagonal directions.
- Examples of Wavelets: de-noising, compression, image fusion
- Example: de-noising



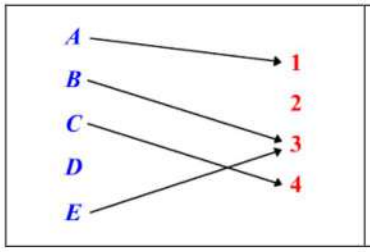
3.5.5 Application: Pyramid Blending

- Pyramid Blending
  - (1) Load the two images, Ia and Ib
  - (2) Find the Gaussian Pyramids for Ia and Ib
  - (3) From Gaussian Pyramids, find their Laplacian Pyramids
  - (4) Now join the left half of Ia and right half of Ib in each levels of Laplacian Pyramids
  - (5) Finally from this joint image pyramids, reconstruct the original image.

Step (4): Join to get Laplacian Pyramid	Step (5): Reconstruct the Gaussian Pyramid $G_i = L_i + \text{expand}(L_{i+1})$
	

3.6 Geometric transformations

- The **domain** of a function $f(x)$ is the set of all values for which the function is defined
- The **range** of the function $f(x)$ is the set of all values that f takes.
- Example

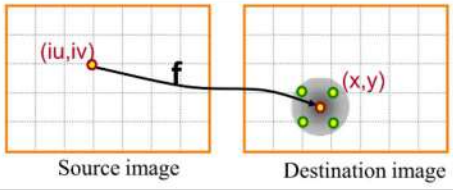
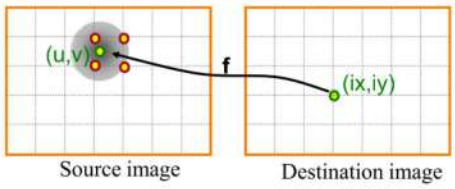


- domain = {A,B,C,E}
- range = {1,3,4}

- example: image transformation
  - image filtering: range is modified
  - image warping: domain is modified

3.6.1 Parametric transformations

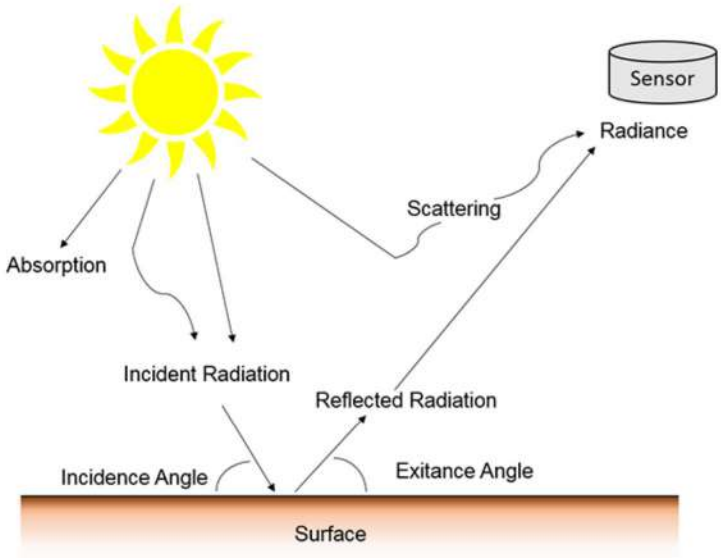
- **Image warping** is the process of digitally manipulating an image such that any shapes portrayed in the image have been significantly distorted.

Warping	Forward Warping	Inverse Warping
		
Idea	Mapping	Resampling
Technique	**Splatting**	**Bilinear Interpolation**

Chapter 10 Computational photography

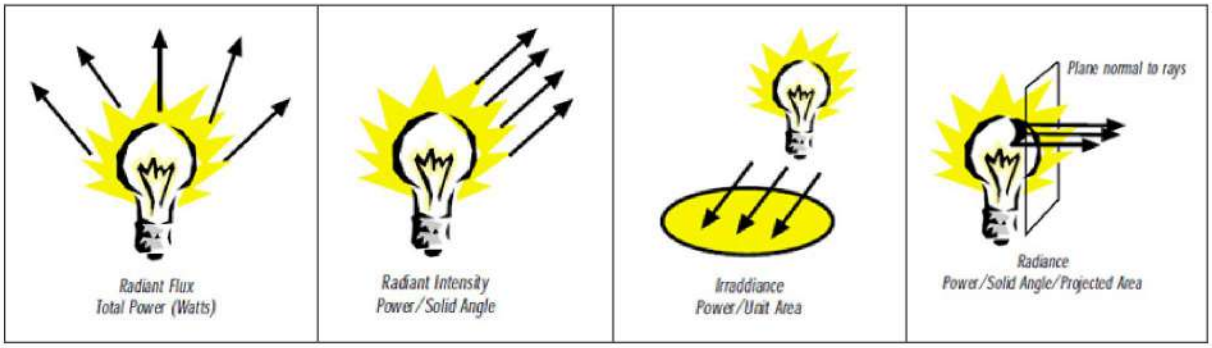
- **Computational photography** refers to digital image capture and processing techniques that use **digital computation** instead of optical processes.
  - Typical modes from your smart phone: portrait mode, panorama mode, high dynamic range (HDR) mode, night mode

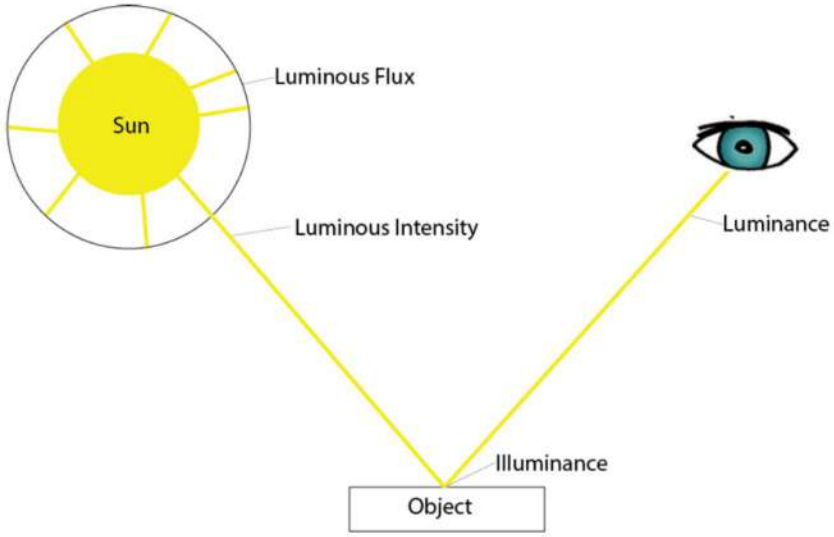
10.1 Photometric calibration



- **Radiometry** is the science of the measurement of electromagnetic (EM) radiation.
- **Radiant Flux** (unit: Watt): the **total** radiant power emitted from a source or received by a surface.
- **Radiant Intensity** (unit: Watt/sr): the **directed** angular density of radiation from a source.
- **Irradiance** (unit: W/m^2): a measure of radiant flux incident on an object's surface (radiant flux per unit area).

- **Radiance** (輻射率) (unit: $W \cdot sr^{-1} m^{-2}$): a measure of the total radiant intensity per unit projected area.
- **Reflectance** (反射率) is the proportion of the radiation striking a surface to the radiation reflected off of it, and it is **dimensionless**.





- **Luminous Flux** (unit: lumen) is the measure of the total light output of a luminous source.
- **Luminance** (unit: candela/area) is the amount of light emitted, passing through or reflected from a surface.
- **Illuminance** (unit: lux) describes the amount of light falling onto a surface area.

10.1.1 Radiometric response function

- **Radiometric response function** f is a function that transforms optical irradiance E into 8-bit pixel values Z that are the output from the camera.

$$Z_{ij} = f(X_i)$$

  Where the exposure $X_i = E_i \Delta t_j$.
  - Factors: (1) aperture and shutter, and (2) analog gain (ISO)

10.1.2 Noise level estimation

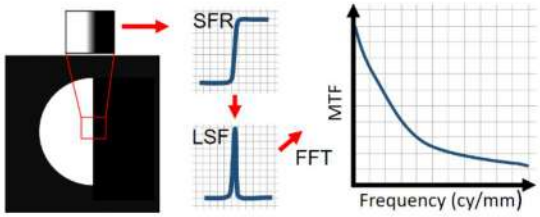
- Noise level depends mainly on analog gain (ISO).

10.1.3 Vignetting

- Lens **vignetting**, also known as "**light falloff**", is a reduction in brightness or saturation on the periphery of an image.
  - Vignetting most occurs at large apertures.

10.1.4 Optical blur (spatial response) estimation

- The **optical transfer function (OTF)** of an optical system such as a camera, microscope, human eye, or projector, specifies how different spatial frequencies are captured or transmitted.
  - Formally, the OTF is defined as the **Fourier transform of the point spread function (PSF)**.
  - Sometimes, the OTF is defined as the **modulation transfer function (MTF)**.
- The **Point Spread Function (PSF, = blurring kernel)** of an optical system is the irradiance distribution that results from a **single** point source.
  - The image of a complex object is the convolution of that object and the PSF, namely, **image = specimen * PSF**
  - shorter wavelengths of light (e.g., blue light, 450 nm) → smaller PSF → better resolution
- The **Modulation Transfer Function (MTF)** is the ability of an optical system to transfer **contrast** at a particular resolution from the object to the image.

▪ The **Spatial Frequency Response (SFR)** is measured using a half-Moon target which is tilted a few degrees. SFR measurement readings is a step function whose derivative is Line Spread Function (LSF). ▪ **Line Spread Function (LSF)** is a function of the angle of view which describes the **sharpness** of the camera. ▪ Modulation Transfer Function (MTF) is the Fourier Transform of the LSF.	

10.2 High dynamic range imaging

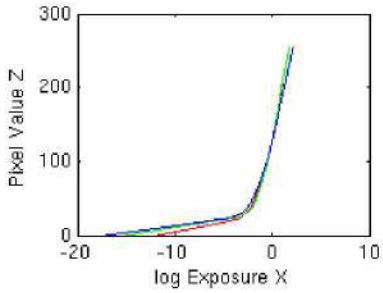
- Dynamic range is the range of the lightest tones to the darkest tones within a photo. Specifically, the **dynamic range** is "the **luminance range** of a scene being photographed".
  - Scenes with high dynamic ranges usually suffer from either over-exposure in the brighter regions or under-exposure in the darker regions.
- **High Dynamic Range Imaging** (**HDRI**) is a set of methods used in imaging and photography to allow a greater dynamic range between the lightest and darkest areas of an image than standard digital imaging or photographic techniques.
  - Objective: Given a set of low-dynamic range images of a scene captured under different exposures, try to reconstruct an image of the scene with high dynamic range; so that the reconstructed image has better exposures for both the darker pixels and brighter pixels.
  - Background Knowledge:

    → A **radiance map** is an **image** that represents the **true illuminance** values of a scene.

    → The Dodge and Burn Technique is used to lighten or darken areas of a photo. **Dodging** is used to lighten a spot on the photo, and **burning** is used to darken a particular area.

- The HDRI pipeline:
  - Step 1: Radiometric Response Function

    → To recover the response curves for the 3 color channels which map the pixel values to the log of exposure values.



  - Step 2: Radiance Map

    → To map the observer pixel values and the exposure times to radiance.
  - Step 3: (Local) Tone Mapping

    → To map the high-dynamic-range **radiance** values in real world to low-dynamic-range **luminance** values such that the details in both dark and bright regions are also visible.

10.2.1 Tone mapping

- idea: to decompose the radiance image into two layers: a **base layer** and a **detail layer**.
- **Local tone mapping** proposed by **Durand**'s algorithm in 2002

Step 1: Compute the log of the intensity: L = ln(I)
Step 2: Apply the bilateral filter to compute the base layer L_b = bilateral_filter(L)
Step 3: Compute the detail layer $L_d = L - L_b$
Step 4: Apply an offset and a scale to the base: $L_b' = (L_b - o) * s$
Step 5: Compute the final luminance value $I' = \exp(L_b' + L_d)$

- A **bilateral filter** is a non-linear, **edge-preserving**, and noise-reducing smoothing filter for images.
  - It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution.

10.3 Super-resolution, denoising, and blur removal

- **Super Resolution (SR)** refers to the task of enhancing the resolution of an image from low-resolution (LR) to high (HR).
- Note: **energy-based** method for optimization
  - The global energy should be minimized **E = Ed + λ Es**
  - Ed is the **data penalty** to measure the distance between the function f and a set of data points di = d(xi, xj), and Ed = $\sum_{i,j}[\, f(x_i, x_j) - d(i,j)\,]^2$
  - Es is the **smoothness penalty**, and ρ is some monotonically increasing function $E_s(d) = \sum_{i,j}[\, \rho(d(i,\ j) - d(i+1,\ j)) + \rho(d(i,\ j) - d(i,\ j+1))\,]$
- Multi-Image Super Resolution
  - A higher-resolution (HR) image of a scene is reconstructed from multiple observed LR images of the scene by using Bayesian methodology.
  - Problem: Find the super-resolved pixels s such that the following least-square problem (in a form of maximum likelihood) is minimized

$$\sum_k \|o_k(x) - D\{PSF_k(x) * s(w(x))\}\|^2$$

where o_k is the observed image, PSF_k is the Point Spread Function (PSF), w the warping function to warp an image based on a reference frame, D is the down-sampling operator, and * is the convolution operator.

  - The global energy should be minimized **$E = E_d + \lambda E_s$** , where
    → $E_d = \sum_k \|o_k(x) - D\{\boldsymbol{PSF}_k(x) * s(w(x))\}\|^2$
    → $E_s = \sum_{i,j}[\, \rho(s(i,\ j) - s(i+1,\ j)) + \rho(s(i,\ j) - s(i,\ j+1))\,]$
    where ρ is some monotonically increasing function, and E_s is an example of image prior called **Gaussian Markov Random Field (GMRF)**.

- Solution: Consider **image priors** and use **Bayesian inference** to solve the above **least-square problem** and reconstruct the higher-resolution (HR) image.

10.3.1 Color image demosaicing

- **Demosaicing** is the process of reconstructing a full-resolution color image from the sampled data acquired by a digital camera that apply a color filter array (CFA) to a single sensor.
- **Demosaicing** is considered as a **special case** of super resolution in our daily life.

10.4 Image matting and compositing

- **Matting**: to extract the (foreground) object from one image (background)
- **Compositing**: to insert an object into another image
- **Alpha Matting**: to extract a foreground object with soft boundaries from a background image.
  - In addition to RGB channels, α channel describe the **opacity** (or fractional coverage) at each pixel
  - Compositing equation **$C = (1 - \alpha) B + \alpha F$**
  - B is background, F is foreground
  - If the object is opaque (不透明), α = 1, the composite C = F

10.4.1 Blue screen matting

- **Blue screen matting** assumes that the blue color only shows in the background B (no blue in the foreground F)
- Then, **$F_B = 0$, $B_R = B_G = 0$.**
- The compositing equation is formulated as $C_i = (1 - \alpha) B_i + \alpha F_i$, where i is represented by 3 RGB channels. Then, these three equations can be simplified to 3 equations with 3 unknowns, and then α can be solved for each pixel.

$$C_R = \alpha F_R$$

$$C_G = \alpha F_G$$

$$C_B = (1 - \alpha) B_B$$

10.4.2 Natural image matting

- Basically, the **trimap** is a rough segmentation of an image into 3 region types: certain foreground, unknown, certain background.
- **Bayesian matting**
- **Knockout matting**

10.4.3 Optimization-based matting

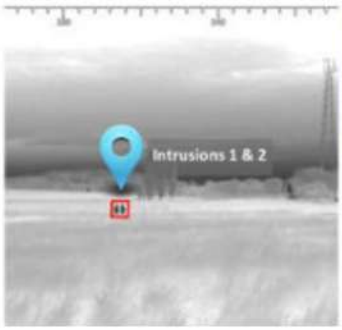
- **Border Matting**
- **Poisson Matting**

10.5 Texture analysis and synthesis

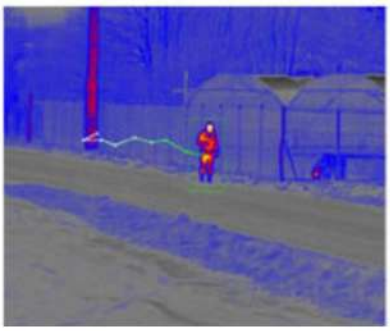
- **Texture synthesis** is the process of constructing a larger realistic versions of the texture from a small sample image.

Chapter 7 Feature detection and matching

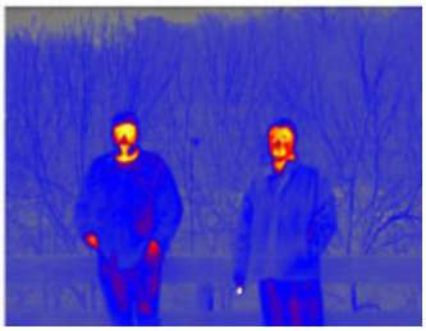
- To test observers' ability to identify targets through sensors, "**DRI**" were defined:
  - **Detection**: ability to distinguish an object from the background
    → to answer "location"!
  - **Recognition** (or **Classification**): ability to classify the object **class** (animal, human, vehicle, boat …)
    → to answer "class"!
  - **Identification**: ability to describe the object in details (a man with a hat, a jeep …)
    → to "analyze"!



Detection: At several kms, two targets are detected out of the background



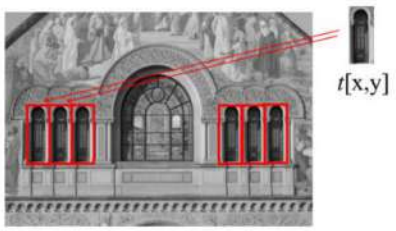
Recognition: A human is walking along the fence



Identification: Two males with trousers and jackets are identified, one is smoking.

- The **aperture problem** can be demonstrated by looking at a moving image through a small hole -- the aperture.
  - The "aperture problem" describes the intrinsic ambiguity of perceiving the motion of an object through a local observation.
- A **barber's pole illusion**
  - a visual illusion
  - The cylinder is rotating around a vertical axis, but the stripes look as if they are rising – which would be impossible.

➢ Template matching

▪ Problem: To locate an object, described by a template t[x,y], in the image s[x,y] ▪ Best match: mean-squared error ↓ or area correlation ↑	 s[x,y]

7.1 Points and patches

➢ An **interest point** is the intersection of two or more edge segments or the point at which the direction of the object's border rapidly changes.
  ▪ interest point ≈ keypoint ≈ salient point

➢ A (interest point) **detector** is an algorithm that chooses points from an image based on some criterion. Typically, an interest point is a local maximum of some function, such as a "cornerness" metric.

➢ A **descriptor** is a vector of values that describes the image patch around an interest point. It could be as simple as the raw pixel values, or it could be more complicated, such as a histogram of gradient orientations.
  ▪ For example, **Scale invariant feature transform (SIFT)** includes both a detector and a descriptor.

➢ The **correspondence problem** refers to the task of finding a set of points in one image which can be identified as the same points in another image.
  ▪ Two main approaches to the correspondence problems:
  (1) correlation-based, and (2) feature-based method
  ▪ **Correlation-based** method: to match image intensities to establish a correspondence.
  ▪ **Feature-based** method: to match a sparse set of image **features** to establish a correspondence.

➢ keypoint detection and matching pipeline: 3 stages
  ▪ Step 1: **feature detection**
  ▪ Step 2: **feature description**
  ▪ Step 3: **feature matching** or **feature tracking**

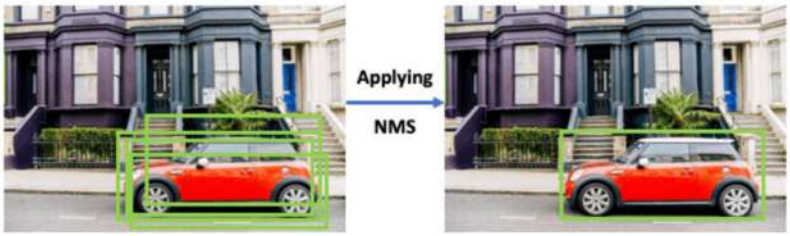
➢ For region detection, **invariance** transformations that should be considered are **illumination** changes, **translation**, **rotation**, **scale** and **full affine transform** (i.e. a region should correspond to the same pre-image for different viewpoints.

7.1.1 Feature detectors

- Feature Detection Algorithm

(1)	Compute I_x, I_y
	(where $I_x = I$*(derivative of Gaussian), or, I_x is the horizontal DoG).
(2)	Compute ${I_x}^2, {I_y}^2, I_x I_y$
(3)	larger kernel → Compute I_x, I_y
(4)	Compute a scalar interest measure
(5)	Find local maxima and detect keypoints

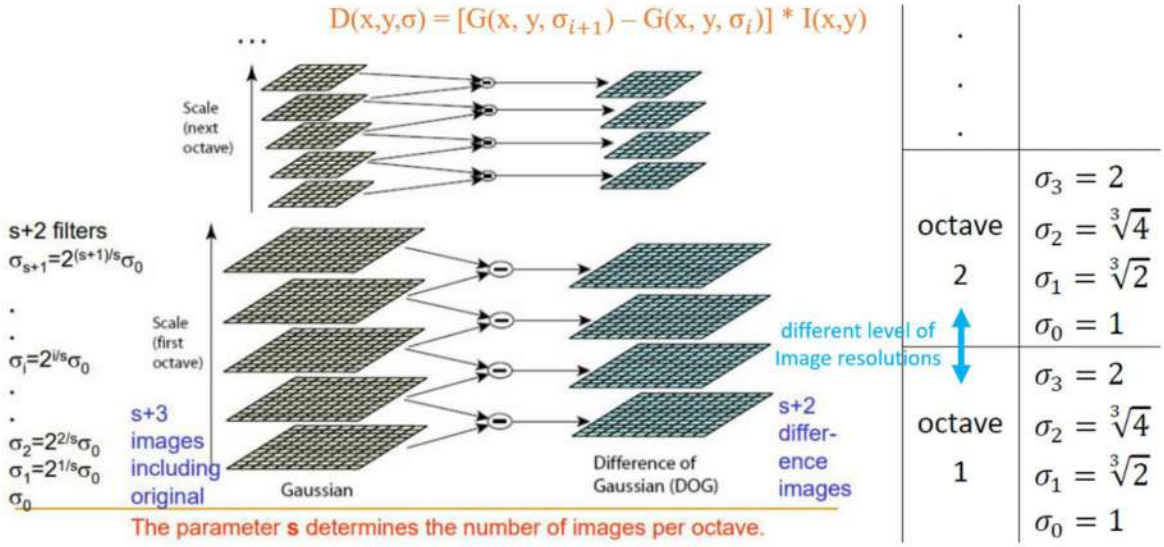
- **maximally stable extremal regions (MSER)** are used as a method of blob detection in images.
  - extremal: the property that all pixels inside the MSER have either higher (bright extremal regions) or lower (dark extremal regions) intensity than all the pixels on its outer boundary
- **Non-Maximum Suppression (NMS)** is a class of algorithms to select one entity (e.g., bounding boxes) out of many overlapping entities. NMS means "suppress the scores that are not maximum".



- **Adaptive Non-Maximum Suppression (ANMS)** is a non-maximum suppression algorithm that applies a **dynamic suppression threshold** to an instance according to the target density.
  - Features: local maximum

7.1.2 Feature descriptors

- **Scale invariant feature transform (SIFT)**
- SIFT includes both a detector and a descriptor. The detector is based on the **difference-of-Gaussians (DoG)**. The DoG detector detects centers of blob-like structures. The SIFT descriptor is a based on a **histogram of gradient orientations (HoG)**.
- DoG is an **approximation of the LoG at different scales** (LoG = Laplacian-of-Gaussains).
  - $D(x,y,\sigma) = [G(x,y,k\sigma) - G(x,y,\sigma)] * I(x,y)$
- In the SIFT scale space, we have **octaves** at different levels of image resolutions and different **scales** of window in each octave level (different σ of gaussian window).
  - If **s = 3** (i.e., 3 scales are used in per octave), then maximum stable keypoints are found.



- Steps of **SIFT algorithm**
  - **(1) Scale-space extrema detection**
    - DoG = difference of Gaussians
  - **(2) Keypoint localization**
    - Hessian matrix (2nd derivative matrix)
    - the eigenvalues are the maximal and minimal principal curvatures of the DoG function.
    - Edge: high maximal curvature / low minimal curvature
    - Corner: high maximal curvature / high minimal curvature
    - Keypoints on edges and flats are removed.
  - **(3) Orientation assignment**
    - histogram of gradient orientations (HoG) → dominant orientation
  - **(4) Keypoint descriptors**
    - Sort each HoG

7.1.3 Feature matching

- Matching Strategy
  - (1) distance threshold
  - (2) confusion matrix
  - (3) receiver operating characteristic (ROC) curve
  - (4) indexing structures

7.1.5 Feature tracking

- Example: **Kanade–Lucas–Tomasi (KLT)** tracker
- Pseudocode: KLT algorithm

Step 1. Find corners satisfying $\min(\lambda_1, \lambda_2) > \lambda$.
Step 2. For each corner, compute displacement to next frame (Lucas-Kanade method).
Step 3. Update each corner position.
Step 4. Repeat 2 and 3.
Step 5. Returns long trajectories for each corner point.

7.2 Edges

7.2.1 Edge detection

- 1st derivative (image gradient): Sobel filter
- 2nd derivative: Gaussian filter
- Laplacian of Gaussian (LoG)

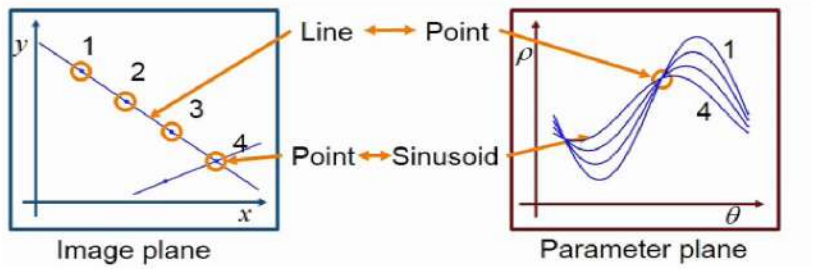
7.4 Lines

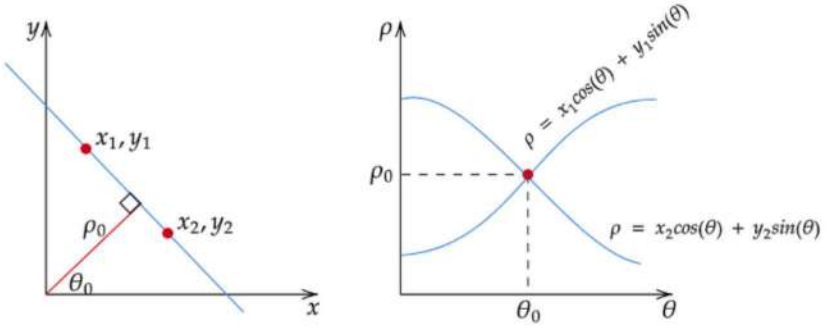
7.4.1 Successive approximation

- **B-spline** curve: a piecewise polynomial curve with minimum support.

7.4.2 Hough transforms

- The **Hough Transform** is a global method for finding straight **lines** (functions) hidden in larger amounts of other data.
  - A line in the image corresponds to a point in Hough parameter space.
  - A point in image space is mapped to a line in Hough parameter space.





- Line Detection using Hough Transform

Step 1: Set ranges of ρ and θ.
E.g., θ ∈ [0, 180], ρ ∈ [-d, d], d is the diagonal length of the edge.
Step 2: Initialize the 2D accumulator array H[ρ, θ] with zero values.
Step 3: for each edge point (x,y) in the image
    for θ = 0, 1, …, 180 degrees
        ρ = x cos(θ) + ysin(θ)
        H[ρ, θ] += 1
Step 4: Find (ρ, θ) where the local maximum H[ρ, θ] occurs.
Step 5: The detected line in the image is given by ρ = x cos(θ) + ysin(θ)

- RANSAC-based line detection

Chapter 6 Recognition

- A **pictorial structure** is a collection of parts arranged in a deformable configuration. Each part is represented using a simple appearance model and the deformable configuration is represented by spring-like connections between pairs of parts.
- **CRAM** stands for **Challenge-Response Authentication Mechanism** and it is a set of protocols used for authenticating a user by giving them a challenge and access is provided to the user only if they answer the challenges correctly. CAPTCHA is one example of Challenge Response Authentication.
- A **CAPTCHA**, a contrived acronym for "**Completely Automated Public Turing test to tell Computers and Humans Apart**", is a type of challenge–response test used in computing to determine whether the user is human.

6.1 Detection

6.3.1 Face detection

(1) Feature-based (= geometric = part-based) face detection
- geometric primitives: points, curves
- Features: eyes, nose, and mouth

(2) Template-based face detection
- **Active Appearance Model (AAM)** is
  (a) a statistical model of the shape and
  (b) grey-level appearance of the deformable object of interest.

(3) Appearance-based face detection

- To scan over small overlapping rectangular patches of the image searching for likely face candidates.
- Methods:
  (a) (K-means) Clustering and PCA
  (b) support vector machines (SVM)
  (c) Neural networks
  (d) Boosting
  (e) Deep networks

➢ **Eigenfaces** refers to an **appearance-based** approach to face detection.

- Specifically, the eigenfaces are the **eigenvectors of the covariance matrix** of the set of face images, where an image with N pixels is considered a point (or vector) in N-dimensional space.

6.3.0 Object detection

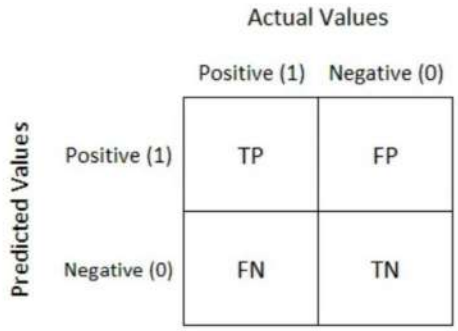
➢ Examples of modern object detectors: VGG-16, ResNet-50, X-152-FPN

➢ Two approaches to object detection using CNNs

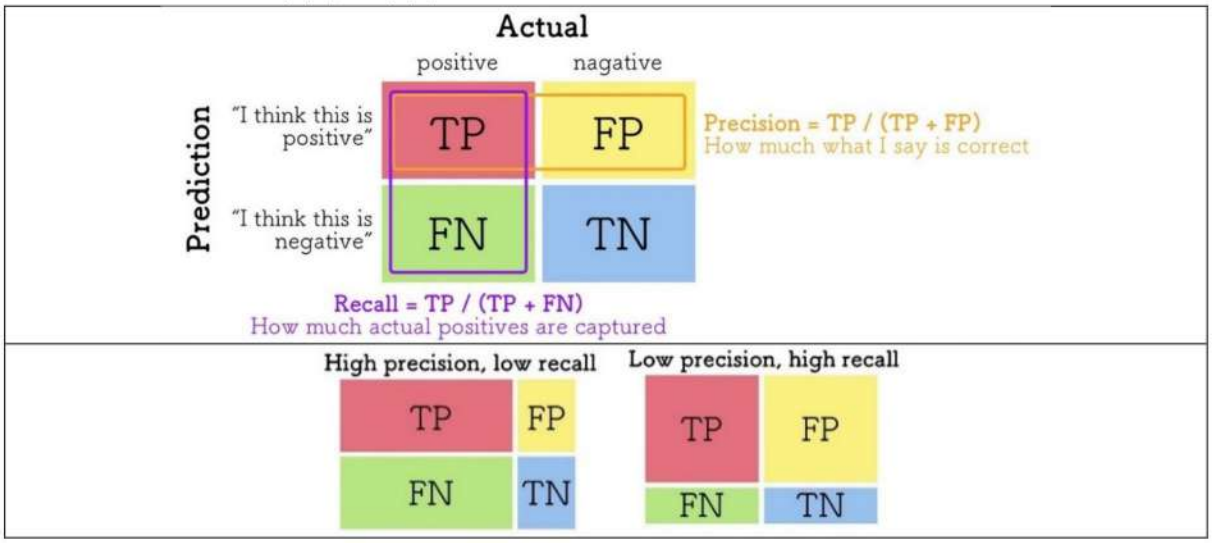
- **One-stage** approach makes a fixed number of predictions on grid.
  - Examples: **SSD** (Single Shot MultiBox Detector), **YOLO** (You Only Look Once).
- **Two-stage** approach leverages a proposal network to find objects and then uses a second network to fine-tune these proposals and output a final prediction
  - Examples: **R-CNN** (Region-Based Convolutional Neural Networks)
  - The first stage of R-CNN identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region.

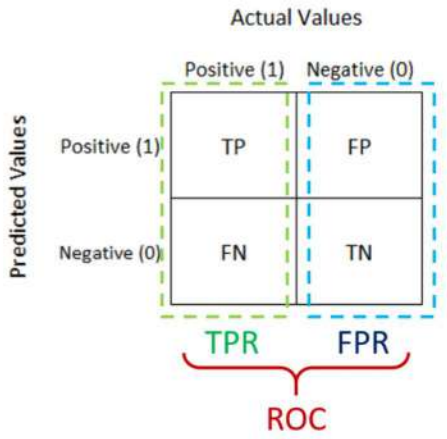
6.2 Image classification

➢ **Confusion matrix** is used to measure the performance of the **classification** model.

➢ Predicted values – Values that are predicted by the model. ➢ Actual Value – Values that are actually in a dataset. ➢ True Positive (TP): Values that are actually positive and predicted to be positive. ➢ False Positive (FP): Values that are actually negative but predicted to be positive. ➢ False Negative (FN): Values that are actually positive but predicted to be negative. ➢ True Negative (TN): Values that are actually negative and predicted to be negative.	

- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)
- **TPR** (True Positive Rate) = TP / (TP+FN)
  = **Recall** = **Sensitivity**
- **FPR** (False Positive Rate) = FP / (FP+TN)
- **Specificity** = TN / (FP+TN) = 1 – FPR
- Sensitivity ↑ , Specificity ↓
- TPR ↑ , FPR ↑
- receiver operating characteristic curve
  (= **ROC** curve) metrics = TPR + FPR
- Area under the ROC Curve (**AUC**) measures the entire 2D area underneath the entire ROC curve from (0,0) to (1,1).





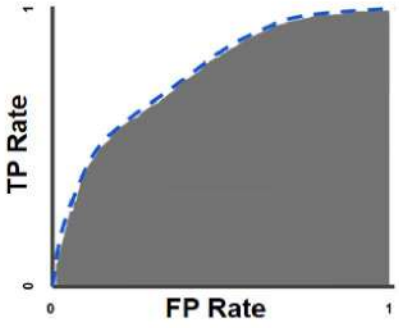
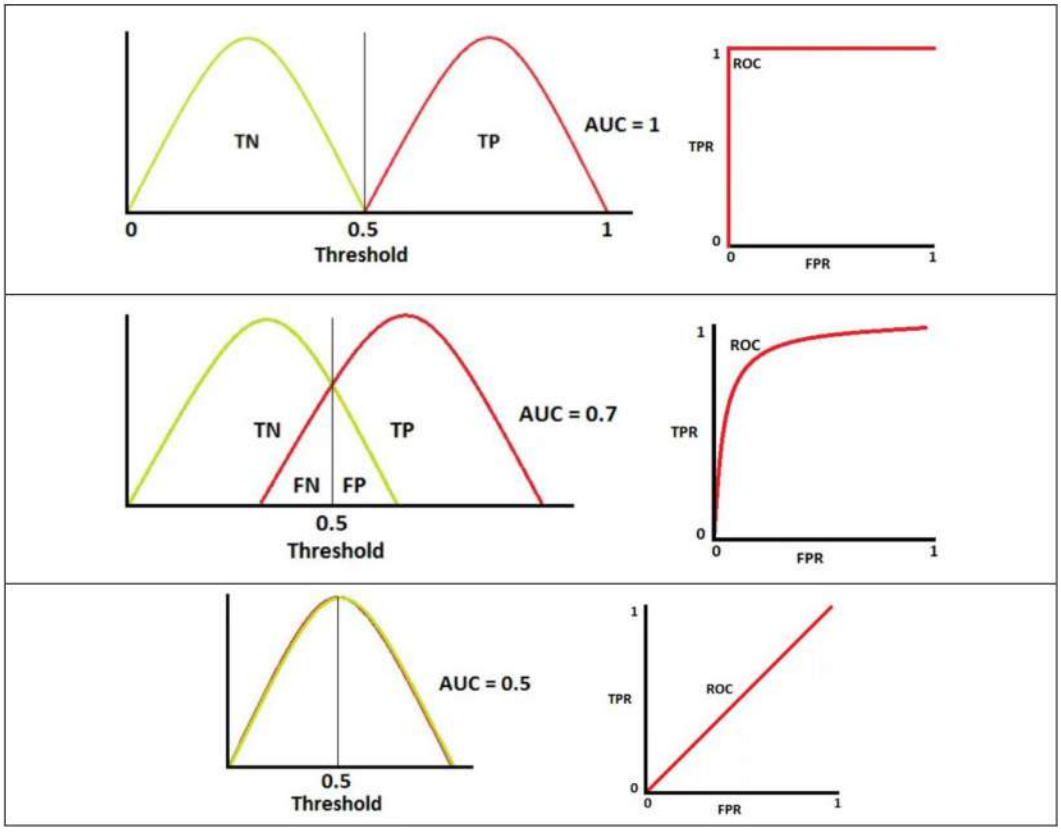
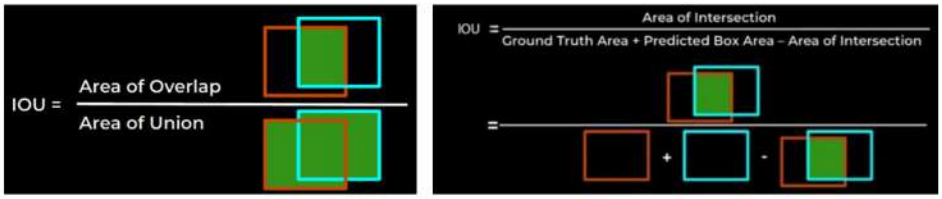


Figure 5. AUC (Area under the ROC Curve).



- Intersection over Union (IoU)
  - **Intersection over Union (IoU)** is the ratio of the overlap area to the combined area of prediction and ground truth. IoU values range from 0 to 1. Where 0 means no overlap and 1 means perfect overlap.



6.2.1 Feature-based methods

- A **bag of visual words** is a vector of occurrence counts of a vocabulary of local image features.
- BoVW consists of three main steps:
  Local feature extraction
  → codebook generation (by k-means)
  → coding (vector quantization, pooling)
  → Result: one feature vector per image.

7.5 Segmentation

7.5.1 Graph-based segmentation

- Graph-based segmentation (= segmentation by **graph cuts**) treats image segmentation as a graph partitioning problem.
- In graph theory, a **cut** is a partition of the vertices of a graph into two disjoint subsets.
- node → every pixel
- edge → weighting (pixel similarity)
- We can set a criterion, such as **normalized cuts**, to segment a graph.

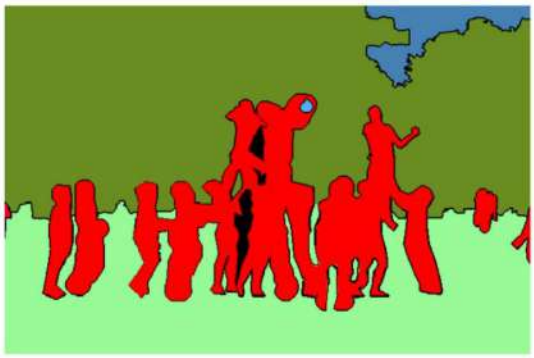
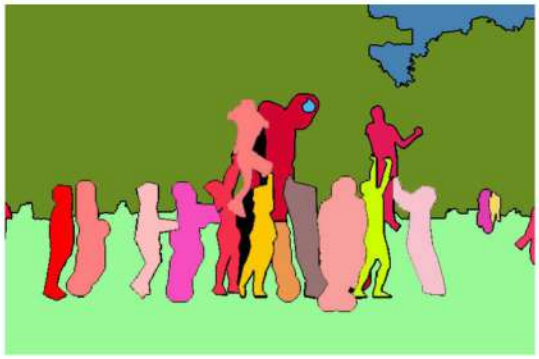
7.5.2 Mean shift segmentation

- **Mean-shift clustering** is a non-parametric, density-based **clustering** algorithm that can be used to identify clusters in a dataset.
- Mean shift is based on the concept of **Kernel Density Estimation** (KDE, or Parzen window technique), which is a way to estimate the probability density function (PDF) of a random variable.

6.4 Semantic segmentation

- Compare the following words:

Task: image classification → class = person, tree, grass, sky	
Task: object detection → find locations	

Task: semantic segmentation → class = ?	
Task: instance segmentation → one instance belongs to one class	
Task: panoramic segmentation → (no overlap)	

- **semantic segmentation**: to categorize each pixel in an image into a class or object.
  - In general, what is this?
- The **instance segmentation** task requires a method to segment each object instance in an image.
  - How many objects are there?
  - Example: R-CNN
- The **panoptic segmentation** task permits only one semantic label and one instance id to be assigned to each pixel.
  - No overlaps are possible by construction for panoptic segmentation.

6.5 Video understanding

- Approaches to Action Recognition for videos
  (1) Single Stream Networks: e.g., LSTM, 3D-ConvNet
  (2) Two Stream Networks (→ spatial + temporal stream): e.g., TwoStreamFusion
- **Image Captioning** is the task of describing the content of an image in words. This task lies at the intersection of computer vision and natural language processing.

Chapter 4 Model fitting and optimization

- Lasso, Ridge, and Elastic-Net are all **Linear Regression** family where the x (input) and y (output) are assumed to have a linear relationship. The main difference is how the model is **penalized** for its **weights**.
- Linear Equation: Y(X) = W X + b
- Linear regression has the cost function $\sum_{i=1}^{n}(y_i - y(x_i))^2$
- λ = α
- f(xi) = y(xi)
- total error = noise + bias * bias + variance
- Smoothing ↑ , bias ↑ , noise ↓ , variance ↓
- Lasso regression (assuming L1 penalty) has the cost function $\sum_{i=1}^{n}(y_i - y(x_i))^2 + \alpha \sum_{j=1}^{p}|w_j|$
- Ridge regression (assuming L2 penalty) has the cost function $\sum_{i=1}^{n}(y_i - y(x_i))^2 + \alpha \sum_{j=1}^{p}(w_i)^2$
- Elastic-Net regression has the cost function $\sum_{i=1}^{n}(y_i - y(x_i))^2 + \alpha \sum_{j=1}^{p}|w_j| + \alpha \sum_{j=1}^{p}(w_i)^2$

4.1 Scattered data interpolation

- Scattered data interpolation
  - Delaunay triangulation
  - Pull-push algorithm
- **terrain modeling**

Given a set of scattered points S = {pi}, for every point pi we have a height measured. Problem: To estimate the height at the point pj, where pj ∉ S. Solution: (1) Triangulation of S (2) Map every triangle to 3D space and apply the heights to the points (3) Height Interpolation: For every point inside a triangle, the height of the piecewise linear surface models the terrain.

- **Pull-Push Algorithm**

- their measured values. For the i-th measured data point,
  - interpolated[i] = measured[i]
- In the **push** step, the interpolated values at the remaining points (i.e., the points that are not measured data points) are updated based on the interpolated values of their **neighboring points**. For the i-th point that is not a measured data point,
  - interpolated[i] = (1 - weight) * interpolated [i] + weight * average_of_neighbors[i]

  - In the **pull** step, the interpolated values at the measured data points are updated to.

4.1.1 Radial basis functions

- A **radial basis function (RBF**) is a function that **maps** a point in a multidimensional space to a **scalar** value.
- The RBF is typically defined as a function of the **distance** between the input point and the center.
- There are many types of RBFs, but a common one is the **Gaussian RBF,** which is defined as:
  - Gaussian RBF Φ(x) = exp{ (- r^2) / (2 * sigma^2) }
  - x is the input point
  - r is the distance between x and a center point
  - sigma is a parameter that controls the width of the RBF
  - c^2 = 2 * sigma^2, and c determines the smoothness.
- If we want our function to exactly interpolate the data values, we solve the following equation to obtain the desired set of weights wk

$$\mathbf{f}(\mathbf{x}_k) = \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) = \mathbf{d}_k$$

Where xk are the locations of the scattered data points.

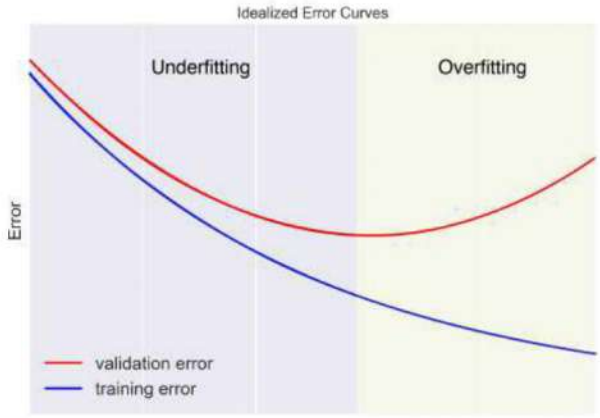
- We try to minimize the energy E with a weight penalty
- Energy **E = Ed + λ Ew**

$$\text{Ed} = \sum_k \| \sum_l \mathbf{w}_l \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) - \mathbf{d}_k \|^2$$

$$\text{Ew} = \sum_k \|\boldsymbol{w}_k\|^p$$

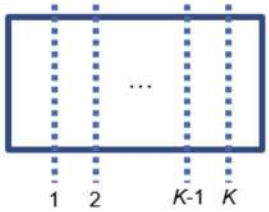
  - Lasso regression if p = 1
  - Ridge regression if p = 2
  - Kernel regression $f(x) = \frac{\sum_k d_k \emptyset(\|x - x_k\|)}{\sum_k \emptyset(\|x - x_k\|)}$

4.1.2 Overfitting and underfitting



Cross validation:

1. Split the data into *K* folds.
2. We train for *K* runs, with different folds to be validation set.
3. Estimate the best result by averaging over all *K* training runs' result.



(*K*=5 is often used.)

4.1.3 Robust data fitting

- global energy E = Ed + λ Ew
  and $E_d = \sum_k \rho(\|r_k\|)$ where the residual error $r_k = f(x_k) - d_k$

4.2 Variational (= energy-based) methods and regularization

The global energy should be minimized

- global energy **E = Ed + λ Es**
- Ed is the data penalty to measure the distance between the function f and a set of data points di = d(xi, yi)
- Ed is the data penalty, and Ed = $\sum_i [f(x_i, y_i) - d_i]^2$
- Es is the smoothness penalty
- Example: One variable x
  - $\mathcal{E}_1 = \int \boldsymbol{f_x^2(x)}\, \boldsymbol{dx}$
  - $\mathcal{E}_2 = \int \boldsymbol{f_{xx}^2(x)}\, \boldsymbol{dx}$
  - $\mathcal{E}_1$ and $\mathcal{E}_2$ are the two variables that measure the **variation** (or **non-smoothness**)

- Example: f(x) = (1/3) * x^3. Calculate the variation of f in [0, 2] and [-1, 1], respectively.
  - f'(x) = x^2, f''(x) = 2x
  - the smoothness penalty **Es =** $\mathcal{E}_1 + \mathcal{E}_2 = \int f_x^2(x)\,dx + \int f_{xx}^2(x)\,dx$
  - Es(x ∈ [0, 2])= $\mathcal{E}_1 + \mathcal{E}_2 = \int_0^2 x^4 dx + \int_0^2 4x^2 dx = \frac{32}{5} + \frac{32}{3} \approx 17.07$
  - Es(x ∈ [-1, 1])= $\mathcal{E}_1 + \mathcal{E}_2 = \int_{-1}^1 x^4 dx + \int_{-1}^1 4x^2 dx = \frac{2}{5} + \frac{8}{3} \approx 3.07$
  - The fact of 17.07 > 3.07 implies that f(x) has greater variation in [0, 2]. That is, f(x) has less smoothness.

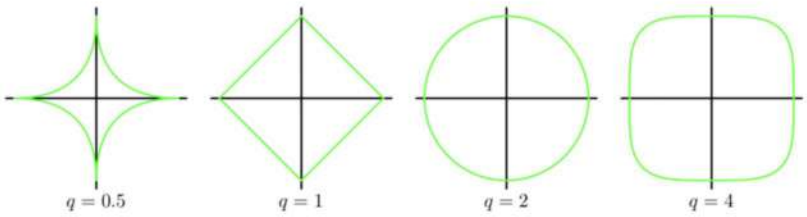
4.2.1 Discrete energy minimization

$$E = E_d + \lambda E_s = x^T A x - 2x^T b + c$$

- Minimizing the quadratic form is equivalent to solving the linear system of $Ax = b$.
- $x = [f(0,0) \ldots f(m-1, n-1)]^T$ is called state vector.
- A is Hessian (symmetric positive-definite matrix)
- b is the weighted data vector

4.2.2 Total variation

- Today, many regularized problems are formulated using L_1 norm, which is often called total variation.



4.2.3 Bilateral solver

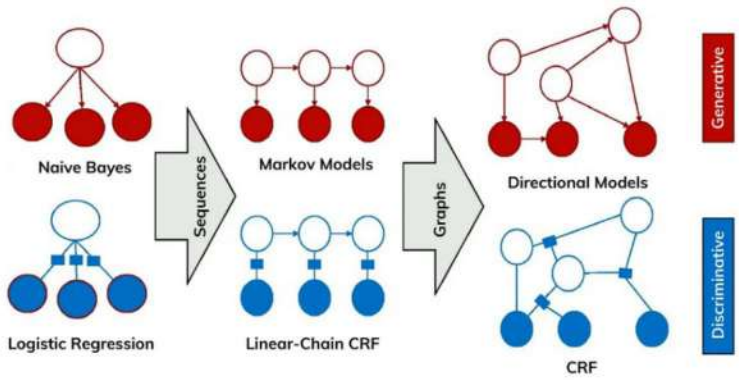
- Application: 3D reconstruction, panoramic stitching, augmented reality (AR)

4.2.4 Application: Interactive colorization

- Image colorization is the process of assigning colors to a grayscale image.
- Technique: edge-aware interpolation

4.3 Markov random fields

Markov Random Field (MRF): an **undirected graphical model** of a joint probability distribution.



4.3.1 Conditional random fields

- (classic) Bayesian model $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$,

  and the prior distribution p(x) is independent of the observations y.
- However, the above Bayes' rule cannot be applied to the smoothness term Es because Es depends on the data.
- A **conditional random field (CRF)** is simply a **conditional distribution p(x|y)** with an associated **graphical** structure.
- If p(x|y) is the posterior distribution, then its negative log likelihood is E(x|y)

$$E(\mathbf{x}|\mathbf{y}) = E_{\mathrm{D}}(\mathbf{x}, \mathbf{y}) + E_{\mathrm{S}}(\mathbf{x}, \mathbf{y})$$
$$= \sum_{p} V_p(x_p, \mathbf{y}) + \sum_{(p,q)\in\mathcal{N}} V_{p,q}(x_p, x_q, \mathbf{y})$$

  Where V is the potential function.
- A conditional random field (CRF) is a **discriminative** statistical modelling method that is used when the class labels for different inputs are **not** independent.
- For example, in image segmentation, the class label for the pixel **depends** on the label of its neighboring pixels also.

4.3.2 Application: Interactive segmentation

- Interactive segmentation is a technique for picking objects of interest in images according to users' input interactions.

Chapter 5 Deep Learning

Machine Learning				
(1) Supervised Learning		(2) Unsupervised Learning		(3) Reinforcement Learning
Classification	Regression	Clustering	Dimensionality Reduction	
KNN	Decision Tree	K-means	PCA	
Naive Bayes	Linear Regression	Mean Shift	Feature Selection	
SVM	Logistic Regression	K-Medoids	Linear Discriminant Analysis (LDA)	

➢ The main difference between supervised and unsupervised learning: **Labeled** data

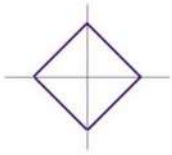
5.1 Supervised learning

5.1.1 Nearest neighbors

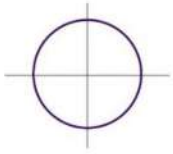
➢ K-nearest neighbors (KNN)

- Supervised
- Non-parametric (no learning)
- KNN = K labels + decision boundary
- Distance metric: L1 and L2 norm

L1 (Manhattan) distance $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$



L2 (Euclidean) distance $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$



5.1.2 Bayesian classification

Bayes' Theorem:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\cdot\mathbb{P}(A)}{\mathbb{P}(B)}$$

Let $X = \{x_1, x_2, ..., x_n\}$ be the features and C_i be the class i.

$\mathbb{P}(c_j|x_i) = \frac{\mathbb{P}(x_i|c_j)\cdot\mathbb{P}(c_j)}{\mathbb{P}(x_i)}$

Knowing that$\mathbb{P}(x_i|c_j) = \frac{Count(x_i,c_j)}{Count(c_j)}$, $Count(x_i, c_j)$ is the number of times that (x_i, c_j) appears in the examples and $Count(c_j)$ is the number of times that c_j appears.

This model is called 'naïve' because we naively assume independence between features given the class variable, regardless of any possible correlations.

With independence assumption:
$\mathbb{P}(x_1, x_2, ..., x_n|c_j) = \mathbb{P}(x_1|c_j) \cdot \mathbb{P}(x_2|c_j) \cdot ... \cdot \mathbb{P}(x_n|c_j)$

12.2 Basic Concepts

Let $X_1, \ldots, X_n$ be n observations sampled from a probability density $p(\mathrm{x} \mid \theta)$. In this chapter, we write $p(\mathrm{x} \mid \theta)$ if we view θ as a random variable and $p(\mathrm{x} \mid \theta)$ represents the conditional probability density of X conditioned on θ. In contrast, we write $p_\theta(\mathrm{x})$ if we view θ as a deterministic value.

12.2.1 The Mechanics of Bayesian Inference

Bayesian inference is usually carried out in the following way.

Bayesian Procedure

1. We choose a probability density $\pi(\theta)$ — called the prior distribution — that expresses our beliefs about a parameter θ before we see any data.
2. We choose a statistical model $p(\mathrm{x} \mid \theta)$ that reflects our beliefs about x given θ.
3. After observing data $\mathcal{D}_n = \{X_1, \ldots, X_n\}$, we update our beliefs and calculate the posterior distribution $p(\theta \mid \mathcal{D}_n)$.

By Bayes' theorem, the posterior distribution can be written as

$$p(\theta \mid X_1, \ldots, X_n) = \frac{p(X_1, \ldots, X_n \mid \theta)\pi(\theta)}{p(X_1, \ldots, X_n)} = \frac{\mathcal{L}_n(\theta)\pi(\theta)}{c_n} \propto \mathcal{L}_n(\theta)\pi(\theta) \tag{12.3}$$

where $\mathcal{L}_n(\theta) = \prod_{i=1}^n p(X_i \mid \theta)$ is the likelihood function and

$$c_n = p(X_1, \ldots, X_n) = \int p(X_1, \ldots, X_n \mid \theta)\pi(\theta)d\theta = \int \mathcal{L}_n(\theta)\pi(\theta)d\theta$$

is the normalizing constant, which is also called the evidence.

We can get a Bayesian point estimate by summarizing the center of the posterior. Typically, we use the mean or mode of the posterior distribution. The posterior mean is

$$\bar{\theta}_n = \int \theta p(\theta \mid \mathcal{D}_n)d\theta = \frac{\int \theta \mathcal{L}_n(\theta)\pi(\theta)d\theta}{\int \mathcal{L}_n(\theta)\pi(\theta)d\theta}.$$

- Naive Bayes Algorithm is a classification technique based on Bayes' Theorem with an independence assumption among predictors.
- In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

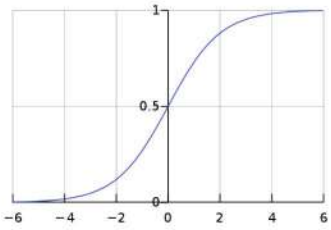
5.1.3 Logistic regression

- logistic regression

(1) binary logistic regression: two classes y ∈ {0, 1}.

(2) multinomial logistic regression: K classes y ∈ {1, 2, 3, ..., K}.

- Sigmoid Function: an S-shaped curve
- Linear regression: y = $\alpha_0 + \alpha_1 x_1 + \cdots + +\alpha_k x_k$

Logistic function: probability p = 1 / [1 + e^(-y)], where y is from linear regression	

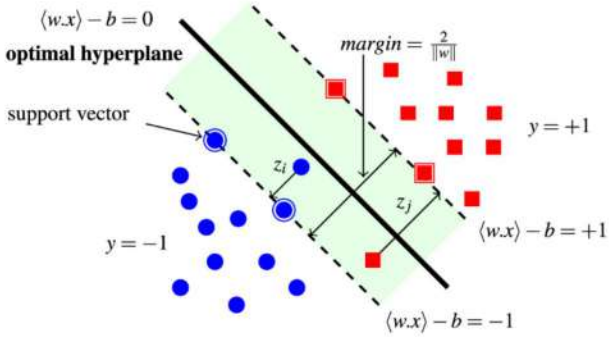
- Logistic function is the special case of the sigmoid function.
- Softmax function:

  The probability of classifying the instance as j is

$$p(y = j|x)\frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

	binary logistic regression	multinomial logistic regression
classification problem	binary classification	multi-class classification
sigmoid function	**logistic**	**softmax**
loss function	log loss (= binary cross-entropy loss)	cross-entropy loss

5.1.4 Support vector machines



- SVM creates a hyperplane or line (decision boundary) which separates data into classes.
- The objective of SVM is to maximize the margin between the data points and the hyperplane.
- **Hinge loss** is the loss function that helps maximize the margin.

Recall: SVM Training

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

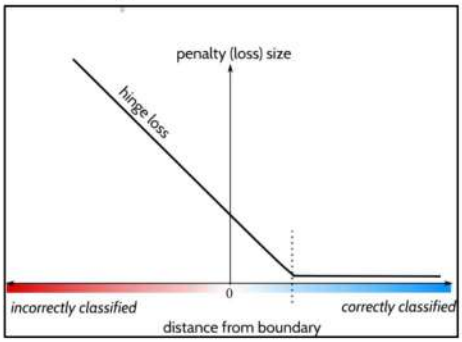
   $\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

   $\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

Quadratic optimization problem:

Minimize $\frac{1}{2}\mathbf{w}^T\mathbf{w}$

Subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ ← One constraint for each training point.



- The basic steps of the SVM are:

(1) Select two hyperplanes (in 2D) which separates the data with no points between them (red lines)

(2) Maximize their distance (the margin)

(3) The average line (the line half way between the two red lines) is the decision boundary

- Note:
  - To handle data that is NOT linearly separable → **kernel SVM**
  - We could map data to a higher dimension, and data there might become linearly separable.

5.2 Unsupervised learning

5.2.2 K-means

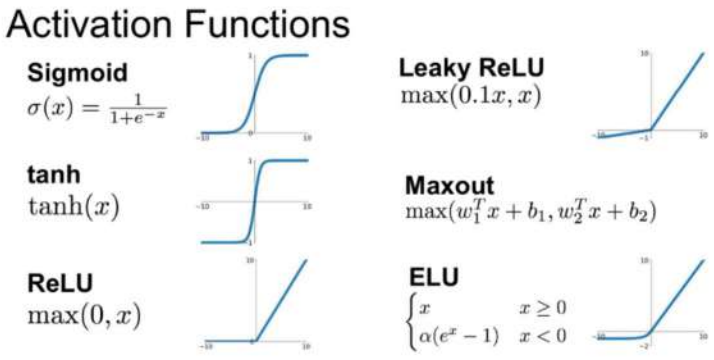
K-Means Clustering: Pseudo Code

1. Choose the number of clusters K and obtain the data points
2. Place the centroids c1, c2, ..., ck randomly
3. for each data point xi:
   - find the nearest centroid (c1, c2, ..., ck)
   - assign the point to that cluster
4. for each cluster j = 1, 2, ..., k
   - new centroid = mean of all points assigned to that cluster
5. Repeat steps 3 and 4 until convergence or until the end of a fixed number of iterations

5.3 Deep neural networks

5.3.1 Weights and layers

5.3.2 Activation functions



- ReLU is a good default choice for most problems

5.3.3 Regularization and normalization

5.3.4 Loss functions

5.3.5 Backpropagation

How to compute the backpropagation: See this example

https://hmkcode.com/ai/backpropagation-step-by-step/

5.4 Convolutional neural networks

Example: convolution

Given:

Input volume: 3 x 32 x 32

10 5x5 filters with stride 1, pad 2

Output volume size: 10 x 32 x 32

What is number of learnable parameters?

<Solution>

Parameters per filter (considering a bias term): 3 x 5 x 5 + 1 = 76

Now consider 10 filters.

total learnable parameters = 10 x 76 = 760.

Chapter 8 Image alignment and stitching

- A **panorama** is any wide-angle view of a physical space, whether in painting, drawing, photography, film, seismic images, or 3D modeling.
- **Image stitching** is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image.
- **Image normalization** is a typical process in image processing that changes the range of pixel intensity values.
  - output_channel = 255 * (input_channel - min) / (max-min)
- Jacobian, Hessian and Gradient?

Gradient	Vector of the 1st order derivatives of a scalar field	$(\nabla f)_i = \frac{\partial f}{\partial x_i}$
Jacobian	Matrix of gradients for components of a **vector** field	$J_{i.j} = \frac{\partial f_i}{\partial x_j}$
Hessian	Matrix of the 2nd order mixed partials of a **scalar** field	$H_{i.j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$

- The Jacobian matrix is a matrix composed of the first-order partial derivatives of a multivariable function

$$f(x_1, x_2, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

- For example, given $f(x, y) = (x^4 + 3y^2x\ ,\ 5y^2 - 2xy + 1)$, and
- Then, $J_f(x, y) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 4x^3 + 3y^2 & 6yx \\ -2y & 10y - 2x \end{pmatrix}$

➢ The Hessian matrix is an n×n square matrix composed of the second-order partial derivatives of a function of n variables.

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

- For example, given $f(x, y) = y^4 + x^3 + 3x^2 + 4y^2 - 4xy - 5y + 8$,
- Then, $H_f(x, y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 6x + 6 & -4 \\ -4 & 12y^2 + 8 \end{pmatrix}$

8.1 Pairwise alignment

8.1.1 2D alignment using least squares

➢ **Linear least squares (LLS)** estimation only works when the transform function is **linear** and doesn't deal well with **outliers**.

➢ General form of linear least squares $E_{lls} = \|Ax - b\|^2$

➢ Minimize the error (expansion): $E_{lls} = x^T(A^TA)x - 2x^T(A^Tb) + b^2$

➢ LLS solution: $\boldsymbol{A^TAx = A^Tb}$

Theorem. *Let A be an $m \times n$ matrix and let b be a vector in $\mathbf{R}^m$. The following are equivalent:*

1. *$Ax = b$ has a unique least-squares solution.*
2. *The columns of A are linearly independent.*
3. *A^TA is invertible.*

In this case, the least-squares solution is

$$x = (A^TA)^{-1}A^Tb$$

- 2D alignment using least squares: Problem Statement
  - Given a set of matched feature points {xi, xi'} and the form of the transformation f
  - Find the best estimate of p
  - Where x' = f(x, p)
  - Least-square error $E_{ls} = \sum_i \|f(x_i, p) - x_i'\|^2$ should be minimized.

8.1.3 Iterative algorithms

- Iterative algorithms: Gauss-Newton
- Note: nonlinear least squares curve-fitting problems
  - $E_{nls} = \sum_{i=1}^{m} \|\hat{y}(x_i, p) - y_i\|^2$
  - A set of data points (x_i, y_i) is given, and $\hat{y}$ is the estimated model, and p is a vector of m parameters.
  - $E_{nls} = \sum_{i=1}^{m} \|\hat{y}(x_i, p) - y_i\|^2 = (\hat{y} - y)^T(\hat{y} - y) = \hat{y}^T\hat{y} - 2\hat{y}^Ty + y^Ty$
- The **Gauss-Newton** Method
  - $\hat{y}(x_i, p + h) = \hat{y}(x_i, p) + \frac{\partial \hat{y}}{\partial p}h = \hat{y} + Jh$
  - $J^TJh = J^T(\hat{y} - y)$
- The **Levenberg-Marquardt** Method
  - $(J^TJ + \lambda I)h = J^T(\hat{y} - y)$

8.1.4 RANdom SAmple Consensus (RANSAC)

- **RANdom SAmple Consensus (RANSAC)** is a resampling technique that generates candidate solutions by using the minimum number of observations (data points) s required to estimate the underlying model parameters.
  - We assume that the parameters can be estimated from s data items out of total M data items.
  - Determine # of samples N:

$$N = \frac{\log(1 - P)}{\log\left(1 - (1 - e)^s\right)}$$

Where P is probability of a sample is inlier, e is proportion of outliers, s is minimum # of samples to fit the model.

- RANSAC

(1) Randomly select N data items.
(2) Estimate the parameters p
(3) Find some data items (out of M) to fit the model with p within a user given tolerance.
(4) If the fraction of fitting data items > threshold τ, then exit with success.
(5) Otherwise, repeat step 1~4 N times

8.1.5 3D alignment

- Example: orthogonal Procrustes algorithm → to solve singular value decomposition

8.2 Image stitching

8.2.3 Rotational panoramas

- When the camera undergoes a pure rotation.

8.2.4 Gap closing

- In the case of 360 degrees panoramas, accumulated error may lead to the presence of a gap between the two ends of the panorama.

8.2.6 Cylindrical and spherical coordinates

- Procedure of Cylindrical panorama

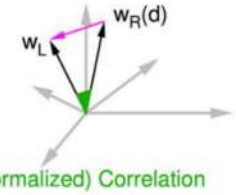
1. Take pictures on a tripod (or handheld)
2. Warp to cylindrical coordinate
3. Compute pairwise alignments
4. Fix up the end-to-end alignment
5. Blending
6. Crop the result and import into a viewer

9.1 Translational alignment

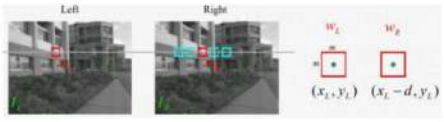
- translational alignment = direct alignment = pixel-based alignment
- To estimate the parameters of motion model for the purpose of alignment between a pair of images, we can apply
  (1) **direct** (pixel-based) approach or (2) **feature-based** approach.
- Direct methods: pixel-to-pixel matching
  (1) **Hierarchical** methods: image pyramids
    - Application: block matching
  (2) **Fourier-based** methods: Fourier transforms speeds up the computation
  (3) **Incremental** methods: Taylor series expansion
- **Template Matching**
  - For stereo pairs that have been rectified, error metrics, such as the **sum of squared differences** or **normalized cross-correlation**, can be used to directly compare the intensities in small patches around each feature point.

➢ Error Metrics

(Normalized) Sum of Squared Differences



(Normalized) Correlation



$\mathbf{w}_L$ and $\mathbf{w}_R$ are corresponding $m \times m$ windows of pixels

Define the window function, $\mathbf{W}_m(x, y)$, by

$$\mathbf{W}_m(x,y) = \left\{(u,v) \mid x - \frac{m}{2} \le u \le x + \frac{m}{2}, y - \frac{m}{2} \le v \le y + \frac{m}{2}\right\}$$

- Assume that W_L and $W_R(d)$ are normalized to unit length
- **sum of squared differences (SSD)**

$$E_{SSD}(d) = \|w_L - w_R(d)\|^2 = \sum_{(u,v)\in W_m(x,y)} [\hat{I}_L(u,v) - \hat{I}_R(u-d,v)]^2$$

- **Correlation**

$$E_{CC}(d) = w_L \cdot w_R(d) = \sum_{(u,v)\in W_m(x,y)} \hat{I}_L(u,v)\hat{I}_R(u-d,v) = cos\theta$$

➢ Error Metrics (~ similarity measure)

- Given a template image $T(X)$ sampled at a discrete pixel locations {Xi = (xi, yi)}, we wish to find where $T(X)$ is located in an image $I(X)$, where d = (u, v) is the displacement.
- **Sum of Absolute Difference (SAD)**

$$E_{SAD}(d) = \sum_i |I(X_i + d) - T(X_i)|$$

- **Sum of Squared Difference (SSD)**

$$E_{SSD}(d) = \sum_i [I(X_i + d) - T(X_i)]^2$$

- **Weighted Sum of Squared Difference (WSSD)**

$$E_{WSSD}(d) = \sum_i w_I(X_i) w_T(X_i + d)[I(X_i + d) - T(X_i)]^2$$

Where w_I and w_T are **spatially varying weights**; both are zero outside the image boundaries.

- **(Cross-) Correlation**

$$E_{CC}(d) = \sum_i I(X_i + d)T(X_i)$$

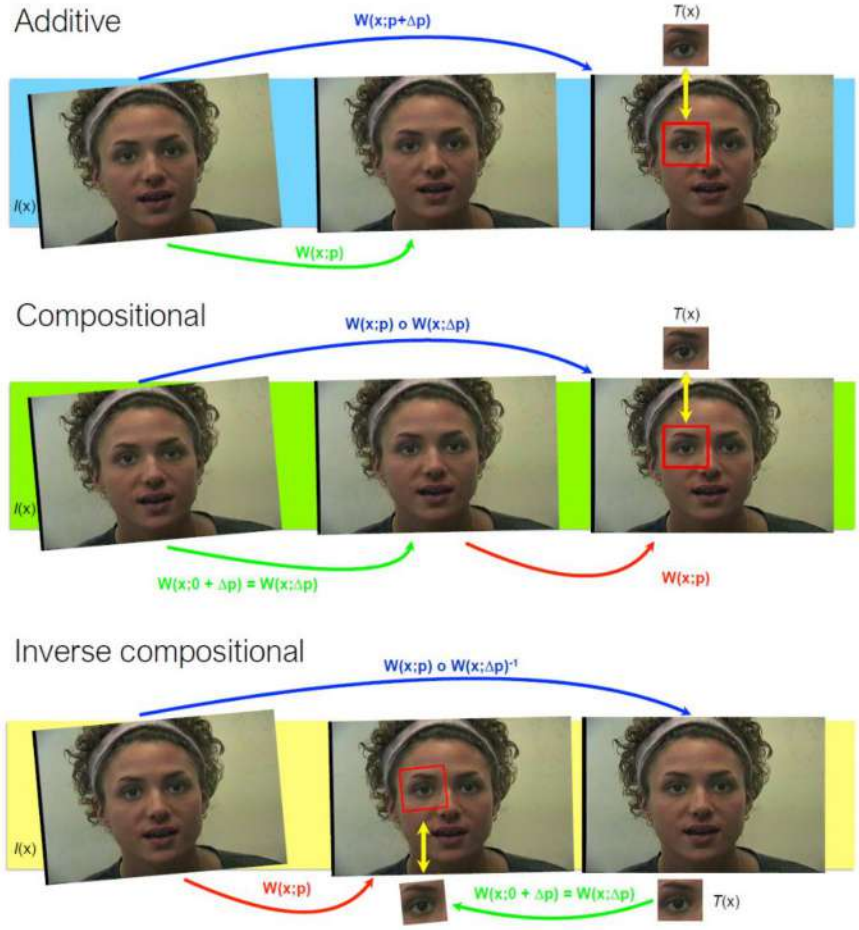
- **Normalized Cross-Correlation (NCC)**

$$E_{NCC}(d) = \frac{\sum_i \hat{I}(X_i + d)\hat{T}(X_i)}{\sqrt{\sum_i \hat{I}^2(X_i + d)\hat{T}^2(X_i)}}$$

Where $\hat{T}(X_i) = T(X_i) - \mu_T$, $\hat{I}(X_i + d) = I(X_i + d) - \mu_I$, μ_T and μ_I are the pixel averages of the patch. Note that $E_{NCC} \in [-\mathbf{1}, \mathbf{1}]$.

9.1.3 Incremental refinement

- Idea: Taylor series expansion



Alignment	Authors	Objective Function
(general form)		$\min_p \sum_X [I(W(X,p)) - T(X)]^2$
forward additive	Lucas-Kanade	$\min_p \sum_X [I(W(X,p+\Delta p)) - T(X)]^2$
forward compositional	Shum-Szeliski	$\min_p \sum_X [I(W(W(X,\Delta p),p)) - T(X)]^2$
inverse additive	Hager-Belhumeur	
inverse compositional	Baker-Matthews	$\min_p \sum_X [T(W(X,p)) - I(W(X,p))]^2$

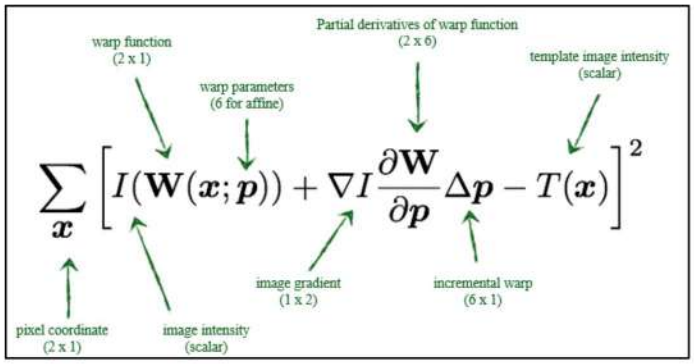
Alignment	Lucas-Kanade	Shum-Szeliski	Baker-Matthews
Step 1. Warp the image $\boldsymbol{I'}$	$I' = I\big(W(X,p)\big)$	$I' = I\big(W(X,p)\big)$	$I' = I\big(W(X,p)\big)$
Step 2. Compute the gradient $\boldsymbol{g}$	$g = \nabla I(x')$	$g = \nabla I(x')$	$g = \boldsymbol{\nabla T(W)}$
Step 3. Compute the Jacobian $\boldsymbol{J}$	$J = \frac{\partial W(x,p)}{\partial p}$	$J = \frac{\partial W(x,\mathbf{0})}{\partial p}$	$J = \frac{\partial W(x,p)}{\partial p}$
Step 4. Compute the Hessian $\boldsymbol{H}$	$H = \sum_X [gJ]^T [gJ]$	$H = \sum_X [gJ]^T [gJ]$	$H = \sum_X [gJ]^T [gJ]$
Step 5. Compute the increment $\boldsymbol{\Delta p}$	$\Delta p = \sum_X H^{-1}[gJ]^T [T(X) - I']$	$\Delta p = \sum_X H^{-1}[gJ]^T [T(X) - I']$	$\Delta p = \sum_X H^{-1}[gJ]^T [T(X) - I']$
Step 6. Update the parameters	$p \leftarrow p + \Delta p$	$W(X,p) \leftarrow W(X,p) \circ W(X,\Delta p)$	$W(X,p) \leftarrow W(X,p) \circ W^{-1}(X,\Delta p)$

- Lucas-Kanade Alignment (Additive Alignment)
  - direct methods of translational alignment, incremental refinement
  - Objective function

$$\min_p \sum_X [I(W(X,p)) - T(X)]^2$$

Where W is the warped image.

  - idea: solve for the increment Δp in $\min_p \sum_X [I(W(X,p+\Delta p)) - T(X)]^2$
  - Taylor series expansion: $\sum_X [I\big(W(X,p)\big) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(X)]^2$



  - Solution $\Delta p = H^{-1} \sum_X [\nabla I \frac{\partial W}{\partial p}]^T [T(X) - I\big(W(X,p)\big)]$

Where Hessian matrix $H = \sum_X [\nabla I \frac{\partial W}{\partial p}]^T \nabla I \frac{\partial W}{\partial p}$.

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} W_x(x,y) \\ W_y(x,y) \end{bmatrix}$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}$$

Affine Transform

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} p_1 x + p_3 y + p_5 \\ p_2 x + p_4 y + p_6 \end{bmatrix}$$

$$\frac{\partial W_x}{\partial p_1} = x \quad \frac{\partial W_x}{\partial p_2} = 0 \quad \cdots$$

$$\frac{\partial W_y}{\partial p_1} = 0 \quad \cdots$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

- Pseudo Code of Lucas-Kanade Alignment

Step 1: Warp the image $I\big(W(X,p)\big)$

Step 2: Compute the error image $T(X) - I\big(W(X,p)\big)$

Step 3: Compute the gradient of the warped image $\nabla I(x')$

Step 4: Evaluate the Jacobian $\frac{\partial W}{\partial p}$

Step 5: Compute the Hessian $H = \sum_X [\nabla I \frac{\partial W}{\partial p}]^T [\nabla I \frac{\partial W}{\partial p}]$

Step 6: Compute the increment $\Delta p = H^{-1} \sum_X [\nabla I \frac{\partial W}{\partial p}]^T [T(X) - I\big(W(X,p)\big)]$

Step 7: Update the parameters $p \leftarrow p + \Delta p$

- Shum-Szeliski Alignment (Compositional Alignment)
  - Idea: solve for the increment Δp in $\min_p \sum_X [I(W(W(X,\Delta p),p)) - T(X)]^2$

    Where $W(W(X,\Delta p),p) = W(x,p) \circ W(x,\Delta p)$.

Step 1: Warp the image $I\big(W(X,p)\big)$

Step 2: Compute the error image $T(X) - I\big(W(X,p)\big)$

Step 3: Compute the gradient of the warped image $\nabla I(x')$

Step 4: Evaluate the Jacobian $\frac{\partial W(x,0)}{\partial p}$

Step 5: Compute the Hessian $H = \sum_X [\nabla I \frac{\partial W}{\partial p}]^T [\nabla I \frac{\partial W}{\partial p}]$

Step 6: Compute the increment $\Delta p = H^{-1} \sum_X [\nabla I \frac{\partial W}{\partial p}]^T [T(X) - I\big(W(X,p)\big)]$

Step 7: Update the parameters $W(X,p) \leftarrow W(X,p) \circ W(X,\Delta p)$

- Baker-Matthews Alignment (Inverse Compositional Alignment)

Step 1: Warp the image $I\big(W(X,p)\big)$

Step 2: Compute the error image $T(X) - I\big(W(X,p)\big)$

Step 3: Compute the gradient $\nabla \mathrm{T}(W)$

Step 4: Evaluate the Jacobian $\frac{\partial W}{\partial p}$

Step 5: Compute the Hessian $H = \sum_X [\nabla \mathrm{T} \frac{\partial W}{\partial p}]^T [\nabla \mathrm{T} \frac{\partial W}{\partial p}]$

Step 6: Compute the increment $\Delta p = \sum_X H^{-1} [\nabla T \frac{\partial W}{\partial p}]^T [T(X) - I\big(W(X,p)\big)]$

Step 7: Update the parameters $W(X,p) \leftarrow W(X,p) \circ W^{-1}(X, \Delta p)$

8.3 Global alignment

8.3.1 Bundle adjustment

- **Bundle adjustment (BA)** is a non-linear squares approach involving adjusting the bundle of rays between each camera pose and the set of 3D points. The purpose of BA is to estimate camera poses and 3D points in order to minimize the re-projection error.
- BA: key ideas
  (1) Start with an initial guess.
  (2) Project the estimated 3D points into the estimated camera images.
  (3) Compare locations of the projected 3D points with measured 2D points.
  (4) minimize the **re-projection error** in the images.

8.3.2 Parallax removal

- The **parallax** refers to the apparent change in relative position of stationary objects caused by a change in viewing position.
- For example, **motion parallax** refers to the fact that objects moving at a constant speed across the frame will appear to move a greater amount if they are closer to an observer (or camera) than they would if they were at a greater distance.

8.4 Compositing

- **Compositing** (= **Photomontage**) is the process through which two or more images combine to make the appearance of a single picture.
- **Ghosts** usually occur in the process of image blending; a moving object may appear in an adjacent stitched image.

8.4.1 Choosing a compositing surface

- Commonly-used choice for compositing large panoramas: cylindrical or spherical projection.

8.4.2 Pixel selection and weighting (deghosting)

- Bad image registration is due to:
  - visible seams (∵exposure differences)
  - blurring (∵mis-registration)
  - ghost (∵moving objects)

Chapter 9 Motion estimation

9.3 Optical flow

- Optical Flow: Problem Statement
  - Given two consecutive image frames, estimate the motion of each pixel.
  - Assumptions: **brightness constancy**, **small motion**
- Brightness constancy (for intensity images)
  - The brightness of the point remains the same.
  - pixel-to-pixel comparison (no image features) is allowed.
- Small motion
  - Assume small time step.
  - the intensity function (brightness constancy constraint) can be linearized.
- Idea: Look for nearby pixels (small motion) with the same color (brightness constancy).
- Constraint

$$\boldsymbol{I_x u + I_y v + I_t = 0}$$

Or, $\underset{(1\times 2)}{\nabla I^{\top}} \underset{(2\times 1)}{\boldsymbol{v}} + I_t = 0$, where

the spatial derivatives $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$ are image gradients (at a point p);

the optical flow $u = \frac{\partial x}{\partial t}$ and $v = \frac{\partial y}{\partial t}$ are flow velocities;

and the temporal derivatives $I_t = \frac{\partial I}{\partial t}$ is temporal gradient

- How to compute?
  - $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$ → Sobel filter, DoG filter
  - $I_t = \frac{\partial I}{\partial t}$ → frame differencing
  - $u = \frac{\partial x}{\partial t}$ and $v = \frac{\partial y}{\partial t}$ are unkown

➢ Algorithms

Lucas-Kanade (L-K) Optical Flow	Horn-Schunck (H-S) Optical Flow
Constant Flow: Flow in **constant** in a surrounding patch (e.g., a 5x5 patch). → define a **window size**	Smooth Flow: Flow varies from pixel to pixel → define a **regularization term λ**
local method → sparse → To Interpolate optical flow for all pixels (**post-processing**)	**global** method → **dense** (no post-processing)

➢ **Lucas-Kanade (L-K) Optical Flow**

- L-K computes the optical flow (u,v) to solve the least squares problem.

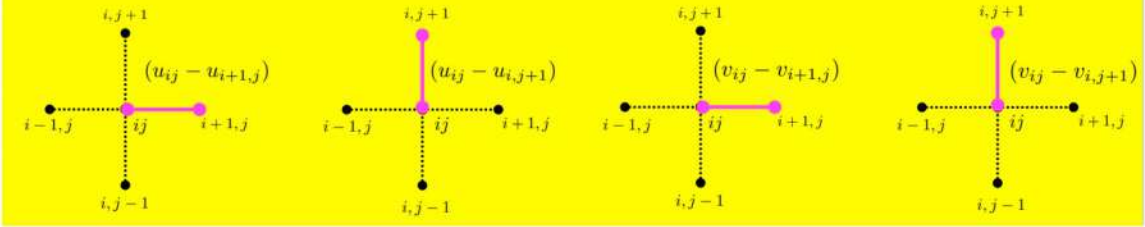
$$\underset{A^\top A}{\begin{bmatrix} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{bmatrix}} \underset{\hat{x}}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{A^\top b}{\begin{bmatrix} \sum_{p\in P} I_x I_t \\ \sum_{p\in P} I_y I_t \end{bmatrix}}$$

- A^TA is also used in Harris corner detection

➢ **Horn-Schunck (H-S) Optical Flow**

- Idea: Assume brightness constancy and smooth flow field
- The energy should be minimized **E = Ed + λ Es**
- Ed is the **data penalty** to measure the distance between the function f and a set of data points di = d(xi, xj), and Ed =$\sum_{i,j}[\, f(x_i, x_j) - d(i,j)\,]^2$.
  Here, Ed is the **bright constancy**, and $E_d(i,j) = [\, I_x u_{ij} + I_y v_{ij} + I_t\,]^2$.
- Es is the **smoothness penalty**, and
  $E_s(d) = \sum_{i,j}[\, \rho(d(i,\ j) - d(i+1,\ j)) + \rho(d(i,\ j) - d(i,\ j+1))\,]$
  where ρ is some monotonically increasing function.
- Overall, the objective function is $\min_{(u,v)} \sum_{i,j} E_d(i,j) + \lambda E_s(i,j)$.

$$E_s(i,j) = \frac{1}{4}\left[(u_{ij} - u_{i+1,j})^2 + (u_{ij} - u_{i,j+1})^2 + (v_{ij} - v_{i+1,j})^2 + (v_{ij} - v_{i,j+1})^2\right]$$



Where local average is defined as

$$\bar{u}_{ij} = \frac{1}{4}\left\{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right\}$$

- Laplacian of u and v

$$12\begin{pmatrix} \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \\ \frac{1}{6} & -1 & \frac{1}{6} \\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12} \end{pmatrix}$$

- H-S Pseudo Code

Step 1: compute the gradients I_x, I_y, and I_t.

Step 2: Initialize the flow $u_k = v_k = 0$ at each pixel (index k means the pixel at (i,j)).

Step 3: Update the flow at each pixel until convergence.

$$u_k \leftarrow \bar{u}_k - \frac{I_x\bar{u}_k + I_y\bar{v}_k + I_t}{\lambda + {I_x}^2 + {I_y}^2} I_x$$

$$v_k \leftarrow \bar{v}_k - \frac{I_x\bar{u}_k + I_y\bar{v}_k + I_t}{\lambda + {I_x}^2 + {I_y}^2} I_y$$

9.2 Parametric motion

9.2.2 Spline-based motion

- Spline-based motion for image registration is an algorithm based on **spline** representations of the **displacement field** d(x,y) = [dx, dy]^T.
- The motion field is represented as a 2D **spline** controlled by a smaller number of **control vertices** c(x,y) = [cx, cy]^T.
- The displacement at a pixel (x,y) is

$$d_i(x,y) = \sum_j c_j B_j(x,y)$$

where $B_j(x,y)$ is the basis function.

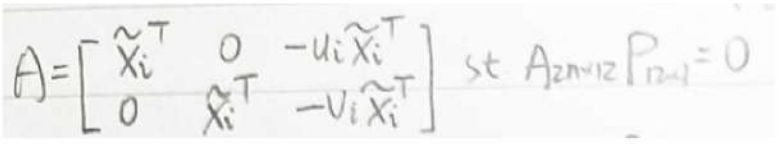
9.4 Layered motion

- To represent **moving** images with sets of **overlapping layers**; and layers are **ordered in depth** and occlude each other.

Chapter 11 Structure from motion and SLAM

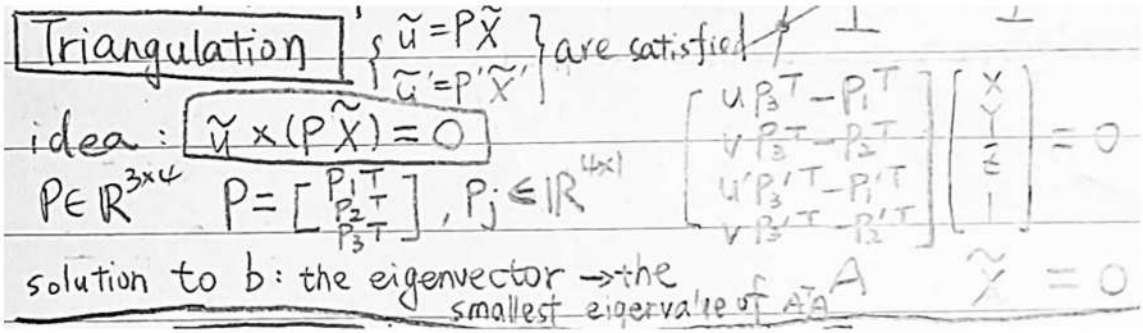
11.2 Pose estimation

11.2.1 Linear algorithms

- Direct Linear Transformation (DLT)
  - Given 2D-3D correspondences {ui, Xi}, n ≥ 6, Find P_{3x4}
  - Solution
  - 
  - $A = USV^T, V = [V_1, V_1, \dots, V_9]^T$
  - $V_9 \rightarrow P_{3x4}$
  - $P = K[R|t] = [M|Kt]$, N=$M^{-1} = (KR)^{-1}$
  - QR decomposition of N

11.2.4 Triangulation

- The problem of determining the position of a 3D point from a set of corresponding image locations and known camera positions is known as **triangulation**.
- Given 2D-2D correspondences {xi, xi'} and camera projection matrices {P, P'}, try to estimate the 3D point X.



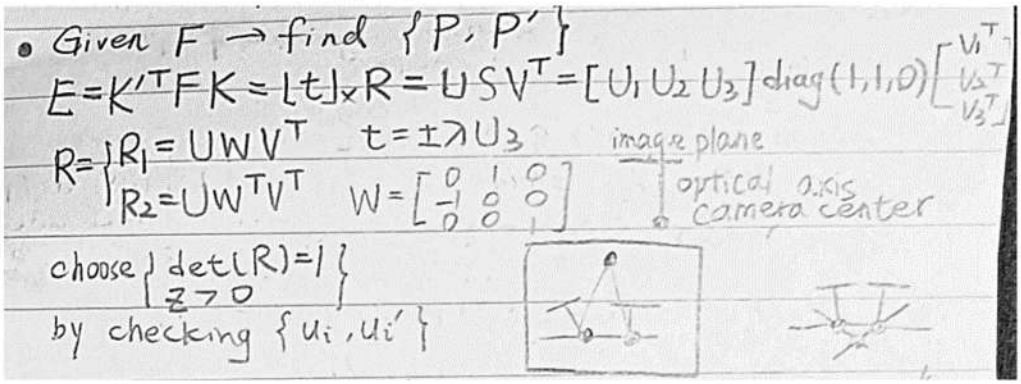
11.3 Structure from motion (SfM)

- **Structure from motion (SfM)** is the process of estimating the 3D structure of a scene from a set of 2D images.
- Structure from Motion (SfM)
  - reconstruction **ambiguity**: estimation up to an unknown scale factor
  - Problem: Affine SfM ambiguity → Solution: **factorization**
  - Problem: Projective SfM ambiguity → Solution: **bundle adjustment**
- Epipolar constraint:
  - $\boldsymbol{x'^T E x = 0}$
  - $\boldsymbol{\tilde{u}'^T F \tilde{u} = 0}$

- Procedure of two-frame SfM

  Given 2D-2D correspondences {xi, xi'}.

  - Step 1: (**8-point algorithm**) Find the fundamental matrix F
  - Step 2: Find the camera projection matrices P and P'
    - ✧ P = [I | 0]
    - ✧ P' = [-[e]$_x$F | e], where e is the epipole satisfying $F^T e = 0$
  - Step 3: Triangularization: DLT with $\tilde{u} = P\tilde{\mathrm{X}}$ and $\tilde{u}' = P'\tilde{\mathrm{X}}'$
- Determine the motion from the essential matrix
  - Given F → Find {P, P'}



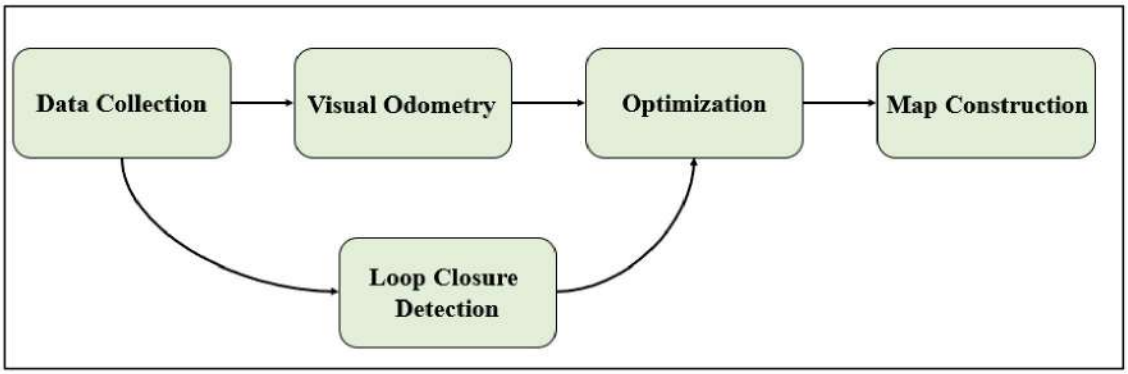
11.3.5 Application: View morphing

- **Morphing** is a geometric **interpolation** technique to change (or morph) one image into another through a seamless transition.
- **image morphing** = feature correspondences + linear interpolation
  - no shape is preserved
- **view morphing** = image pre-warping + image morphing + image post-warping
  - view morphing is a **shape**-preserved morph
- Pseudocode of view morphing
  - Step 1: **Pre-warp** first and last images to parallel views
  - Step 2: **image morph** between pre-warped images
  - Step 3: **Post-warp** to interpolated view

11.5 Simultaneous localization and mapping (SLAM)

- Simultaneous Localization and Mapping – or SLAM – is a method used to create a **map** of an environment while **localizing** a moving mobile robot in the environment at the same time.

- SLAM Flowchart



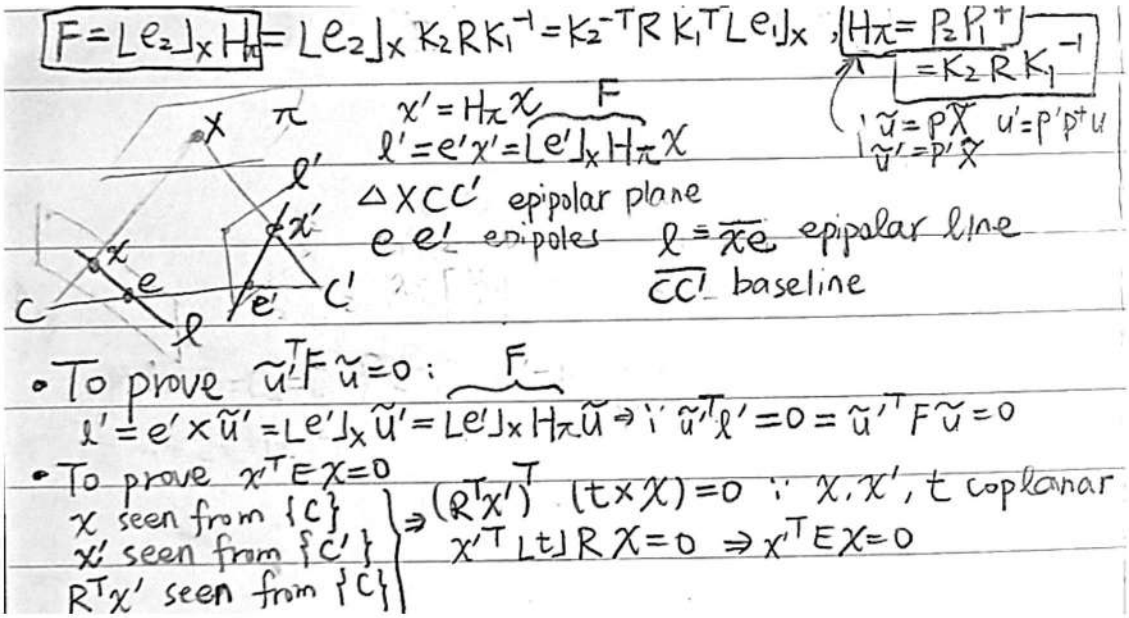
Laser	Camera
• Precise	• High computational complexity
• Low latency	• Environment dependency (night)
• Low computational complexity	• Sensitive with noise
• High power and huge size	• Cheap
• Expensive	• Light

- **Visual odometry** is the process of determining the position and orientation of a robot by analyzing the associated camera images.
- **Loop closure** is a sub algorithm of SLAM that is about identifying **previously visited** locations and using them to correct the accumulated errors in the robot's pose estimation.
- **TF-IDF**, short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a corpus.
- **TF-IDF** weights among images and calculates the **cosine similarity** to enable loop closing detection.

Chapter 12 Depth estimation

12.1 Epipolar geometry

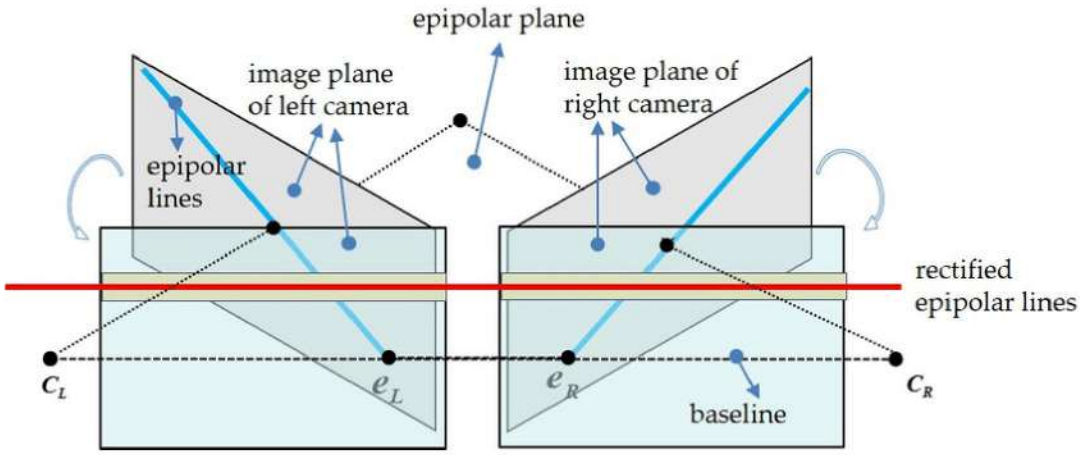
- The **epipolar geometry** describes the geometric relationship between **two** perspective views of the same 3D scene.
- "**Epipolar constraint**" states that if a point x is observed by one camera, its corresponding point x' observed by the other camera must lie on a line (epipolar line). This constraint reduces the search space of correspondences from two dimensions to one dimension.
- Define the epipolar plane, 2 epipoles, 2 epipolar lines, a baseline



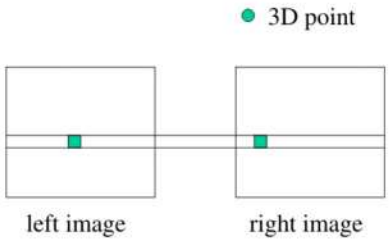
- Why only 7 degrees of freedom (DoFs) are present in the fundamental matrix?
  - There are 9 parameters in the fundamental matrix F.
  - We lose 1 DoF because we are using **homogeneous** coordinates. For example, the pixel coordinate is represented as $[u, v, 1]^T$.
  - We lose another DoF because det(F) = 0.

12.1.1 Rectification

- **Stereo matching** is the task of estimating a 3D model of a scene from two or more images.
- **Image rectification** is a transformation process used to project images onto a common image plane.



- The (stereo) **disparity** is the difference in image location of the same 3D point when projected under perspective to two different cameras, i.e., Disparity is the horizontal displacement of a point's projections between the left and the right image.



- For a simple stereo system, the depth of a point (z) is given by

$$z = \frac{fb}{d}$$

where f is the focal length, b is the baseline (or the distance between the cameras), and d the disparity between corresponding points.

12.1.2 Plane sweep

- The **Plane Sweep** Algorithm is a popular technique used in multi-view stereo for reconstructing 3D scenes from a set of 2D images.
  - Step 1: Define a plane (reference coordinate) π to be swept.
  - Step 2: Warp the i-th image Ii onto π
  - Step 3: Triangulate the 2D correspondences on π → get depth information
  - Step 4: reconstruct the 3D scene

12.2 Sparse correspondence

- **sparse correspondence** deals with a few keypoints in an image.
- **dense correspondence** tries to map all of the parts of an image to another related image.

12.3 Dense correspondence

- Correspondence problems include (1) **stereo** and (2) optical flow

motion estimation	optical flow	stereo matching (rectification)
search space	2D translation	1D translation (along the horizontal epipolar line)
motion to be considered	object motion	object depth

- **stereo** is simpler since the search for correspondences is restricted to 1D epipolar lines (versus 2D search for non-rigid motion).
- Dense correspondence problem from stereo vision is solved by 4 steps:
  - Step 1: Cost computation
  - Step 2: Cost aggregation
  - Step 3: Disparity computation and optimization
  - Step 4: Disparity refinement

12.3.1 Similarity measures

- How to measure the dense correspondences? → **similarity**.
- If illumination varies, we can apply the following filters to pre-processing images to improve their similarity in dense correspondences.
  (1) Mean filter
  (2) LoG filter
  (3) BilSub filter (BilSub = bilateral background subtraction)
  (4) Rank filter
  (5) SoftRank filter

12.4 Local methods

- Local methods → cost aggregation
- Local methods for stereo problems which are window-based
  (1) fixed windows
  (2) shiftable windows
  (3) variable windows

12.5 Global optimization

- **energy-based** method for optimization
  - The global energy should be minimized **E = Ed + λ Es**
  - Ed is the **data penalty** to measure the distance between the function f and a set of data points di = d(xi, xj), and Ed = $\sum_{i,j}[\, f(x_i, x_j) - d(i,j)\,]^2$
  - Es is the **smoothness penalty**, and
    $E_s(d) = \sum_{i,j}[\, \rho(d(i,\ j) - d(i+1,\ j)) + \rho(d(i,\ j) - d(i,\ j+1))\,]$
    where ρ is some monotonically increasing function.

- Optimization for dense correspondence problem can be solved by:

12.5.1 Dynamic programming

12.5.2 Segmentation-based techniques

12.5.3 Z-keying and background replacement

12.6 Deep neural networks (DNNs)

- DNN-based stereo correspondence algorithms
  (1) Learning in the stereo pipeline
  (2) End-to-end learning with 2D architectures
  (3) End-to-end learning with 3D architectures

12.7 Multi-view stereo

- Multi-view stereo algorithms
  (1) voxel coloring
  (2) level set methods
  (3) graph cuts
  (4) shape from silhouettes (e.g., visual hull)

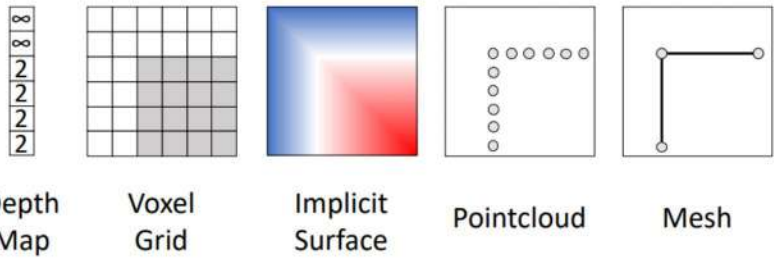
12.7.3 Shape from silhouettes

- A **silhouette** is a solid, dark image of a subject against a brighter background.
  - Shape from silhouettes: e.g., **visual hull, octree**
- **Visual hulls** can be defined as maximal silhouette-equivalent of an object with respect to a set of viewing regions.
- An **octree** is a tree data structure where each internal node has **8** children. Octrees are commonly used for spatial partitioning of 3D point clouds.

Chapter 13 3D reconstruction

- **Tomography** (斷層掃描) (or **tomographic imaging**) is imaging by sections or **sectioning** that uses any kind of **penetrating wave**.
- A **computed tomography (CT)** scan combines a series of **X-ray** images taken from different angles around your body and uses computer processing to create **cross-sectional** images (slices) of the bones, blood vessels and soft tissues inside your body.
- **Holography**, often known as **hologram** technology, is a type of photography that records the light emitted by an object and then projects it as a 3D object that can be seen without the use of any extra equipment.
- **Teleportation** (瞬間移動) is the hypothetical transfer of matter or energy from one point to another without traversing the physical space between them.
- **Holoporation** = hologram + teleportation
  - Microsoft Holoportation is a 3D capture technology that allows 3D models of people to be reconstructed, compressed and transmitted anywhere in the world in real time.

- 3D Shape Representations



13.1 Shape from X

- **Foreshortening** refers to the visual effect or optical illusion that an object or distance appears shorter than it actually is because it is angled toward the viewer.
- **Perspective** is a technique for drawing a 3D object on a 2D piece of paper. **Foreshortening** is a method of showing perspective by shortening the object.

- **Photometric stereo** is a technique in computer vision for estimating the surface **normals** of objects by observing that object under different lighting conditions. The special case where the data is a **single** image is known as **shape from shading**.
- **Shape from Shading** is the process of computing the 3D shape of a surface from one image of that surface **normal**.
  - As surface normal changes, brightness also changes.
- **Shape from texture** is the process by which the 3D structure of a surface is determined from the spatial distribution of surface markings.
  - Texture is the **repeated** appearance across local neighborhoods of a surface.
  - We need to estimate the relative **foreshortening**.
- **Shape from focus** (= **Depth from focus/defocus**) is a method for estimating the 3D surface of a scene from a set of two or more images of that scene based on information about the **focus** of an optical system.
  - depth from defocus:
    The depth information is estimated based on the amount of **blur** of the considered object. → **more blur** causes more **larger** focus distances.
  - depth from focus:
    The distance to the camera is estimated by the sharpness of an object by comparing multiple images with different focus distances.

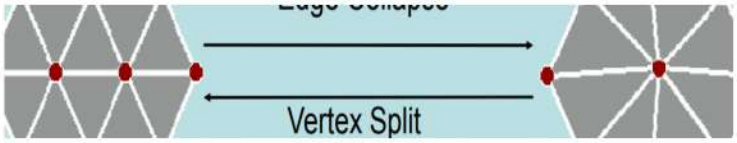
13.2 3D scanning

13.2.1 Range data merging

- **Iterative closest point (ICP)** is an algorithm employed to minimize the difference between two clouds of points.

13.3 Surface representations

- Surface representations
  - Triangle Meshes
  - Splines
  - Subdivision Surfaces
- Sparse data
  - To implement surface interpolation → **scattered data interpolation** or **radial basis function**
- Dense data
  - To implement surface simplification → **edge collapse** operations
  - An edge can be collapsed by removing it, merging its adjacent vertices to one, and reconnecting the adjacent edges to the merged vertex.



13.5 Volumetric representations

13.5.1 Implicit surfaces and level sets

- **Explicit Surface** Representations, e.g., a list of points or triangles
- **Implicit Surface** Representations: a **level set of a function** defined over the space in which the geometry is embedded
- The implicit surfaces use an **indicator function** $f(x,y,z)$ to indicate whether a 3D point is:
  - inside the surface ($f < 0$) or
  - outside the surface ($f > 0$).
- **Signed Distance Function (SDF)** gives us a distance of point X from the boundary of a surface.

13.6 Model-based reconstruction

- Model-based reconstruction considers geometric **primitives**.

13.6.2 Facial modeling and tracking

- 3D head model applications:
  - head tracking
  - face transfer, i.e., replacing one person's face with another in a video
  - face beautification by warping face images toward a more attractive "standard"
  - face de-identification for privacy protection
  - face swapping

13.7 Recovering texture maps and albedos

- **Albedo** is defined as the fraction of incident radiation that is reflected by a surface.
- **Albedo** is the property of a specific material, whereas **reflectance** is not specific to any material.

13.7.1 Estimating BRDFs

- The **bidirectional reflectance distribution function (BRDF)** is a function that defines **how light is reflected at an opaque surface** and is a function of illumination geometry and viewing geometry.