

Introduction

2021年9月8日 13:11

Computer vision: enables computers to process and interpret visual data

- Image/video
- Sensing device
- Interpreting device (interpretation)

CV problems

- Computing properties of the 3D world from visual data (**measurement**)
 - Ill-posed
 - Impossible to invert the image formation process
- Algorithms and representations to allow a machine to recognize objects, people, scenes and activities (**perception and interpretation**)
 - Computationally intensive/expensive
 - Do not fully understand the processing mechanisms involved
- Algorithms to mine, search, and interact with visual data (**search and organization**)
 - Scale is enormous, explosion of visual content
- Algorithms for manipulation or creation of image or video content (**visual imagination**)

Challenges: lighting, scale, deformation, occlusions, background clutter, local ambiguity and context, motion, object inter-class variation

Current techs

- Optical character recognition (OCR): convert scanned documents to text
- Face detection
- Smile detection
- Face recognition
- Vision for biometrics
- Object recognition
- 3D urban modeling and virtual tourism
- Visual special effects (VFX): shape and motion capture
- Sport vision
- Automotive safety and smart cars
- Interactive games
- Vision for robotics, space exploration
 - Panorama stitching
 - 3D terrain modeling
 - Obstacle detection, position tracking
- Medical imaging
- Captioning and visual question answering

Image formation

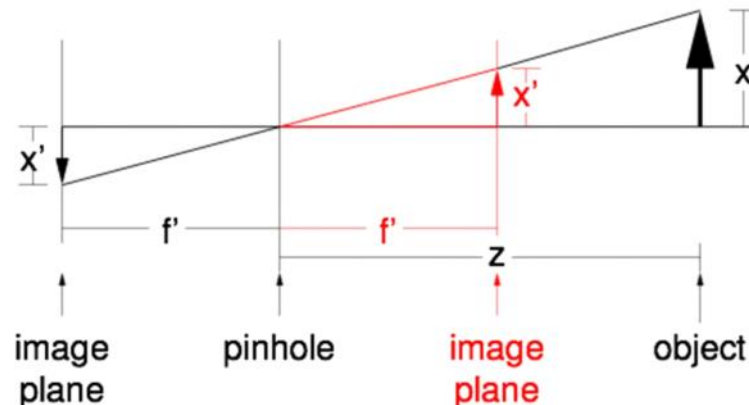
2021年9月10日 12:59

Graphics takes a model and turn it into images

Vision does the inverse: analysis and synthesis

Image formation: process that produces a particular image depends on

- Lighting conditions
- Scene geometry
- Surface properties
 - Textures
 - **Reflection:**
 - Depends on the viewing and illumination (light) direction
 - BRDF: Bidirectional Reflection Distribution Function
 - Lambertian surface: $BRDF = \frac{\rho d}{\pi}$, where ρd is constant called albedo.
 - A surface that looks the same from any direction (points are the same brightness)
 - Mirror surface: all incident light reflected in one direction
- Camera optics
 - **Pinhole** camera:
 - A box with a small hole (aperture) in it
 - Each scene point contributes to **only one sensor pixel**
 - Image is upside down



- Changing the focal length changes the size of the resulting image

- Sensor properties
 - Captures the amount of light reflected from the object
 - **Bare-sensor** imaging:
 - All scene points contribute to all sensor pixels

Perspective effects

- **Far objects appear smaller**
 - Size is inversely proportional to distance
- Parallel lines meet at a point (**vanishing point**)
 - Each set of parallel lines meets at a different point
 - Sets of parallel lines on the same plane lead to **collinear** vanishing points (**horizon** for the plane)
 - Vanishing points must be on the same horizon line
 - Can have multiple point perspective
- Good way to spot fake images
 - Scale and perspective do not work
 - Vanishing points behave badly

Properties of projection

- Points project to points
- Lines project to lines
- Planes project to the whole or half image
- **Angles are not preserved**
- Incidences (intersections) are preserved
- **Degeneration**
 - Line through focal point projects to a point
 - Plane through focal point projects to a line

Perspective projection

- A 3D object point $P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ projects to 2D image point $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$
 - $x' = f' \frac{x}{z}$.
 - $y' = f' \frac{y}{z}$.
 - This assumes world coordinate frame at the optical center (pinhole) and aligned with the image plane, image coordinate frame aligned with the camera coordinate frame
- Camera matrix
 - $C = \begin{pmatrix} f'_x & 0 & 0 & c_x \\ 0 & f'_y & 0 & c_y \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbb{R}^4$.
 - If pixels are squared/lens is perfectly symmetric, then $f'_x = f'_y$.
 - If sensor and pinhole perfectly aligned, then $c_x = c_y$.
 - If coordinate system centered at the pinhole, then $\mathbb{R}^4 = I^4$.
 - $P = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$.
 - $P' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = CP$.
- Camera calibration is the process of estimating parameters of the camera matrix based on set of 3D-2D correspondences
 - Usually requires a pattern whose structure and size is known

Weak perspective

- $P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ in a plane at $z = z_0$ projected to $P' = \begin{pmatrix} x' \\ y' \end{pmatrix}$ where $x' = mx, y' = my, m = \frac{f'}{z_0}$.

Orthographic projection

- $x' = x, y' = y$.

Projection pros and cons

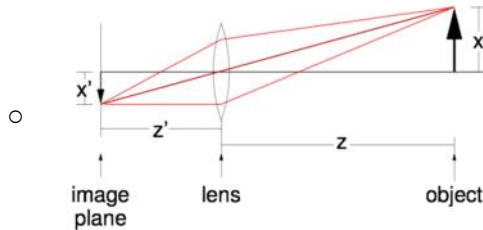
- Weak perspective
 - Accurate when object is small and/or distant
 - Useful for recognition
 - Simpler math
- Perspective
 - Accurate for real scenes
- When maximum accuracy is required, it is necessary to model additional details of a particular camera
 - Use perspective projection with additional parameters

Snell's law

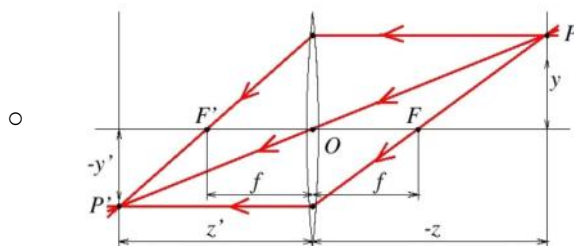
- $n_1 \sin \alpha_1 = n_2 \sin \alpha_2$.
- Light rays in particular homogeneous medium travel in straight lines
- At the interface between medium, they either reflect or refract

Lens:

- Capture more light while preserving the abstraction of an ideal pinhole camera
- Pinhole model with lens
 - Light bends twice at both interfaces of the lens



- Thin lens: a lens is considered thin if its thickness (t) is much less than the radii of curvature of its surfaces (R_1 and R_2)
 - Can assume that light bends only once at the center of the lens
 - $\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}$



- If y' and z are positive, we will have a plus sign.
- z' tells where to place the image plane to image objects a specified distance away.
- Solving f is used in auto-focus.
- z is the depth from focus.
- Points at **different depths**
 - Only certain points will be in complete focus
 - Others will result in circle of confusion and ultimately blur
- Focal length
 - The incoming rays, parallel to the optical axis, converge to a single point a distance f behind the lens.
 - This is where we want to place the image plane
 - For thin lens, when $z \rightarrow \infty$, $z' = f$
- Out of focus
 - The image plane is slightly closer/farther than the focal length
 - Creates **circle of confusion**
- Spherical aberration
 - Rays that strike closer to the edge of the lens will generally focus closer
 - **Circle of least confusion**: the smallest spot created by the lens when imaging a point source
- Compound lens system:
 - Aberrations can be minimized by aligning several simpler lenses
 - Can be still modeled by thick lens equation
- Vignetting
 - Some parts of the beam never reaches the second lens
- **Lens effects**
 - Chromatic aberration
 - **Index of refraction depends on wavelength**, λ , of light.
 - Light of different colors follows different paths

- Not all colors can be in equal focus
- Scattering at the lens surface
 - Some light is reflected at each lens surface
- Other geometric phenomena/distortions
 - Pincushion distortion
 - Barrel distortion

Human eye

- Iris: like a camera
- Pupil: pinhole/aperture
- Retina: film/digital sensor
 - Contains light receptors:
 - Rods:
 - Not involved in color vision
 - Grey-scale vision only
 - Operate at night
 - Highly sensitive, can respond to a single photon
 - Yield relatively poor spatial detail
 - Better motion sensitivity
 - Cones:
 - Color vision
 - Operate in bright light
 - Less sensitive
 - Yield higher resolution
- Focusing is done by changing shape of lens
- When the eye is properly focused, light from an object outside the eye is imaged on the retina
- Blind spot: where the nerves from the retina exit the eye, collected into the optic nerve, there are no receptors

Image filter

September 15, 2021 12:48 PM

Image as a 2D function

- A grey scale image is a 2D function
 - Domain: $(X, Y) \in ([1, width], [1, height])$
 - Range: $I(X, Y) \in [0, 255] \subset \mathbb{Z}$
- Addition (average): $\frac{I(X, Y)}{2} + \frac{G(X, Y)}{2}$
 - Note: $\frac{I(X, Y)}{2} + \frac{G(X, Y)}{2} \geq \frac{I(X, Y) + G(X, Y)}{2}$.
 - This is because of overflow of $I(X, Y) + G(X, Y)$.
 - Convenient to convert images to doubles when doing processing

Warping: changes domain of image function

Filtering: changes range of image function

- **Point operation** (point processing)
 - Darken: $I(X, Y) - 128$
 - Lower contrast: $\frac{I(X, Y)}{2}$
 - Non-linear lower contrast: $\left(\frac{I(X, Y)}{255}\right)^{1/3} \times 255$
 - Invert: $255 - I(X, Y)$
 - Lighten: $I(X, Y) + 128$
 - Raise contrast: $I(X, Y) \times 2$
 - Non-linear raise contrast: $\left(\frac{I(X, Y)}{255}\right)^2 \times 255$
- Neighborhood operation (filtering)

Linear filters

- Let $I(X, Y)$ be an $n \times n$ digital image, $F(X, Y)$ be another $m \times m$ digital image (**filter or kernel**)
 - F is usually normalized so the **sum of the elements is 1**.
- Let $k = \lfloor \frac{m}{2} \rfloor$, $I'(X, Y) = \sum_{j=-k}^k \sum_{i=-k}^k F(i, j) I(X + i, Y + j)$.
 - Each pixel in the output image is a linear combination of the central pixel and its neighboring pixels in the original image
 - There are a total of $m^2 \times n^2$ multiplications
- Boundary effects
 - **Ignore the locations**: make the computation undefined for the top and bottom k rows and the leftmost and rightmost k columns
 - **Pad the image with zeros**: return zero whenever a value of I is required at some position outside the defined limits of X and Y .
 - **Assume periodicity**: the top row wraps around to the bottom row, the leftmost column wraps around to the rightmost column
 - **Reflect border**: copy rows/columns locally by reflecting over the edge
- E.g.:
 - $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ returns the original image.
 - $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$ returns the image shift left by 1 pixel.
 - $\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ blur with a box filter.

$$\circ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \text{ sharpening.}$$

- **Correlation**

- $I'(X, Y) = \sum_{j=-k}^k \sum_{i=-k}^k F(i, j) I(X + i, Y + j).$

- **Convolution**

- $I'(X, Y) = \sum_{j=-k}^k \sum_{i=-k}^k F(i, j) I(X - i, Y - j).$

- Correlation filter rotated by 180.

- If $F(X, Y) = F(-X, -Y)$, then correlation=convolution

- CNN:

- Basic operations are convolutions followed by non-linear functions (non-linear filters)

- **Superposition:** $(F_1 + F_2) \otimes I(X, Y) = F_1 \otimes I(X, Y) + F_2 \otimes I(X, Y).$

- **Scaling:** $(kF) \otimes I(X, Y) = F \otimes (kI(X, Y)) = k(F \otimes I(X, Y))$

- **Shift invariance:** output is local (no dependence on absolute position)

- An operation is **linear** if it satisfies both superposition and scaling

- **Any** linear, shift invariant operation can be expressed as convolution

Box filter

- $\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

- Filter has equal positive values that sum up to 1

- Replaces each pixel with the average of itself and its local neighborhood

- It is also referred to as average filter or mean filter

Smoothing

- Smoothing with a box doesn't model lens defocus well

- Smoothing with a box filter depends on direction

- For images in which the center point is 1 and every other point is 0, the output will be a square of the same size as the box filter

- Smoothing with a **circular pillbox is a better model for defocus**

- Defocus: out of focus

- Gaussian is a good general smoothing model

- For phenomena that are the sum of other small effects

- Whenever the central limit theorem applies

Smoothing with a **Gaussian**

- Weight contributions of pixels by spatial proximity

- 2D Gaussian (continuous case)

- $G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$, σ is the standard deviation.

- If σ is larger, **more** blur.

- If σ is smaller, less blur.

- To get shadows, we can blur with a Gaussian kernel, then compose the blurred image with the original (with some offset)

- Better Gaussian filter

- Sums to 1 (normalized).

- Captures $\pm 2\sigma$.

- In general, we want the Gaussian filter to capture $\pm 2\sigma$.

Separability

- A 2D function of x and y is **separable** if it can be written as the product of two functions, one a function only of x and the other a function only of y . ($F(X, Y) = F(X)F(Y)$)

- If a 2D filter can be expressed by an outer product of two 1D filters, then it is separable

- It can be implemented as two 1D convolutions

- First convolve each row with a 1D filter

- Then convolve each column with a 1D filter
- The **2D Gaussian** is the only (non-trivial) 2D function that is both **separable and rotationally invariant**
 - A filter is rotationally invariant if we rotate the filter by some angle then the result of the convolving the image with the filter does not change
- Gaussian separability:
 - $G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) = \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right)$.
 - Reduces multiplications from $m^2 \times n^2$ to $2m \times n^2$.

Smoothing with a pillbox

- Let the radius of the filter be r , the 2D pillbox filter is defined as $f(x, y) = \frac{1}{\pi r^2} \begin{cases} 1, & \text{if } x^2 + y^2 \leq r^2 \\ 0, & \text{otherwise} \end{cases}$.
 - The scaling constant ensures that the area of the filter is one
- The **2D pillbox is rotationally invariant but not separable**
- Efficient implementation:
 - As a difference between convolution with a box filter and convolution with the extra corner bits filter
 - Postpone scaling the output to a single final step, so that convolution involves filters containing only 0 and 1. (no need for multiplication)

Speeding up the convolution

- steps
 - Let z be the product of two numbers, $z = xy$.
 - Take log: $\ln z = \ln x + \ln y$.
 - Then $z = \exp(\ln x + \ln y)$.
- At the expense of two \ln and one \exp , multiplication is reduced to addition.

Speeding up rotation

- Standard approach: use matrix multiplication $\begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$.
- Transform to polar coordinates, it becomes addition, at the expense of one polar coordinate transform and one inverse polar coordinate transform
- Similarly, some image processing operations become cheaper in a transform domain

Convolution theorem:

- Let $i'(x, y) = f(x, y) \otimes i(x, y)$ (convolution)
- Then $I'(w_x, w_y) = F(w_x, w_y) I(w_x, w_y)$ (element-wise multiplication) where I', F, I are Fourier transforms
- At the expense of two Fourier transforms and one inverse Fourier transform, convolution can be reduced to complex multiplication
- Cost of FFT/IFFT for image: $O(n^2 \log n)$.
- Cost of FFT/IFFT for filter: $O(m^2 \log m)$
- Cost of convolution: $O(n^2)$

Interpretations of correlation and convolution

- **Correlation**: measures **similarity** between two signals (filter and image)
 - Correlation in general is not associative
- **Convolution**: measures the effect one signal has on another signal
 - **Associative**: $G \otimes (F \otimes I(X, Y)) = (G \otimes F) \otimes I(X, Y)$
 - **Symmetric**: $(G \otimes F) \otimes I(X, Y) = (F \otimes G) \otimes I(X, Y)$

Pre-convolving filters

- Convolution of two filters of size $m \times m$ and $n \times n$ results in a filter of size $(n + 2 \lfloor \frac{m}{2} \rfloor)^2$
 - The sequence does not matter
- For a set of K filters of size $m_k \times m_k$, the resulting filter will have size $(m_1 + 2 \sum_{k=2}^K \lfloor \frac{m_k}{2} \rfloor)^2$.

- Convolution of two 1D Gaussian filters $G_{\sigma_1}(x), G_{\sigma_2}(x)$:
 - $G_{\sigma_1}(x) \otimes G_{\sigma_2}(x) = G_{\sqrt{\sigma_1^2 + \sigma_2^2}}(x)$.
 - Special case: convolving twice is $G_{\sqrt{2}\sigma}(x)$.

Non-linear filters

- **Median filter**: take the median value of the pixels under the filter size
 - Effective at reducing certain kinds of noise (impulse/salt and pepper/shot noise)
 - The filter forces points with distinct values to be more like their neighbors
- **Bilateral filter** (edge preserving non-linear filter)
 - Like a Gaussian
 - The filter weights depend on spatial distance from the center pixel
 - Pixels nearby should have greater influence than pixels far away
 - Unlike a Gaussian
 - The filter weights also depend on range distance from the center pixel
 - Pixels with similar brightness value should have greater influence than pixels with dissimilar brightness value
 - The weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ is given by a product: $\exp\left(-\frac{x^2+y^2}{2\sigma_d^2}\right) \exp\left(-\frac{(I(X+x, Y+y)-I(X, Y))^2}{2\sigma_r^2}\right)$.
 - The first part is the domain kernel
 - The second part is the **range kernel (different for each location in the image)**
 - Application: flash photography
 - **Non-flash images** taken under low light conditions often suffer from excessive **noise and blur**
 - Flash images
 - Color is unnatural
 - Strong shadows or specularities
 - We can combine flash and non-flash images to achieve better exposure and color balance, and to reduce noise
 - Joint/cross bilateral: range kernel is computed using a separate guidance image instead of the input image
- ReLU (rectified linear unit)
 - Set negative values to be 0.
 - Keep all positive values

Given $A \otimes B$, A symmetric convolution kernel. if B is not rotated, it is convolution, otherwise, it is correlation

Sampling

2021年9月22日 12:53

Images are a **discrete (sampled)** representation of a continuous world

Continuous case

- Image suggests a 2D surface whose appearance varies from point-to-point
- Appearance can be greyscale (b and w) or color
 - **Greyscale**: variation in appearance can be described by a single parameter corresponding to the amount of light reaching the image at a give point in a given time
- $i(x, y)$ is a **real-valued function** of **real spatial variables** x and y .
 - $i(x, y)$ is bounded, $0 \leq i(x, y) \leq M$.
 - $i(x, y)$ is bounded in extent, it has a value over at most a bounded region.
- Images can also be considered a function of time $i(x, y, t)$
 - t is the **temporal variable**.
- To make dependence of brightness on wavelength, we can have a **spectral variable** (λ)
 - More commonly, we think of color as discrete and write $i_R(x, y), i_G(x, y), i_B(x, y)$ for R, G, B color channels

Discrete case

- Superimpose grid on continuous image
- Sample the underlying continuous image according to the **tessellation or tiling** imposed by the grid
- Each grid cell is called a picture element (**pixel**)
- Denote $I(X, Y)$.
- Point sampling is useful for theoretical development
- Area-based sampling occurs in practice
- **Grey-levels**: Divide the range $[0, M]$ into a finite number of equivalence classes (**quantization**)
 - Suppose n bits-per-pixel are available, we can define $i(x, y) \rightarrow \left\lfloor \frac{i(x, y)}{M} (2^n - 1) + \frac{1}{2} \right\rfloor$.
 - Typically, $n = 8$ gives range of 255.

Sampling

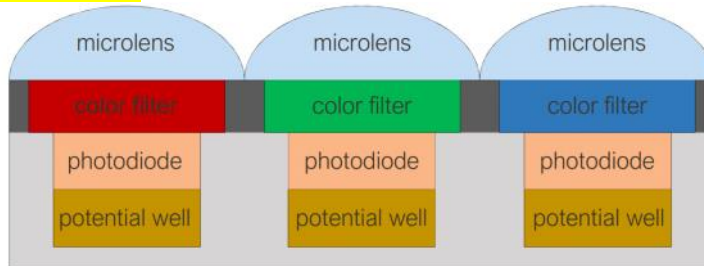
- Some information may be lost
- To reconstruct the original image, we need interpolation
- Case 0: $i(x, y)$ has a discontinuity not falling precisely at an integer
 - Cannot reconstruct exactly
- **Sampling theory**
 - Exact reconstruction requires constraint on the rate at which $i(x, y)$ can change between samples (bandlimited signal)
 - Music is bandlimited if it has some maximum **temporal frequency**
 - An image is bandlimited if it has some maximum **spatial frequency**
- Under sampling (**aliasing**)
 - Signal can be confused with one at lower frequency
 - Things are missing. There are artifacts
- Fundamental result:
 - For bandlimited signals, if we sample regularly at or above twice the maximum frequency (**Nyquist rate**), we can reconstruct the original signal exactly
- Over sampling (greater than the Nyquist rate)
 - Samples are redundant and wasted
- **Reducing** aliasing artifacts
 - **Oversampling**: sample more than we need and average
 - **Smoothing** before sampling
- Under sampling is unavoidable

- Medical imaging: try to maximize information content, tolerate some artifacts
- Computer graphics: try to minimize artifacts, tolerate some information missing

Color

- It is an artifact of human perception
 - How we subjectively perceive a very small range of the wavelengths
- It is not an objective physical property of light (light is characterized by its wavelength)

Color Filter Arrays (CFA)



- Design choices
 - Spectral sensitivity functions $f(\lambda)$.
 - Each camera has its unique and secret spectral sensitivity functions
 - Spatially arrange (mosaic) different color filters
 - Generally do not match human sensitivity
- RAW Bayer image
 - Lots of noise
 - Mosaic artifacts
- Demosaicing
 - Produce full RGB image from mosaiced sensor output
 - Interpolate from neighbors
 - Bilinear (average 4 neighbors)
 - Neighborhood changes for different channels
 - Bicubic (more neighbors, may over-blur)
 - Edge-aware

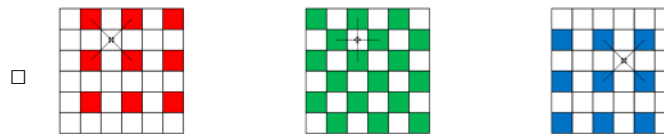
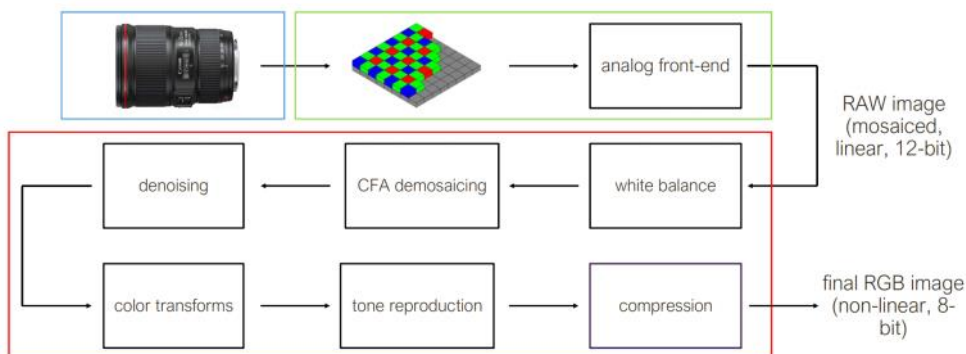


Image Processing Pipeline

- It is applied by the camera's image signal processor (ISP) to convert a RAW image into a conventional image



Template matching

- Use the pattern as a template to find part of one image that matches another
- Use convolution/correlation as comparing a template with each local image patch
 - Consider the filter and image patch as vectors
 - Applying a filter at an image location can be interpreted as computing the dot product

between the filter and the local image patch

- The dot product may be large because the image region is **bright**. Need **normalization**
- Correlation is a dot product
- **Normalized correlation** varies between -1 and 1 (magnitude 1). it attains 1 when the filter and image region are identical (up to a scale factor)
- **Linear filtering** the entire image computes the entire set of dot products, one for each possible alignment of the filter and the image
- If template is all positive, there is a **high correlation** between the template and the image at where there is a **bright spot** on the correlation map
 - **Detection** can be done by comparing correlation map score to a threshold
 - If the threshold is relatively **low**, the detection will become inaccurate



- If the threshold is very **high**, we might not detect anything
- Template matching **fails when**
 - Different scales (fixed in scaled representation)
 - Different orientation
 - Bad lighting conditions
 - Left/right hand
 - Partial occlusions
 - Different perspective
 - Motion/blur
- Good:
 - Works well in presence of noise
 - Easy to compute

Scaled representations

- Make template matching robust to changes in 2D scale
- Build a scaled representation: Gaussian image pyramid
- Alternatives
 - Use multiple sizes for each given template
 - Ignore the issue of scale
- To find **template matches** at all scales
 - Template size constant, image scale varies
 - Finding hands or faces when we don't know what size they are in the image
- **Efficient search** for image-to-image correspondences
 - Look first at coarse scales, refine at finer scales
 - Much less cost, but may miss best match
- To examine all levels of detail
 - Find edges with different amounts of blur
 - Find textures with different spatial frequencies (different levels of detail)
- **Shrinking** the image
 - Cannot take every second pixel
 - Artifacts appear
 - Small phenomena looks bigger
 - Fast phenomena looks slower
- **Sub-sample with Gaussian Pre-filtering**
 - Apply a smoothing filter first, then throw away half of the rows and columns

- Smoothing should be sufficient to ensure that the resulting image is bandlimited enough to ensure we can sample every other pixel
 - Practically, for every image reduction of $\frac{1}{2}$, smooth by $\sigma = 1$.

Image pyramid

- A collection of representations of an image
- Each layer is half the width and half the height of the previous layer
- Gaussian pyramid
 - Each layer is smoothed by a Gaussian filter and resampled to get the next layer
 - Details are smoothed out as we move to higher levels
 - Large uniform regions in the original image are preserved
 - Impossible to reconstruct the original image

Local feature detection

- Detects edges and corners
- Estimating derivatives
 - $\frac{\partial f}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x+\epsilon, y) - f(x, y)}{\epsilon}$.
 - Differentiation is linear and shift invariant, and thus can be implemented as a convolution
 - Discrete approximation: $\frac{\partial f}{\partial x} \approx F(X + 1, y) - F(x, y)$.
 - Image noise tends to result in pixels not looking exactly like their neighbors, so simple finite differences are sensitive to noise
 - To deal with this, smooth the image prior to derivative estimation

Edge & corner detection

September 27, 2021 11:15 AM

Example ($\frac{\partial f}{\partial x}$):

| | | | | | |
|---|---|-----|-----|---|---|
| 1 | 1 | 0.6 | 0.3 | 0 | 0 |
| 1 | 1 | 0.6 | 0.3 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | |
|---|------|------|------|---|--|
| 0 | -0.4 | -0.3 | -0.3 | 0 | |
| 0 | -0.4 | -0.3 | -0.3 | 0 | |
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | |

Example ($\frac{\partial f}{\partial y}$):

| | | | | | |
|---|---|-----|-----|---|---|
| 1 | 1 | 0.6 | 0.3 | 0 | 0 |
| 1 | 1 | 0.6 | 0.3 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | |
|---|---|-----|-----|---|---|
| | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0.6 | 0.3 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

The weights of any filter used for **differentiation** need to **sum to 0**. (for constant image, the derivative is 0)

Edges

- It is a location with **high gradient** (derivative)
- It is a property of the 2D images
- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (change in surface material properties)
- Illumination discontinuity (shadow)

Smoothing and differentiation

- Need smoothing to reduce noise prior to taking derivatives
- Need two derivatives in x and y direction
- Can use **derivative of Gaussian** filters:
 - Differentiation is convolution
 - Convolution is associative

Gradient magnitude

- Let I_x and I_y be estimates of partial derivatives in x and y directions
- Then $[I_x, I_y]$ is the **gradient**
 - It points in the direction of most rapid increase of intensity
- $\sqrt{I_x^2 + I_y^2}$ is the **gradient magnitude (edge strength)**
 - Increased smoothing:
 - Eliminates noise edges
 - Makes edges smoother and thicker
 - Removes fine detail
- **Gradient direction:** $\theta = \arctan \frac{f_y}{f_x}$.

Sobel edge detector

- Use central differencing to compute gradient image instead of first forward differencing (more accurate)
- Use threshold to obtain edges

Two generic approaches to edge point detection

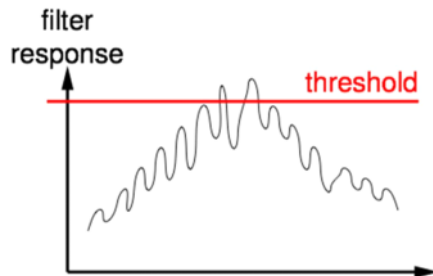
- Local extrema of a first derivative operator
- Zero crossing of a second derivative operator

Laplacian of Gaussian

- Zero crossing approach
- Criteria
 - Localization in space
 - Localization in frequency
 - Rotationally invariant
- Steps
 - Gaussian for smoothing
 - Laplacian ∇^2 for differentiation, $\nabla^2 f = f_{xx} + f_{yy}$.
 - Locate zero-crossing in the Laplacian of the Gaussian where $\nabla^2 G = -\frac{1}{2\pi\sigma^4} \left(2 - \frac{x^2+y^2}{\sigma^2} \right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$.

Canny Edge detector

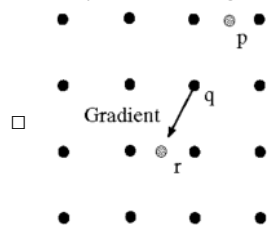
- Local extrema of a first derivative operator



- 2 edges
- Criteria
 - Good detection
 - Low error rate for omissions (missed edges)
 - Low error rate for commissions (false positive)
 - Good localization
 - One response to a given edge

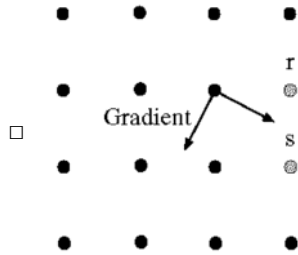
Steps

- Apply directional derivatives of Gaussian
- Compute gradient magnitude and gradient direction
- Non-maximum suppression
 - Thin multi-pixel wide ridges down to single pixel width
 - Idea: suppress near-by similar detections to obtain one true result
 - Select the image maximum point across the width of the edge
 - Value at q must be larger than interpolated values at p and r



◆ Keep q if gradient of q > p (backward) and q > r (forward)

- Linking and thresholding
 - Low, high edge-strength thresholds
 - Accept all edges over low threshold that are connected to edge over high threshold
 - Assume the marked point is an edge point, take the normal to the gradient at that point and use this to predict continuation points



Edge **hysteresis**

- One way to deal with **broken edge chains**
- Hysteresis: a lag or momentum factor
- Maintain two thresholds k_{high} and k_{low}
 - Use k_{high} to find strong edges to start edge chain.
 - Use k_{low} to find weak edges which continue edge chain.
- Typical ratio of the thresholds: $\frac{k_{high}}{k_{low}} = 2$.

Comparing edge detectors

- Good detection: minimize probability of false positives/negatives edges
- Good localization: found edges should be as close to true image edge as possible
- Single response: minimize the number of edge pixels around a single edge

| | Approach | Detection | Localization | Single Resp | Limitations |
|--------------------------|--|-----------|--------------|-------------|------------------------|
| • Sobel | Gradient Magnitude Threshold | Good | Poor | Poor | Results in Thick Edges |
| • Marr / Hildreth | Zero-crossings of 2nd Derivative (LoG) | Good | Good | Good | Smooths Corners |
| • Canny | Local extrema of 1st Derivative | Best | Good | Good | |

Laplacian pyramid

- Building an approximate Laplacian pyramid
 - Create a Gaussian pyramid
 - Take the difference between one Gaussian pyramid level and the next (before subsampling)
 - Laplacian = Unit - Gaussian
- Algorithm
 - Repeat:
 - Filter
 - Compute residual
 - subsample
 - Until min resolution reached
- Properties
 - Known as the difference-of-Gaussian (DOG) function, a close approximation
 - It is a band pass filter, each level represents a different band of spatial frequencies
- At each level, retain the residuals instead of blurred images themselves
- We can reconstruct the original image using the pyramid
 - Original = up sampled current + Laplacian
 - Repeat:
 - Up sample
 - Sum with residual
 - Until original resolution reached

Image blending

- Algorithm:
 - Build Laplacian pyramid LA and LB from images A and B
 - Build a Gaussian pyramid GR from mask image R (the mask defines which image pixels should be coming from A or B)

- Form a combined (blended) Laplacian pyramid LS, using nodes of GR as weights: $LS = GR * LA + (1 - GR) * LB$.
 - Reconstruct the final blended image from LS
- High level intuition
 - Smoother blending of flatter regions, sharper blending of more detailed regions

Edge matching in scaled representation fails when

- Different orientation
- Left/right hand
- Partial occlusions
- Different perspective
- Motion blur

Boundary detection

- We can formulate boundary detection as a high-level recognition task
- Many boundary detectors output a probability or confidence that a pixel is on a boundary
- Approach
 - Consider circular windows of radii r at each pixel cut in half by an oriented line through the middle
 - Compare visual features on both sides of the cut
 - If features are very different on the two sides, the cut line may correspond to a boundary
 - This gives an idea of the orientation of the boundary as well
- Features
 - Raw intensity
 - Orientation energy
 - Brightness gradient
 - Color gradient
 - Texture gradient
- For each feature type
 - Compute non-parametric distribution for left side
 - Compute non-parametric distribution for right side
 - Compare two histograms, on left and right, using statistical test
- Use all the histogram similarities as features in a learning based approach that outputs probabilities

Good feature:

- Can be easily found in different images (different orientation)
- Local: features are local, robust to occlusion and clutter
- Accurate: precise localization
- Robust: noise, blur, compression do not have a big impact on the feature
- Distinctive: individual features can be easily matched
- Efficient: close to real-time performance

Corner:

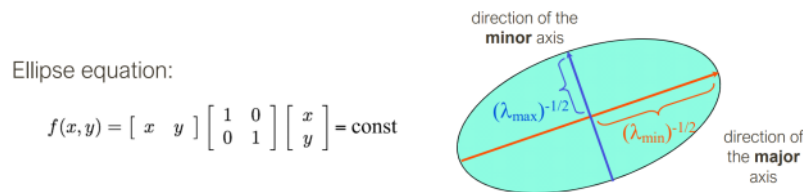
- Any locally distinct 2D image feature that corresponds to a distinct position on a 3D object of interest in the scene
- A corner can be **localized reliably**.
 - Place a small window over a patch of constant image value
 - Sliding the window in any direction, the image in the window will not change
 - Place a small window over an edge
 - Sliding the window in the direction of the edge, the image in the window will not change
 - Cannot estimate location along an edge (**aperture** problem)
 - Place a small window over a corner
 - Sliding the window in any direction, the image in the window changes
- To find a corner
 - Shifting a small window should give **large change in intensity**.

Autocorrelation

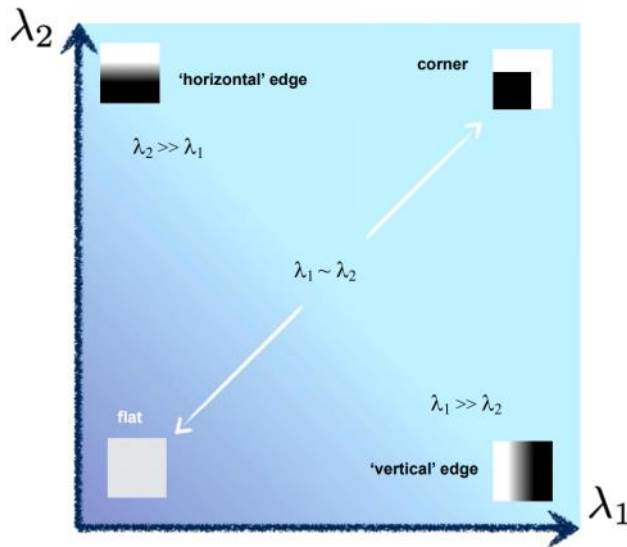
- Correlation of the image with itself
- Window centered on an edge point will have autocorrelation that falls slowly in the direction along the edge and rapidly in the direction perpendicular to the edge
- Windows centered on a corner point will have autocorrelation that falls rapidly in all directions

Corner detection:

- Edge detectors perform poorly at corners
- Observations
 - The gradient is ill-defined exactly at a corner
 - Near a corner, the gradient has two or more distinct values
- Harris corner detection
 - Compute image gradients over **small region**
 - $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$.
 - The distribution shows the **orientation and magnitude**
 - Compute the covariance matrix
 - $C = \begin{pmatrix} \sum_{p \in P} I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{pmatrix} = R^{-1} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} R$.
 - They are sum of products of gradients over small region around the corner
 - The matrix is symmetric
 - We are fitting a **quadratic** to the gradients over a small image region
 - ◻ $f(x, y) = x^2 + y^2 = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$.
 - We can visualize C as an ellipse with axis lengths determined by eigenvalues λ and orientation determined by R .



- Harris uses **Gaussian** instead
- Compute eigenvectors and eigenvalues



- Use threshold on eigenvalues to detect corners
 - Can use the smallest eigenvalue as the response function
 - Eigenvalues need to be bigger than 1, then $\lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2 = \det C - \kappa \text{trace}^2 C$ (more efficient).
 - Also can use $\frac{\det C}{\text{trace} C + \epsilon}$.
 - Harris also checks that ratio of eigenvalues is not too high

Properties:

- **Rotational invariance**
 - Ellipse rotates but its shape (eigenvalues) remains the same
 - Corner response is invariant to image rotation
- **Partial invariance to intensity shifts and scaling**
 - If only derivatives are used, invariant to intensity shifts
 - Intensity scale could affect performance
- **Not invariant to scale changes**
 - After scaling, corner may become an edge

- We can find local maxima in both position and scale
- When using a Laplacian filter, highest response when the signal has the same **characteristic scale** as the filter

| Representation | Result is. . . | Approach | Technique |
|----------------|-------------------|---------------------------|--------------------------|
| intensity | dense | template matching | (normalized) correlation |
| edge | relatively sparse | derivatives | $\nabla^2 G$, Canny |
| corner | sparse | locally distinct features | Harris |

Characteristic scale

- The scale that produces peak filter response
- We need to apply Laplacian filter at different scales and search over characteristic scales
- Implementation
 - For each level of Gaussian pyramid, compute feature response (Harris, Laplacian)
 - For each level of Gaussian pyramid
 - If local maximum and cross-scale
 - Save scale and location of feature (x, y, s) .

| Representation | Results in | Approach | Technique |
|----------------|-------------------|---------------------------|--------------------------|
| intensity | dense | template matching | (normalized) correlation |
| edge | relatively sparse | derivatives | Sobel, LoG, Canny |
| corner | sparse | locally distinct features | Harris (and variants) |
| blob | sparse | locally distinct features | LoG |

Texture

September 27, 2021 11:16 AM

Textures

- Texture is widespread, easy to recognize, but hard to define
- Views VL large numbers of small objects are considered textures
- Patterned surface markings are considered textures
- **Def**: detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements
- **Textons**: identifiable elements that repeatedly composes a pattern for texture

Use of texture

- Object identity: if the object has distinctive material properties
- Object's shape: deformation of the texture from point to point
- **Shape from texture**: estimating surface orientation or shape from texture

Texture analysis

- How do we represent texture

Texture synthesis

- How do we generate new examples of a texture
- Why:
 - To fill holes in images (inpainting)
 - To produce large quantities of texture for computer graphics
 - Good textures make object models look more realistic
- Idea: use an image of the texture as the source of a probability model
 - Draw samples directly from the actual texture
 - Can account for more types of structure
 - Very simple to implement
 - Success depends on choosing a correct distance

Texture synthesis by non-parametric sampling

- Like copying, but not just repetition
- Efros and Leung:
 - **synthesizing one pixel**
 - Conditional probability distribution of p given the neighborhood window:
 - directly search the input image for all such neighborhoods to produce a histogram for p
 - To synthesize p , pick one match at random
 - Since the sample image is finite, an exact neighborhood match might not be present
 - Find the **best match** using SSD (sum of squared difference) error, weighted by Gaussian to emphasize local structure and take all samples within some distance from that match
 - Synthesizing many pixels
 - For multiple pixels, grow the texture in layers
 - In the case of hole-filling, start from the edges of the hole
- **Big data**
 - Big data enables simple non-parametric, matching-based techniques to solve complex problems in CG and vision
 - Algorithm
 - Create a short list of a few hundred best matching images based on global image statistics
 - Find patches in the short list that match the context surrounding the image region

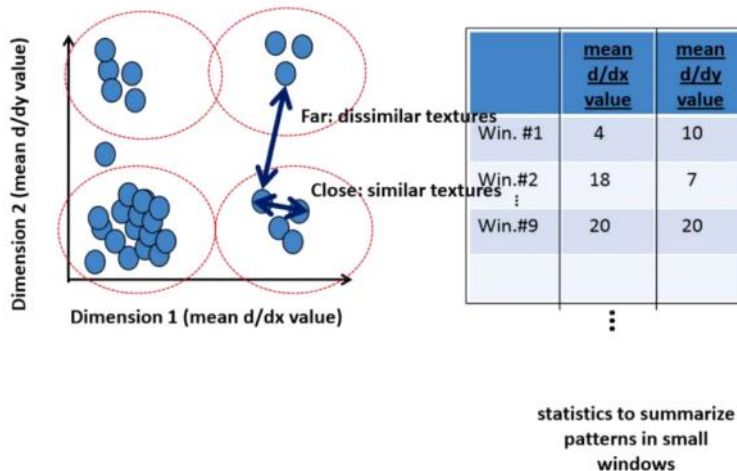
- we want to fill
 - Blend the match into the original image
 - Purely **data-driven**, requires no manual labeling of images

Texture segmentation

- Texture is a **property of a region**
- Texture segmentation** can be done by detecting boundaries between regions of the same or similar texture
- Texture **boundaries** can be detected using standard edge detection techniques applied to the texture measures determined at each point
- Uses a local window to estimate texture properties and **assigns the texture properties as point properties** of the window's center row and column

Texture representations

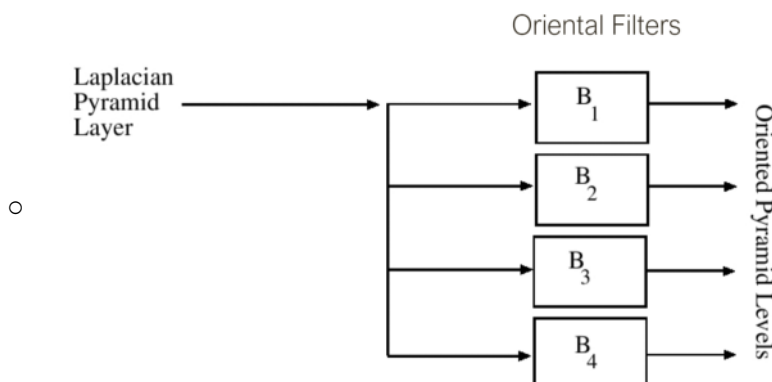
- Using RGB channel
- There are **infinitely many degrees of freedom** to texture
- Textures are made up of generic **sub-elements**, repeated over a region with similar statistical properties
- Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region
 - Spots and oriented edge filters** at a variety of different orientations and scales



- Texture representation is hard
 - Difficult to define and analyze
 - Texture synthesis appears more tractable**

Oriented pyramids

- Laplacian pyramid** is **orientation independent**
- Apply an oriented filter at each layer
 - Represent image at a particular scale and orientation



Goal of texture synthesis

- Given a finite sample of some texture, the goal is to synthesize other samples from that same texture
 - The sample needs to be large enough
- Compare textures and decide if they're made of the same stuff

Bag-of-words representation

- Take a large corpus of text
 - Represent every letter by a 26 dimensional one-hot vector
 - Represent each word by an average of letter representations in it
 - Cluster the words to get a dictionary.
 - Words that have very similar representations would get clustered together
 - Represent every document by histogram of dictionary atoms by associating every word to an atom that is closest in terms of distance in 26D
- By comparison
 - Corpus of text = collection of images
 - Letter = pixel location
 - Word = patch with pixel in the center
 - Dictionary = textons

Texture representation and recognition

- Texture is characterized by the repetition of basic elements or textons
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters

Color

September 27, 2021 11:16 AM

Color

- Light is produced in different amounts at different wavelengths by each light source
- Light is differentially reflected at each wavelength, which gives objects their native color (surface albedo)
- The sensation of color is determined by the visual system, based on the product of light intensity and reflectance

Color appearance

- Reflected light at each wavelength is the product of illumination and surface reflectance at that wavelength
- Surface reflectance is modeled as having two components
 - Lambertian: equal bright in all directions (diffuse)
 - Specular: mirror reflectance (shiny spots)

Color matching

- additive
 - $T = w_1P_1 + w_2P_2 + w_3P_3$.
 - Light is a weighted mixture of primaries
 - Many colors can be represented as a positive weighted sum
 - This defines a color description system
 - With an agreed P_1, P_2, P_3 , we only need supply w_1, w_2, w_3
 - RGB primaries, CRT monitors
- A negative weight means we need to add it to the test color side (subtractive)
 - $M + aA = bB + cC$
 - Interpreted as $(-a, b, c)$.
 - This raises a problem for designing displays
 - E.g. Ink, CMY primaries, films, prints
- We choose R,G,B to be the bases so that positive linear combinations match a large set of colors

Principles of Trichromacy

- Three primaries work, provided we allow subtractive matching
- Exceptional people can only match with two or one primary

Metameric lights

- Two lights whose spectral power distributions appear identical to most observers are called metamers

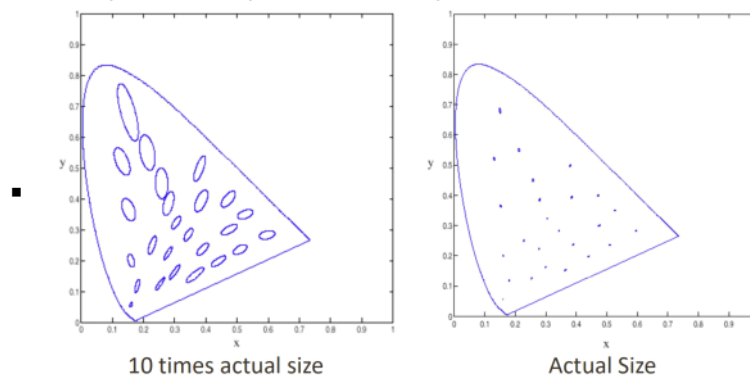
Grassmans law for color matches:

- Symmetry
- Transitivity
- Proportionality
- Additivity
- This means that color matching is linear

Representing color

- Linear color space
 - The coordinates of a color are given by the weights of the primaries used to match it
 - Choice of primaries is equivalent to choice of color space
 - RGB: R: 645.2nm, G: 526.3nm, B: 444.4nm.
 - Negative parts means some colors can be matched only by subtraction

- The subspace of CIE colors that can be displayed on a typical computer monitor
- **CIE XYZ**: primaries are imaginary, but have some convenient properties. Color coordinates are (X, Y, Z) , where X is the amount of the X primary
 - Always positive, but primaries are imaginary
 - $x = \frac{X}{X+Y+Z}, y = \frac{Y}{X+Y+Z}$
 - Overall brightness is ignored
 - **Geometry** of Color:
 - ◆ White is in the center, with saturation increasing towards the boundary
 - ◆ Mixing two colored lights creates colors on a straight line
 - ◆ Mixing 3 colors creates colors within a triangle
 - ◆ Curved edge means there are no 3 actual lights that can create all colors that humans perceive
- May not
 - Encode properties that are common in language or important in applications
 - Capture human intuitions about the topology of colors
- **Uniform** color spaces
 - One cannot reproduce colors exactly
 - It is important to know whether a color difference would be noticeable to a human viewer
 - **McAdam ellipses**: each ellipse shows colors perceived to be the same



- Differences in x, y are a poor guide to differences in perceived color
- A uniform color space is one in which differences in coordinates are a good guide to differences in perceived color
- **HSV** color space
 - More natural description of color for human interpretation
 - **Hue**: attribute that describes a pure color
 - Red, blue
 - **Saturation**: measure of the degree to which a pure color is diluted by white light
 - Pure spectrum colors are fully saturated
 - **Value**: intensity or brightness
 - Hue + saturation is referred to as chromaticity

Color constancy

- Image color depends on both light color and surface color
- **Color constancy**: determine hue and saturation under different colors of lighting
- **Goal**: correct the changes in illumination in different parts of the scene so that the entire image is made consistent with what would be seen under any fixed reference light source (white light)
- Difficult to predict what colors a human will perceive in a complex scene
 - Depends on context, other scene information
- Humans can usually perceive
 - The color a surface would have under white light
- Environmental effects
 - **Chromatic adaptation**: if the human visual system is exposed to a certain color light for a

- while, color perception starts to skew
- **Contrast effects**: nearby colors affect what is perceived

Object recognition

September 27, 2021 11:16 AM

Photometric transformations: change in color

Geometric transformations: objects will appear at different **scales, translation and rotation**

Good local features

- Corner
- Blob
- **Patch** around the local feature is very informative

Intensity image

- Just use the pixel values of the patch
- Perfectly fine if geometry and appearance is unchanged (template matching)
- Problems:
 - Sensitive to absolute intensity values

Image gradients/edges

- Use pixel differences
- Feature is invariant to absolute intensity values
- Problems
 - Sensitive to deformations

Key point:

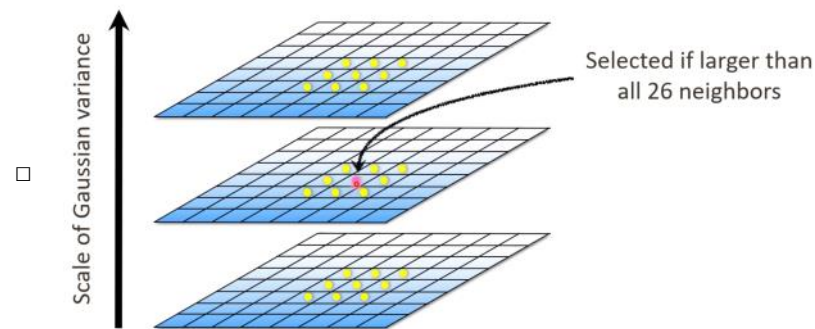
- An image location at which a descriptor is computed
 - Locally distinct points
 - Easily localizable and identifiable

Feature **descriptor**

- Summarizes the local structure around the key point
- Allows unique matching of key points in presence of object pose variations, image and photometric deformations

Scale invariant features (SIFT)

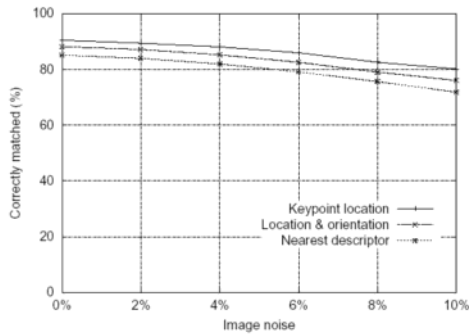
- Can help detect locally distinct features (corners)
- **David Lowe's** invariant local features
 - Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale and other imaging parameters
- **Advantages** of invariant local features
 - **Locality**: robust to occlusion and clutter
 - **Distinctiveness**: individual features can be matched to a large database of objects
 - **Quantity**: many features can be generated for even small objects
 - **Efficiency**: close to real-time performance
- SIFT describes both a detector and descriptor
 - Multi-scale extrema detection
 - Extreme in both scale and dimensions
 - $\sigma = 2^{1/s}$



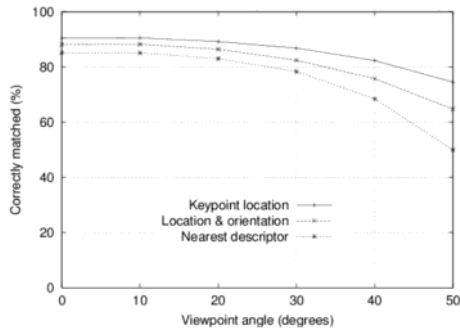
- Apply Laplacian filter at different scales (usually **half the size**)
- Detect maxima and minima of **DoG** (difference of Gaussian) in scale space
- Sampling frequency:
 - More points are found as sampling frequency increases
 - Accuracy of matching decreases after **3 scales/octave**
- **Key point** localization
 - Select **stable key points**
 - After key points detection, remove those that have **low contrast** or are **poorly localized** along an edge
 - To check poor localization, use the **ratio of eigenvalues of C** (Harris corners), check if it is greater than a threshold
- Orientation assignment
 - Create **histogram** of local gradient directions computed at selected scale
 - With 10 degree increments, we have histogram of 36 bins
 - Size of window is 1.5 scale (Gaussian filter)
 - Gaussian-weighted voting
 - Highest peak and peaks above 80% of highest also considered for calculating dominant orientations
 - Assign canonical orientation at peak of smoothed histogram
 - Each key specifies stable 2D coordinates (x, y, scale, orientation)
 - Multiply gradient magnitude by a Gaussian kernel
- Key point descriptor
 - Should be robust to local shape distortions, changes in illumination or 3D viewpoint
 - **SIFT descriptor**
 - Thresholded image gradients are sampled over 16×16 array of locations in scale space (weighted by a Gaussian with sigma half the size of the window)
 - Create array of orientation histograms
 - 8 orientations \times 4 \times 4 histogram array.
 - **Normalized** to unit length to reduce the effects of illumination change
 - ◆ If brightness values are multiplied by a constant, the gradients are scaled by the same constant. Normalization cancels the change
 - ◆ If brightness values are increased/decreased by a constant, the gradient does not change
- **Dimensions** in a SIFT descriptor
 - $4 \times 4 \times 8 = 128$.
 - 16 histograms
 - 8 orientations in each histogram

Feature stability to **noise**

- Match features after random change in image scale and orientation, with differing levels of image noise

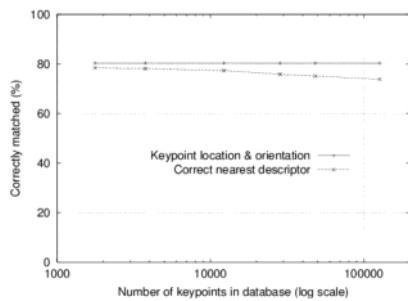


Feature stability to affine change



Distinctiveness of features

- Vary size of database of features, with affine change, image noise

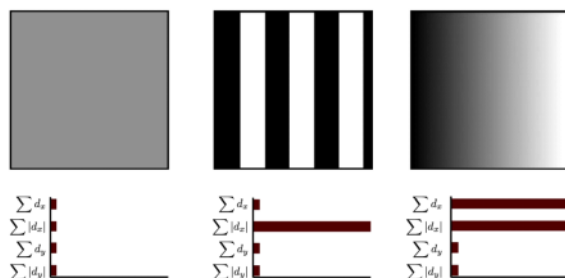


Histogram of Oriented Gradient (HOG) features

- From a cell (8 × 8 pixels), extract a block (2 × 2 pixels)
- Histogram of unsigned gradients, one for each cell
- Concatenate and L-2 normalization
- Single scale, no dominant orientation
- Redundant representation due to overlapping blocks

Speeded up robust features (SURF)

- In each 4 × 4 cell grid, we have 5 × 5 sample point.
 - Each cell is represented by 4 values $(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$.
- 64 dimensions



Object recognition with invariant features

- Identify objects or scenes and determine their pose and model parameters
- Applications
 - Industrial automation and inspection
 - Mobile robots, toys, user interfaces
 - Location recognition
 - Digital camera panoramas
 - 3D scene modeling, augmented reality

| Keypoint Detection Algorithms | Representation | | Keypoint Description Algorithms | Representation |
|-------------------------------|-----------------|--|---------------------------------|----------------|
| Harris Corners | (x,y,s) | | SIFT | 128D |
| LoG / Blobs | (x,y,s) | | Histogram of Oriented Gradients | 3780D |
| SIFT | $(x,y,s,theta)$ | | SURF | 64D |

Object recognition

- It requires us to first match each key point independently to the database of key points
 - Many features will not have any correct match in the database because they arise from background clutter
 - It would be useful to have a way to discard features that do not have any good match

Probability of correct match

- Compare ratio of distance of nearest neighbor to second nearest neighbor (from different object)
- Threshold of 0.8 provides excellent separation

Nearest-neighbor matching

- Hypotheses are generated by approximate nearest neighbor matching of each feature to vectors in the database
 - Finding the nearest neighbor in large (N) high-dimensional (d) datasets is linear
 - Only approximate methods are feasible
- Can give speedup by factor of 1000 while finding nearest neighbor

Identify consistent features

- Identify clusters of at least 3 features that agree on an object and its pose
 - Detecting less than 1% inliers among 99% outliers
- Hough transform
 - Vote for each potential match according to model ID and pose
 - Insert into multiple bins to allow for error in similarity approximation

Model verification

- Examine all clusters with at least 3 features
- Perform least-squares affine fit to model
- Discard outliers and perform top-down check for additional features

Forms of model

- D.O.F. = degree of freedom = # parameters

| Name | Matrix | # D.O.F. |
|-------------------|--|----------|
| translation | $\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$ | 2 |
| rigid (Euclidean) | $\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$ | 3 |
| similarity | $\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$ | 4 |
| affine | $\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$ | 6 |
| projective | $\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$ | 8 |

- Rigid: 2 for translation, 1 for rotation
 - Similarity: 3 for rigid, 1 for similarity
 - A projective transformation (homography) warp projective plane into another
- **Warping a model**: transform any pixel in the original image to the corresponding image
 - Changes domain of image function
 - If $\begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$, we have $M = \begin{pmatrix} m_1 & m_2 & t_x \\ m_3 & m_4 & t_y \\ 0 & 0 & 1 \end{pmatrix}$.
 - This is 6 D.O.F.
- Solutions for affine parameters
 - Mapping (x_i, y_i) to (u_i, v_i)

$$\begin{matrix} \circ \\ \circ \end{matrix} \begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ & & \dots & \dots & & \\ & & \dots & \dots & & \\ x_k & y_k & 0 & 0 & 1 & 0 \\ 0 & 0 & x_k & y_k & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \dots \\ \dots \\ u_k \\ v_k \end{bmatrix}$$

Homography (projection) can be used when

- The scene is planar
- The scene is very far or has small depth variation (approximately planar)
- The scene is captured under camera rotation only (no translation or pose change)

3D object recognition

- Extract outlines with background
- Subtraction
- Only 3 key points are needed, so extra key points provide **robustness**
- Under **occlusion**
 - Solve for M (affine transform) for each object by leveraging SIFT matches of key points for that object, then apply M to the outline of that object

Object recognition with SIFT

- Match each key point independently to database of known key points extracted from training examples
 - Use fast (approximate) nearest neighbor matching
 - Threshold based on ratio of distances to best and to second best match
- Identify clusters of at least 3 matches that agree on an object and a similar pose
 - Use generalized Hough transform
- Check each cluster found by performing detailed geometric fit of affine transformation to the model
 - Accept/reject interpretation accordingly

- **Limitation**: we need to have correct matches
 - Very difficult
 - If we can find exact match 80% of the time, we can find 3 matches correctly only about 50% of the time
 - Image noise, deformations will make this worse
 - Multiple object instances will make this impossible

Fitting a model to noisy data

- Suppose we are fitting a line to a dataset that consists of 50% outliers
- Using two points
 - Draw pairs of points uniformly at random, $\frac{1}{4}$ of the pairs will consist entirely of inliers
 - Can identify these good pairs by noticing that a large collection of other points lie close to the line fitted to the pair
 - A better estimate of the line can be obtained by refitting the line to the points that lie close to the line

RANSAC (RANDOM Sample Consensus)

- **Randomly** choose minimal subset of data points necessary to fit model (a **sample**)
- Points with some distance threshold t of model are a **consensus set**
 - The size of the consensus set is the model's **support**
- Repeat for N samples, the model with biggest support is the most robust fit
 - Points within distance t of best model are inliers
 - Fit final model to all inliers

| Algorithm 15.4: RANSAC: fitting lines using random sample consensus | |
|---|---|
| Determine: | <ul style="list-style-type: none"> n — the smallest number of points required k — the number of iterations required t — the threshold used to identify a point that fits well d — the number of nearby points required to assert a model fits well |
| Until k iterations have occurred | <ul style="list-style-type: none"> Draw a sample of n points from the data uniformly and at random Fit to that set of n points For each data point outside the sample <ul style="list-style-type: none"> Test the distance from the point to the line against t; if the distance from the point to the line is less than t, the point is close end If there are d or more points close to the line then there is a good fit. Refit the line using all these points. |
| end | Use the best fit from this collection, using the fitting error as a criterion |

- Choosing **sample number**
 - Let ω be the fraction of inliers
 - Let n be the number of points (DoF) needed to define hypothesis ($n = 2$ for a line in the plane, $n = 3$ for a circle in the plane)
 - Suppose k samples are chosen (each of n points)
 - The probability that a **single sample of n points is correct is ω^n**
 - The probability that all sample fails is $(1 - \omega^n)^k$
 - Choose k large enough to keep the failure probability below target level
- RANSAC divides data into inliers and outliers and yields an estimate computed from the minimal set of inliers
 - Improve this initial estimate with estimation over all inliers (with standard least squares minimization)
 - But this may change inliers, so alternate fitting with re-classification as inlier/outlier
- **Advantages**

- General method suited for a wide range of model fitting problems
- Easy to implement and easy to calculate its failure rate
- **Disadvantages**
 - Only handles a moderate percentage of outliers without cost blowing up
 - Many real problems have high rate of outliers, but sometimes selective choice of random subsets can help

Automatic matching

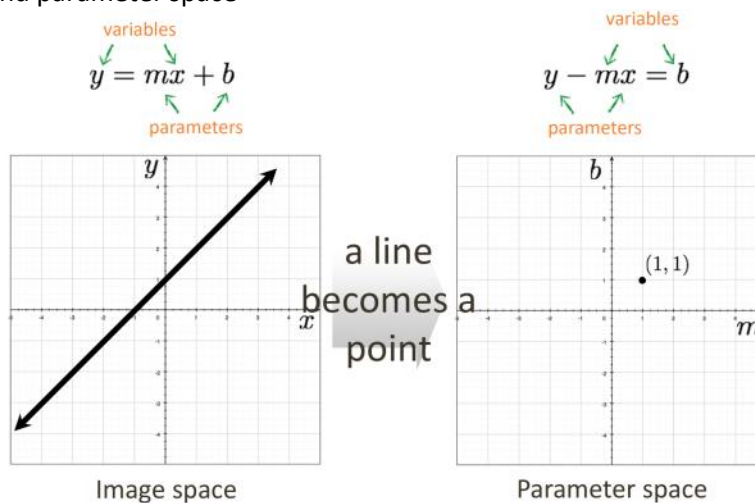
- Feature extraction
 - Find features in pair of images using Harris corner detector
 - Assumes images are roughly the same scale
- Initial match hypothesis
 - Select best match over threshold within a square search window using SSD or normalized cross-correlation for small patch around the corner
- Use RANSAC to find outliers and inliers, then find match

Fit a model to a set of tokens is **difficult**

- Extraneous data: clutter or multiple models
 - We do not know what is part of the model
- Missing data: only some parts of model are present
 - Noise causes erroneous matches, perturbs solution
- Computational cost:
 - Not feasible to check all combinations of features by fitting a model to each possible subset

Hough transform

- Idea
 - For each token, vote for all models to which the token could belong
 - Return models that get many votes
- Image and parameter space



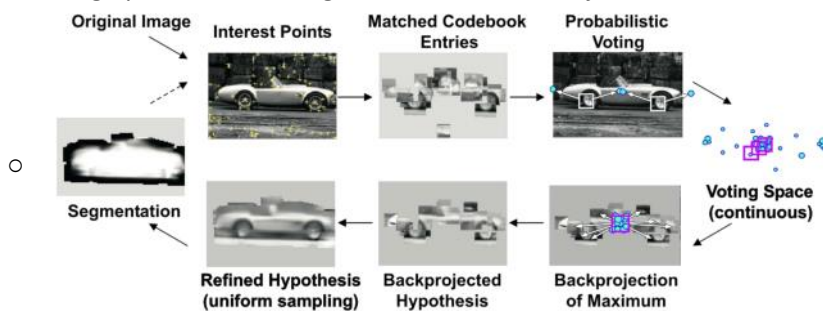
- A point in image space becomes a line in parameter space
- n points in image spaces gives n lines in the parameter space
 - Use the intersection to find m and b . The more lines intersecting, the better the approximation.
- **Line detection in slope and intercept form**

1. Quantize Parameter Space (m, c)
 2. Create Accumulator Array $A(m, c)$
 3. Set $A(m, c) = 0 \quad \forall m, c$
 - 4. For each image edge (x_i, y_i)
 - For each element in $A(m, c)$
 - If (m, c) lies on the line: $c = -x_i m + y_i$
 - Increment $A(m, c) = A(m, c) + 1$
 5. Find local maxima in $A(m, c)$
- **Parametrization**
 - Space of m and c are huge, the accumulator needs to be very large
 - Lines in **normal form**
 - $x \sin \theta + y \cos \theta + \rho = 0, r \geq 0, \theta \in [0, 2\pi]$.
 - r is bounded by the edge of the image
 - A point in image space becomes a wave in the parameter space
 - n points becomes n waves in the parameter space
 - A line in image space becomes a point in the parameter space
 - **Hough transform for lines uses normal form**
 - Each point votes for the lines that pass through it
 - A line is the set of points, (x, y) such that
$$x \sin \theta - y \cos \theta + r = 0$$
 - Different choices of θ, r give different lines
 - — For any (x, y) there is a one parameter family of lines through this point. Just let (x, y) be constants and for each value of θ the value of r will be determined
 - Each point enters votes for each line in the family
 - If there is a line that has lots of votes, that will be the line passing near the points that voted for it
 - **Mechanics**
 - Construct a quantized array to represent θ and r
 - For each point, render curve (θ, r) into this array adding one vote at each cell
 - **Difficulties: size of cells (bins)**
 - Too big, merge different lines
 - Too small, noise causes lines to be missed
 - Number of lines
 - Count the peaks in the Hough array
 - Treat adjacent peaks as a single peak
 - Practical details
 - It is best to **vote for the two closest bins** in each dimension, as the locations of the bin boundaries are arbitrary
 - Peaks are blurred and noise will not cause similar votes to fall into separate bins
 - Use **hash table** to store the votes
 - No effort is wasted on initializing and checking empty bins
 - Avoids the need to predict the maximum size of the array, which can be non-rectangular
 - A key is to have each feature (token) determine as many parameters as possible
 - Lines are detected more effectively from edge elements with both **position and orientation**
 - For object recognition, each token should predict **position, orientation and scale**
 - The Hough transform can extract feature groupings from clutter in linear time
 - **Advantages**
 - Can handle high percentage of outliers: each point votes separately
 - Can detect multiple instances of a model in a single pass
 - **Disadvantages**
 - Complexity of search time increases exponentially with the number of model parameters

- Can be tricky to pick a good bin size
- **Summary**
 - Hough transform is a technique for fitting data to a model
 - A voting procedure
 - Possible model parameters define a quantized accumulator array
 - Data points vote for compatible entries in the accumulator array
 - Key is to have each data point (token) constrain model parameters as tightly as possible
- VS. RANSAC
 - Hough is better with **large number of outliers**, well over 50%
 - Setting bin size to account for certain level of **noise is more difficult in Hough**
 - RANSAC better for **high dimensional parameter spaces**

Object recognition - **implicit shape model**

- Combined object detection and segmentation using an implicit shape model
- Image patches cast weighted votes for the **object centroid**
- Index displacements by visual codeword
- Basic idea
 - Find **interest points/key points** in an image
 - **Match patch** around each interest point to a training patch
 - **Vote** for object center given the training instances
 - Find the patches that voted for the peaks (**back-project**)
- Easy but slow
 - We need to match a patch around each key point to all patches in all training images
- **Segmentation: inferring other information**
 - Can combine object detection and segmentation using an implicit shape model
 - Image patches cast weighted votes for the object centroid



- When back-projecting, back-project labeled segmentations per training patch
- Can also infer: **part labels, depth**, etc

Visual words

- Visual vocabulary
- Compare each patch to a small set of visual words (clusters)

Object recognition - **boundary fragments**

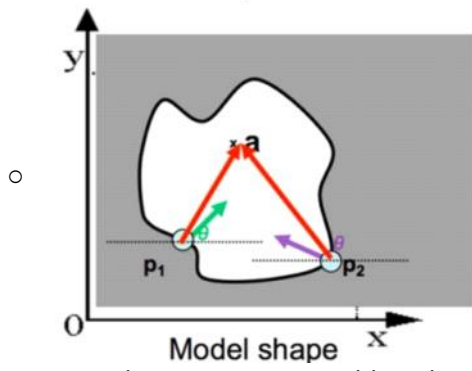
- Boundary fragments cast weighted votes for the **object centroid**. Also obtains an estimate of the **object's contour**

Object recognition - Poselets

- Poselets are image patches that have distinctive appearance and can be used to infer some of the configuration of a parts-based object
- Detected poselets vote for the object configuration

Generalized Hough transform

- Detect an **arbitrary** geometric shape
 - Normal Hough transform only detects a definite geometric shape (lines, circles)
- Offline procedure
 - At each boundary point, compute displacement vector $r = a - p_i$



- Store these vectors in a table indexed by gradient orientation θ

Camera & Motion

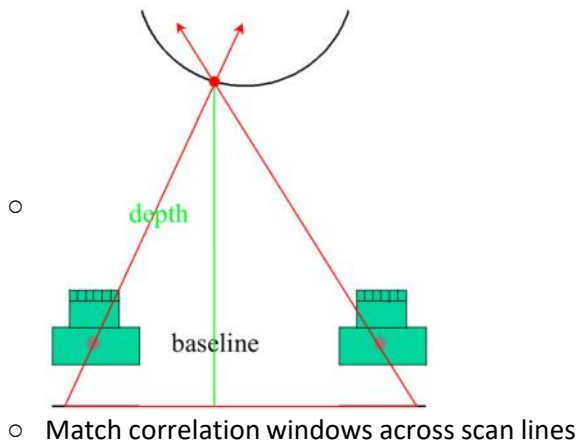
October 27, 2021 11:07 AM

Binoculars

- They enhance binocular depth perception in two ways
 - Magnification
 - Longer baseline compared to human eyes

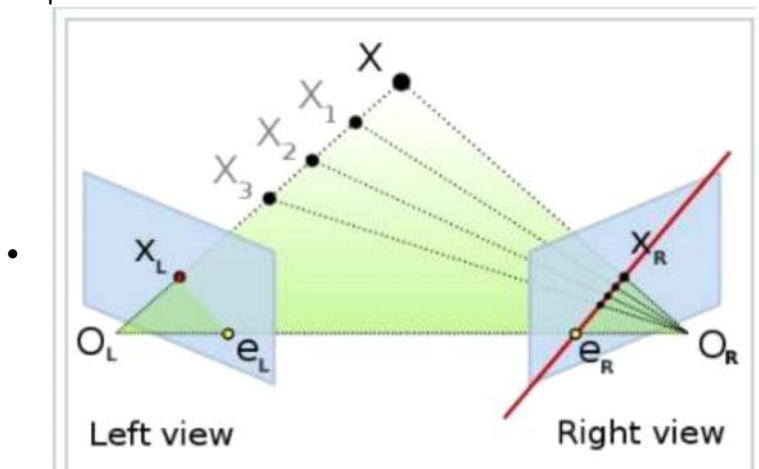
Stereo vision

- Key idea: 3D coordinates of each point imaged are constrained to lie along a ray. This is true also for a second image obtained from a slightly different viewpoint. Rays for the same point in the world intersect at the actual 3D location of that point
- Use two cameras, acquire images of the world from slightly different viewpoints
- Perceive **depth** based on **differences in the relative position of points** in the left and right image
- **Task:** compute depth from two images acquired from slightly different viewpoints
- **Approach:** match locations in one image to those in another
- **Sub-tasks**
 - Calibrate cameras and camera positions
 - Focal length
 - Displacement of two cameras
 - **Find all corresponding points**
 - Compute depth and surfaces
- Triangulate on two images of the same point



- Match correlation windows across scan lines
- **Scan lines:** horizontal lines in the sensor, picking pixels

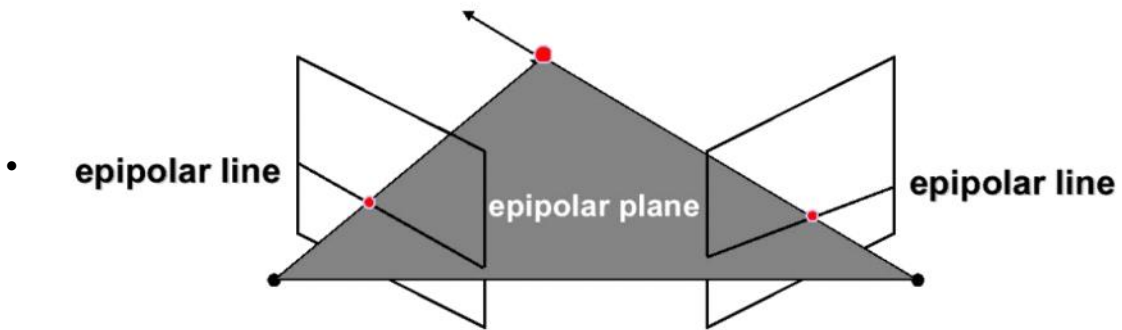
Correspondence



- X_L could be the projection of any point along the ray O_L to X

- The projection of those points in the right image forms a line called the **epipolar line**

Epipolar constraint



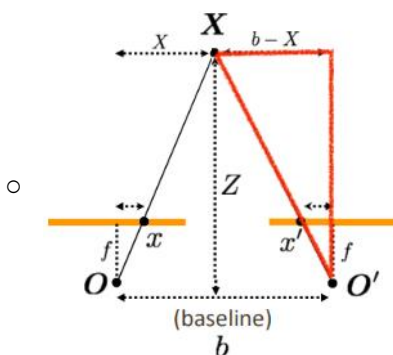
- Match points lie along corresponding epipolar lines
- Reduces correspondence problem to 1D search along conjugate epipolar lines
- Greatly reduce cost and ambiguity of matching
- Epipolar constraints hold in all different cases
 - 2 cameras have different focal points
 - Different height offset from the ground
 - Different lens characteristics
 - Different resolution of sensors

Rectified images (results)

- Image planes of cameras are **parallel**
- Focal **points** are at the same height
- Focal **lengths** are the same
- **Epipolar lines** fall along the **horizontal scan lines** of the images
- Assume images have been rectified so that epipolar lines correspond to scan lines
 - Simplifies algorithms
 - Improves efficiency

Rectified stereo pair

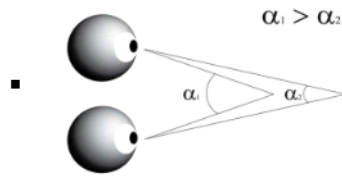
- Reproject image planes onto a common plane parallel to the line between camera centers
- Need two homographies (3×3 transform), one for each input image reprojection
- **Depth estimate**



- $\frac{X}{Z} = \frac{x}{f}$.
- $\frac{b-X}{Z} = \frac{x'}{f}$.
- **Disparity**: $d = x - x' = \frac{bf}{Z}$ (inversely proportional to depth).

Human stereo vision

- When a human looks at a point on a surface
 - The eyes focus on the surface (accommodation)
 - The eyes verge on the point (vergence)



Stereo algorithm

1. Rectify images
(make epipolar lines horizontal)

2. For each pixel

- a. Find epipolar line
- b. Scan line for best match
- c. Compute depth from disparity $Z = \frac{bf}{d}$

• Scan line match

- Recognition is not needed
- Pixel matching

For each epipolar line

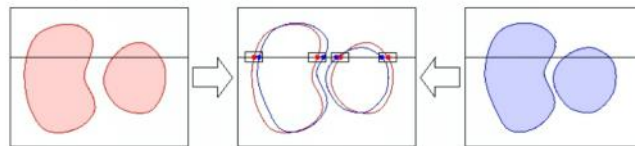


For each pixel in the left image

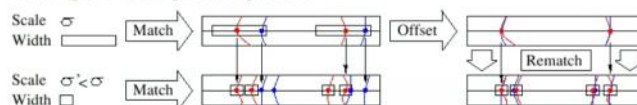
- — compare with every pixel on same epipolar line in right image
- — pick pixel with minimum match cost
- Too much ambiguity
- Use SSD, correlation

○ Edge matching

Matching zero-crossings at a single scale



Matching zero-crossings at multiple scales



▪ Marr/Poggio multiscale stereo algorithm

1. Convolve the two (rectified) images with $\nabla^2 G_\sigma$ filters of increasing $\sigma_1 < \sigma_2 < \sigma_3 < \sigma_4$

2. Find zero crossings along horizontal scanlines of the filtered images

□ 3. For each filter scale σ , match zero crossings with the same parity and roughly equal orientations in a $[-\mathbf{w}_\sigma, +\mathbf{w}_\sigma]$ disparity range, with $\mathbf{w}_\sigma = 2\sqrt{2}\sigma$ (parity=bw or wb)

4. Use the disparities found at larger scales to control eye vergence (shifts zero-crossings) and cause unmatched regions at smaller scales to come into correspondence

○ Comparison

- Edges are more meaningful, but hard to find
- Edges tend to fail in dense texture
 - Edge based methods are sparse
- Correlation tend to fail in smooth, featureless regions
 - Correlation based methods are dense

Sum of squared (pixel) difference (SSD):

- w_L and w_R are corresponding $m \times m$ windows of pixels
- Define the window function by

$$W_m(x, y) = \left\{ (u, v) : x - \frac{m}{2} \leq u \leq x + \frac{m}{2}, y - \frac{m}{2} \leq v \leq y + \frac{m}{2} \right\}.$$

- SSD measures intensity difference as a function of disparity:

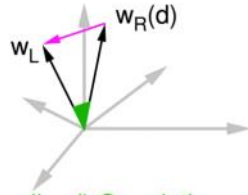
$$\circ C_R(x, y, d) = \sum_{(u,v) \in W_m(x,y)} (I_L(u, v) - I_R(u - d, v))^2.$$

Image normalization

- Average pixel: $\bar{I} = \frac{1}{|W_m(x,y)|} \sum_{(u,v) \in W_m(x,y)} I(u, v)$
- Window magnitude: $\|I\|_{W_m(x,y)} = \sqrt{\sum_{(u,v) \in W_m(x,y)} I(u, v)^2}$
- Normalized pixel: subtract the mean, normalize to unit length
 - $\hat{I}(x, y) = \frac{I - \bar{I}}{\|I - \bar{I}\|_{W_m(x,y)}}$.

Image metrics

(Normalized) Sum of Squared Differences



- Assume w_L and w_R are normalized to unit length
- SSD: $C_{SSD}(d) = \|w_L - w_R(d)\|^2$.
- Correlation: $C_{NC}(d) = w_L \cdot w_R(d) = \cos \theta$.
- Let d^* be the value of d that minimizes C_{SSD} , then it also minimizes C_{NC} .

Similarity measures

| Similarity Measure | Formula |
|------------------------------------|---|
| Sum of Absolute Differences (SAD) | $\sum_{(i,j) \in W} I_1(i, j) - I_2(x + i, y + j) $ |
| Sum of Squared Differences (SSD) | $\sum_{(i,j) \in W} (I_1(i, j) - I_2(x + i, y + j))^2$ |
| Zero-mean SAD | $\sum_{(i,j) \in W} I_1(i, j) - \bar{I}_1(i, j) - I_2(x + i, y + j) + \bar{I}_2(x + i, y + j) $ |
| Locally scaled SAD | $\sum_{(i,j) \in W} I_1(i, j) - \frac{\bar{I}_1(i, j)}{\bar{I}_2(x + i, y + j)} I_2(x + i, y + j) $ |
| Normalized Cross Correlation (NCC) | $\frac{\sum_{(i,j) \in W} I_1(i, j) \cdot I_2(x + i, y + j)}{\sqrt{\sum_{(i,j) \in W} I_1^2(i, j) \cdot \sum_{(i,j) \in W} I_2^2(x + i, y + j)}}$ |

Effect of window size

- Smaller window:
 - More detail
 - More noise
- Larger window
 - Smoother disparity
 - Less detail
 - Fails near boundaries
- Adaptive window size
 - Try multiple sizes and select best match

Ordering constraints

- We might have some objects blocking the object we want to match
- **Block matching:**
 - Too many **discontinuities**
 - Disparity values change slowly

Stereo matching as energy minimization

- Assume depth should change smoothly
- Energy function for one pixel

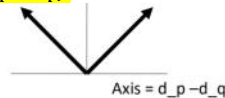
$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \lambda \underbrace{E_s(d)}_{\text{smoothness term}}$$

- Want each pixel to find a good match in the other image
(block matching result)
- Adjacent pixels should (usually) disparity about the same
(smoothness function)

- Smoothness term: $E_s(d) = \sum_{(p,q) \in \epsilon} V(d_p, d_q)$.

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

L₁ distance



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

"Potts model"



- Can minimize this independently per scanline using **dynamic programming** (DP)
- Interpret the solution as finding a path that is
 - Near the data (springs connecting data to solution)
 - Smooth, without bending of the flexible path
- Trades off to fit the data vs smoothness of solution

We can use more cameras to reduce ambiguity in stereo matching

Structured light imaging

- Structured light and one camera
- Projector acts like reverse camera

Summary

- Stereo is formulated as a **correspondence** problem
 - Determine match between location of a scene point in one image and its location in another
- If we assume calibrated cameras and image rectification, epipolar lines are horizontal scan lines
- We can use the following to match
 - Pixels
 - Patches
 - Edges

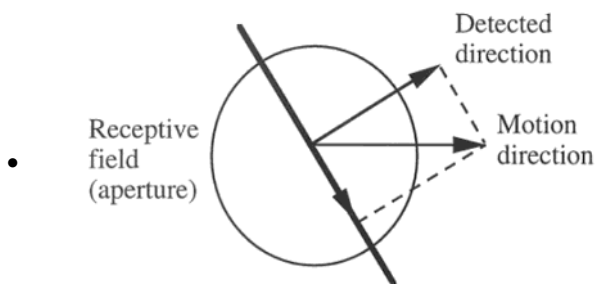
Optical flow

- Determine how many objects (and/or the camera) moves in the 3D world
- Key idea:
 - images acquired as a continuous function of time provide additional constraints.
 - Formulate motion analysis as finding (dense) points correspondences over time
- Optical flow is the **apparent motion of brightness** patterns in the image
- Applications
 - Image and video stabilization
 - Motion-compensated video compression
 - Image registration
 - Action recognition
 - Motion segmentation

Optical flow and motion

- **Motion** is geometric
- **Optical flow** is radiometric
- Optical flow but no motion
 - Moving light sources, lights going on/off, inter-reflection, shadows
- Motion, but no optimal flow
 - Spinning sphere
- Examples: Three percepts
 - **Veridical**: a 2D rigid, flat, rotating ellipse
 - A narrow ellipse oscillating rigidly about its center appears **rigid**
 - A fat ellipse undergoing the same motion appears **nonrigid**
 - The apparent nonrigidity of a fat ellipse is not a visual illusion.
 - The ellipse's motion can be influenced by features not physically connected to the ellipse
 - **Amoeboid**: a 2D, non-rigid smoothly deforming shape
 - **Stereokinetic**: a circular rigid disk rolling in 3D
- Example: flying insects and birds
 - Balance the speeds of motion of the images of the two walls
 - Adjust speed to hold constant the optical flow in the vicinity of the target
 - Approach speed decreases as the target is approached and reduces to zero at the point of touch down
 - No need to estimate the distance to the target at any time

Aperture problem



- Without distinct features to track, the true visual motion is ambiguous
- Locally, one can compute only the component of the visual motion in the direction perpendicular to the contour

Visual motion

- **Visual motion** is determined when there are distinct features to track, provided
 - The features can be detected and localized accurately
 - The features can be correctly matched over time

Motion as matching

| Representation | Result is . . . |
|--------------------------------------|---------------------|
| Point/feature based | (very) sparse |
| Contour based | (relatively) sparse |
| (Differential) gradient based | dense |

Optical flow **constraint equation**

- Consider image intensity being a function of time $I(x, y, t)$.
- Using chain rule, $\frac{dI}{dt} = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t$.
- **2D velocity space**: the space of all points $(\frac{dx}{dt}, \frac{dy}{dt})$.

- It is the motion field, a vector at every point in the image
- If $\frac{dl}{dt} = 0$, we get the optical flow constraint equation $I_x u + I_y v + I_t = 0$.
 - $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$.
- **Brightness constancy assumption**: brightness of the point moving through image sequence remains the same
- For a small space-time step, the brightness of a point in the scene remains the same
- If the time step is small, we can linearize the intensity function
- Computing:
 - Spatial derivate
 - Forward difference
 - Sobel filter
 - Scharr filter
 - Temporal derivative
 - Frame differencing
 - Optical flow
 - $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ need to be solved.
 - Equation determines a **straight line** in velocity space

Lucas-Kanade

- A dense method to compute the motion at every location in an image
- Observations
 - The 2D motion at a given point has two degrees of freedom
 - The partial derivatives provide one constraint
 - The 2D motion cannot be determined locally from I_x, I_y, I_t alone
- Idea: obtain additional local constraint by computing the partial derivatives in a window centered at the given $[x, y]$.
- Assumption: nearby pixels will likely have the same optical flow
- Let $[x_1, y_1]$ be the center, and $[x_2, y_2]$ be a point in the window, then $\begin{pmatrix} u \\ v \end{pmatrix} =$

$$-\begin{pmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \end{pmatrix}^{-1} \begin{pmatrix} I_{t_1} \\ I_{t_2} \end{pmatrix}.$$

- Considering all points, we can get $Av = b$.

$$\circ A = \begin{pmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \dots & \dots \\ I_{x_n} & I_{y_n} \end{pmatrix}.$$

$$\circ b = -\begin{pmatrix} I_{t_1} \\ I_{t_2} \\ \dots \\ I_{t_n} \end{pmatrix}.$$

$$\circ v = (u, v)^T.$$

- A window size is chosen such that (u, v) is constant

$$\circ \text{The least squares solution is: } \bar{v} = (A^T A)^{-1} A^T b.$$

- A window size is chosen such that $A^T A$ has rank 2 and is invertible

$$\circ \text{Here, } A^T A = C = \begin{pmatrix} \sum_{p \in P} I_x I_x & \sum I_x I_y \\ \sum I_y I_x & \sum I_y I_y \end{pmatrix}.$$

Horn-Schunck optical flow

- $\min_{u,v} \sum_{i,j} E_s(i,j) + \lambda E_d(i,j)$.
 - E_s is smoothness.
 - E_d is brightness constancy.
 - λ is weight.
- Brightness constancy: $E_d = (I_x u_{ij} + I_y v_{ij} + I_t)^2$.

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

Image classification

September 27, 2021 11:16 AM

Problem:

- Assign new observations into one of a fixed set of categories (classes)

Key idea

- Build a model of data in a given category based on observations of instances in that category

Classifier

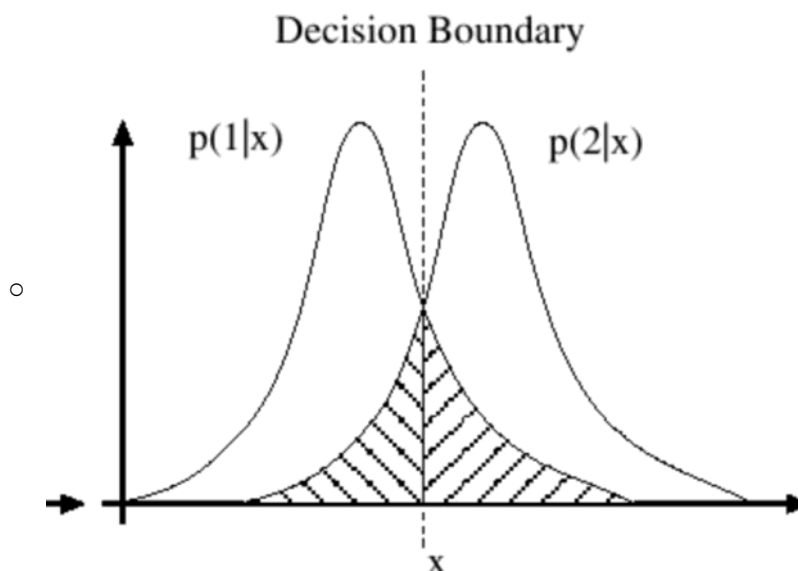
- A procedure that accepts as input a set of features and outputs a class **label**
- Types
 - Binary
 - Multi-class
- Build a classifier using a training set of labelled examples $\{(x_i, y_i)\}$
 - Each x_i is a feature vector
 - y_i is class label

Classification

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images
- Issue:
 - Classification assumes that incoming image belongs to one of k classes
 - In practice, it is impossible and useless to enumerate all relevant classes in the world
- Solution: create an unknown or irrelevant class

Bayes rule

- Let c be the class label and x be the measurement (evidence)
- Then the **posterior probability**: $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$.
 - Class-conditional probability (likelihood): $P(x|c)$
 - Prior probability $P(c)$
 - Unconditional probability (marginal likelihood) $P(x)$
- Bayes' risk
 - Errors may be inevitable
 - The minimum risk (shaded) is called the Bayes' risk



Discriminative vs generative

- Finding a **decision boundary** is not the same as modeling a conditional density
- The quality of the classifiers depends only on how well the boundary is positioned

Loss functions

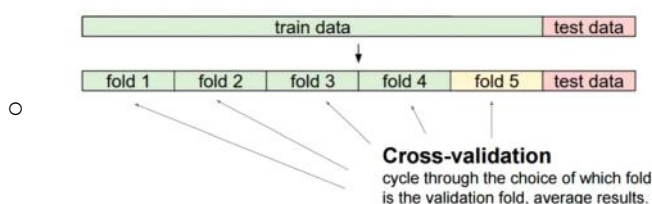
- Loss
 - Some errors may be more expensive than others
 - Let $L(1 \rightarrow 2)$ be the loss caused by calling 1 a 2
- **Total risk** of using classifier s is:
 - $R(s) = P(1 \rightarrow 2|s)L(1 \rightarrow 2) + P(2 \rightarrow 1|s)L(2 \rightarrow 1)$.
- Two class classification, classify x as:
 - 1 if $P(1|x)L(1 \rightarrow 2) > p(2|x)L(2 \rightarrow 1)$
 - 2 if $P(1|x)L(1 \rightarrow 2) < p(2|x)L(2 \rightarrow 1)$
- Decision boundary: points where the loss is the same for either class

Errors and overfitting

- **Training error**: the error a classifier makes on the training set
- **Testing error**: the error the classifier makes on the unseen testing set
 - We want to minimize this
- **Overfitting**
 - Phenomenon that causes testing error to be worse than training error
 - Model is too complex and fits irrelevant characteristics (noise) in the data
 - Classifiers have small training error may not necessarily have small testing error
- Underfitting
 - Model is too simple to represent all the relevant class characteristics

Cross validation

- We cannot reliably estimate the error rate of the classifier using the training set
- **Validation** set:
 - Split some training data
 - Try out what hyperparameters work best on test set
- Cross-validation
 - Performs multiple splits and averaging the errors over all splits



Confusion matrix

- When evaluating a multi-class classifier, it may be useful to know how often certain classes are often misclassified as others
- Def: a table whose (i, j) entry is the frequency an item of true class i was labelled as j by the classifier

Receiver operating characteristics (ROC)

- ROC curves plot trade-off between false positives and false negatives

Classifier strategies

- **Parametric**
 - Model driven
 - Parameters are learned from training examples
 - New data points are classified by the learned model
 - **Properties**
 - Fast, compact

- Flexibility and accuracy depend on model assumptions
- **Non-parametric**
 - Data driven (data is the model)
 - New data points are classified by comparing to the training examples directly
 - Properties
 - Slow
 - Highly flexible decision boundaries

Nearest neighbor classifier

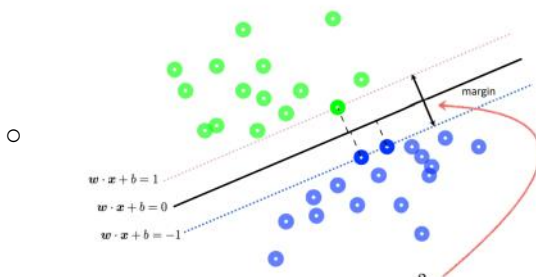
- Given a new data point, assign the label of nearest training example in feature space
- **k-Nearest Neighbor (kNN)** classifier
 - Can gain robustness to noise by voting over multiple neighbors
 - Given a new data point, find the k nearest training example. Assign the label by **majority vote**
 - Works well if the distance measure correctly weights the various dimensions
 - For large data sets, as k increases, kNN approaches optimality in terms of minimizing probability of error
 - Decision boundaries respond to local clusters where one class dominates

Linear classifier

- Define a score function: $f(x_i, W, b) = Wx_i + b$.
 - x_i is the image feature vector
 - W is the weight parameter
 - b is the bias parameter

Support Vector Machines(SVM)

- Try to obtain the decision boundary directly
- Parametrized as a **separating hyperplane** in feature space
- Choose the hyperplane that is as far as possible from each class
 - Maximize the distance to the closest point from either class
- Uses a linear classifier
 - Find hyperplane w such that the gap between parallel hyperplanes $\frac{2}{\|w\|}$ is maximized



Difficulties

- Intra-class variation
- Viewpoint
- Illumination
- Clutter
- Occlusion

Bag of words representation

- **Word**: a local image patch - described by descriptor (SIFT)
- A **codebook of visual words** contains representative local patch descriptors
 - Can be constructed by clustering local descriptors in training images
 - Deals well with occlusion
 - Scale invariant
 - Rotation invariant
 - Spatial information of local features can be ignored for object recognition (verification)

- The **bag of words** model accumulates a histogram of occurrences of each visual word
 - **Dictionary learning**: learn visual words using clustering
 - Extract features (SIFT) from images
 - Learn visual dictionary (K-means clustering)
 - **Encode**: build bags of words vectors for each image
 - **Quantization**: image features get associated to a visual word (nearest cluster center)
 - **Histogram**: count the number of visual word occurrences
 - **Classify**: train and test data using BOWs
- **Algorithm**
 - Initialize an empty K-bin histogram, K = the number of code words
 - Extract local descriptors from the image
 - For each local descriptor x
 - Quantize x to its closest codeword $c(x)$
 - Increment the histogram bin for $c(x)$
 - Return the histogram
 - Classify the histogram using a trained classifier (SVM, kNN)

Spatial pyramid

- The bag of words representation does not preserve any spatial information
- Spatial pyramid is one way to incorporate spatial information into descriptor
- partitions the image and counts visual words within each grid box, repeated at multiple levels

Vector space model

- A document (datapoint) is a vector of counts over each word (feature)
 - $v_d = [n(w_{1,d}), n(w_{2,d}), \dots, n(w_{T,d})]$.
 - A histogram over words
- Similarity between two documents
 - Use cosine similarity $d(v_i, v_j) = \cos \theta = \frac{v_i \cdot v_j}{|v_i| |v_j|}$.

K-means clustering

- Assume the cluster centers are known. Assign each point to the closest cluster center
- Assume the assignment of points to clusters is known. Compute the best cluster center for each cluster (as the mean)
- Initialization dependent and converges to a local minimum

Vector of locally aggregated descriptors (VLAD)

- Instead of incrementing the histogram bin by 1, we **increment it by the residual vector** $x - c(x)$.
- **Dimensionality: Kd**
 - K : number of codewords
 - d : dimensionality of the local descriptor
- VLAD characterizes **the distribution of local descriptors with respect to the codewords**

Decision tree

- Simple non-linear parametric classifier
- Each internal node is associated with a feature test
- A data point starts at the **root** and recursively proceeds to the child node determined by the feature test, until it reaches a **leaf node**
- Leaf node stores a class label or a probability distribution over class labels
- Learning a decision tree involves selecting an efficient sequence of feature tests
- **Entropy**: $H(S) = -\sum_{c \in C} p(c) \log(p(c))$.
 - C is the set of classes represented in S .
 - $p(c)$ is the empirical distribution of class c in S .
 - It is highest when data samples are spread equally across all classes, and zero when all

data samples are from the same class

- Information gain: $I = H(S) - \sum_{i \in \{children\}} \frac{|S^i|}{|S|} H(S^i)$
 - We want to select the feature test that maximizes the information gain

Random forest

- An ensemble of decision trees
- Randomness is incorporated via training set sampling and/or generation of the candidate binary tests
- Prediction of random forest is obtained by averaging over all decision trees

Better classifier

- Combine multiple classifiers
 - Train an ensemble of independent classifiers and average the predictions
- Boosting
 - Train an ensemble of nonlinear classifiers sequentially
 - Bias subsequent classifiers to correctly predict training examples that previous classifiers got wrong
 - Increase the weight of falsely predicted data to build the next classifier
 - Final boosted classifier is a weighted combination of the individual classifiers

Object detection

November 22, 2021 1:19 PM

Intro

- In image classification, we assumed the image contained a single, central object
- **Task of object detection**: detect and localize all instances of a target object class in an image
 - Put a tight bounding box around the object

Sliding window

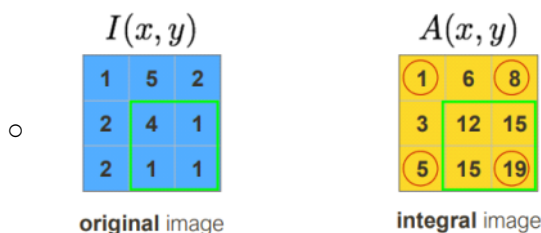
- Train an image classifier
- Slide a **fixed-sized** detection window across the image and evaluate the classifier on each window
- Search over **location, scale and aspect ratio**

Data for classifier

- Basic **image classifier**
 - Works badly for regions/windows
- **Object classifier**
 - Works better
 - Require expensive bounding box annotations
 - Normalization to fixed size

Face detection (Viola Jones)

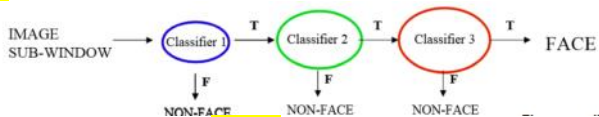
- Use features that are fast to evaluate to reject most windows early
- A **rectangular feature** is computed by summing up pixel values within rectangular regions and then differencing those region sums
 - Integral image speeds up region summation, computing **Haar wavelets is fast**
- **Integral image**
 - $A(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$.
 - Sum of left and above.
 - $A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1)$.



- **Constant time**: doesn't depend on the size of the region. Can avoid scale images, just scale features directly
- **Weak classifier**:
$$h_j(x) = \begin{cases} 1, & \text{if } f_j(x) > \text{threshold } j \\ 0, & \text{otherwise} \end{cases}$$
- Use **boosting** to both select the informative features and form the classifier.
 - Each round chooses a weak classifier that simply compares a single rectangular feature against a threshold
- **Algorithm**
 - Select best filter/threshold combination
 - Normalize the weights $w_{i,j} = \frac{w_{i,j}}{\sum_{j=1}^n w_{i,j}}$
 - For each feature $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$
 - 0: predicted label and true label are the same
 - 1: predicted label and true label are different
 - Choose the classifier h_t with the lowest error ϵ_t

- Reweight examples
 - $w_{t+1,i} = w_{t,i} \beta_t^{1-|h_t(x_i)-y_i|}$, $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
 - When correct, updated to $w_{t+1,i} = w_{t,i} \beta_t^{1-|h_t(x_i)-y_i|}$
 - When incorrect, not changed
- Final strong classifier
 - $$h(x) = \begin{cases} 1, & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0, & \text{otherwise} \end{cases}$$
, $\alpha_t = \log \frac{1}{\beta_t}$
 - A weighted linear combination of the T weak classifiers, weights are **inversely proportional** to the training errors
- **Observations**
 - On average only 0.01% of all sub-windows are positive
 - Equal computation time is spent on all sub-window
 - We want to spend most time only on potentially positive windows
- A simple 2-feature classifier can achieve almost 100% detection rate, with 50% false positive rate
 - Can act as the 1st layer of a series to filter out most negative windows
 - 2nd layer with more features to tackle harder negative windows

Cascade classifiers

- 
- To make detection **faster**, features can be reordered by increasing complexity of evaluation and the thresholds adjusted so that the early tests have few or no false negatives
 - Windows rejected by early tests can be discarded quickly

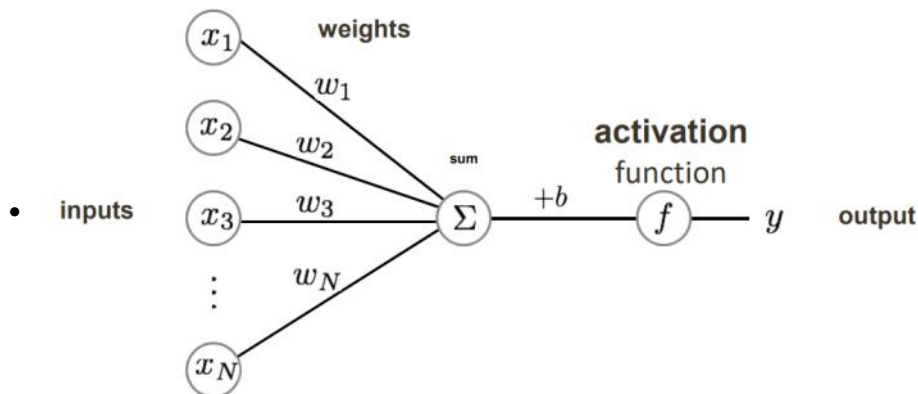
Object proposal

- There are a lot of possible windows for sliding
- Object proposal algorithms generate a short list of regions that have generic object-like properties
 - Regions that are likely to contain some objects instead of background
- Object detectors considers the candidate regions only
- **Object-ness score based on cues**
 - Objects are unique within the image and stand out as salient
 - Objects have strong contrasting appearance from surroundings
 - Objects have a well-defined closed boundary in space
- **Multiscale saliency**
 - Favors regions with a unique appearance within the image
- **Color contrast**
 - Favors regions with a contrasting color appearance from immediate surroundings
- **Super-pixels straddling**
 - Favors regions with a well-defined closed boundary
 - Measures the extent to which super-pixels obtained by image segmentation contain pixels both inside and outside of the window

Neural network

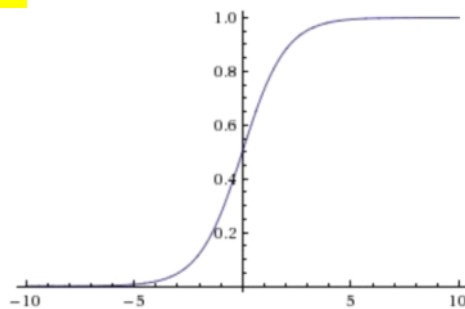
November 24, 2021 10:28 AM

Neuron



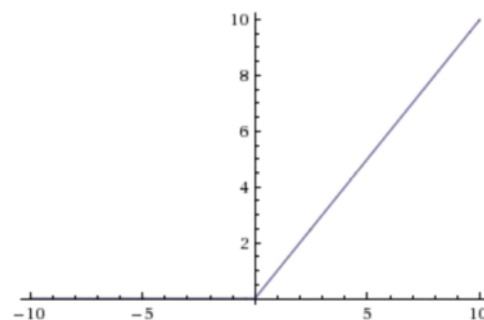
- $y = f(\sum_{i=1}^N w_i x_i + b) = f(WX + b)$.
- The basic unit of computation in a neural network
- Accepts input, computes weighted sum and applies an **activation function (non-linearity)** to the sum
- Common activation function: sigmoid, ReLU

- **Sigmoid**

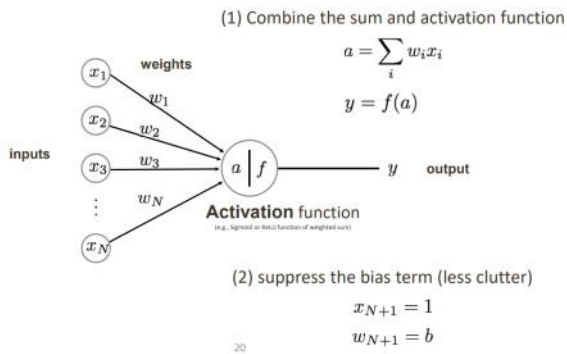


- $f(x) = \frac{1}{1+e^{-x}}$.
- Common in early neural networks
- Saturated firing rate of neurons

- **ReLU**

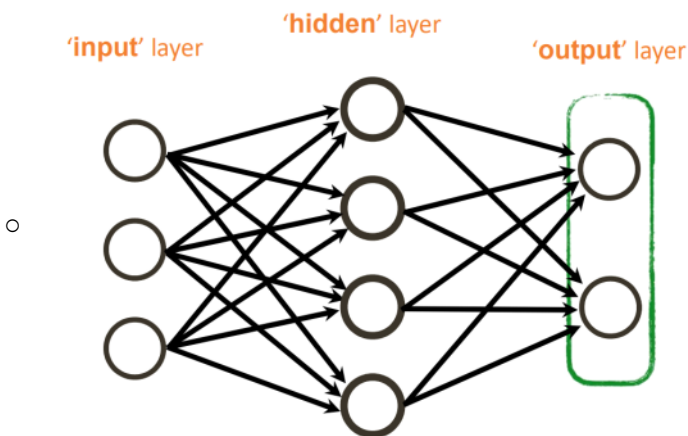


- $f(x) = \max(0, x)$.
- Accelerate convergence during learning
- Used in the most recent neural networks



Neural network

- A collection of connected neurons
- A neural network comprises neurons connected in an acyclic graph
 - Outputs of neurons can be inputs to other neurons
 - Neural networks contain multiple layers of neurons
- **Multi-layer perceptron (MLP)**



- Hidden unit: classifier/feature
- Note: each neuron will have its own vector of weights and a bias
- More layers
 - More complex function mapping
 - More efficient due to distributed representation

Activation function

- Non-linear activation is required to make the neural net a **universal function approximator**
- With ReLU: linear spline approximation to any function
- **Optimization**: finding slopes and transitions of linear pieces
- Quality of approximation depends on the number of linear segments

Universal approximation theorem

- Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity
- Revised: a network of infinite depth with a hidden layer of size $d + 1$ neurons, where d is the dimension of the input space, can approximate any continuous function
- Further revised: ResNet with a single hidden unit and infinite depth can approximate any continuous function

Loss

- When training a neural network, the final output will be some loss (error) function
- **Cross entropy loss**
 - $L_i = -\log\left(\frac{e^{f y_i}}{\sum_j e^{f y_j}}\right)$.
 - Loss for i -th training example with true class index y_i , f_{y_j} is the j -th element of the vector of

class scores coming from the neural net

- $\frac{e^{f y_i}}{\sum_j e^{f y_j}}$ is the softmax function multi-class classifier

Back propagation

- Compute the gradient of the loss with respect to the network parameters so that we can incrementally adjust the network parameter
- Parameters of a neural network are learned using back propagation, which computes gradients via recursive application of the chain rule

Gradient descent

- Start from random value of W_0, b_0 .
- For $k = 0$ to max number of iterations
 - Compute gradient of the loss with respect to previous parameters
 - $\nabla L(W, b)_{W=W_k, b=b_k}$.
 - Re-estimate the parameters
 - $W_{k+1} = W_k - \lambda \frac{\partial L}{\partial W}$.
 - $b_{k+1} = b_k - \lambda \frac{\partial L}{\partial b}$.
 - λ is the learning rate.
- **Stochastic** gradient descent
 - For large datasets computing the sum is expensive
 - Compute instead approximate gradient with mini-batches of much smaller size
- Numerical differentiation

We can approximate the gradient numerically, using:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ij}} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W} + h \mathbf{1}_{ij}, \mathbf{b}) - \mathcal{L}(\mathbf{W}, \mathbf{b})}{h} \quad \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial b_j} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W}, \mathbf{b} + h \mathbf{1}_j) - \mathcal{L}(\mathbf{W}, \mathbf{b})}{h}$$

- Even better, we can use central differencing:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ij}} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W} + h \mathbf{1}_{ij}, \mathbf{b}) - \mathcal{L}(\mathbf{W} - h \mathbf{1}_{ij}, \mathbf{b})}{2h} \quad \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial b_j} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W}, \mathbf{b} + h \mathbf{1}_j) - \mathcal{L}(\mathbf{W}, \mathbf{b} - h \mathbf{1}_j)}{2h}$$

Symbolic differentiation

- Input function is represented as computational graph (symbolic tree)
 - Each node is an input, intermediate or output variable
- Difficult to deal with piece-wise functions or complex functions

Automatic differentiation

- Interleave symbolic differentiation and simplification
- Apply symbolic differentiation at the elementary operation level, evaluate and keep intermediate results

FC layer

- Spatial correlations are generally local
- Waste of resources
- Not enough data

LC layer

- Stationarity: statistics is similar at different locations

Convolutional layer

- Share the same parameters across the locations
- Interpretation
 - Multiple neurons that share weights
 - One neuron applied as convolution by shifting
 - $\sigma\left(\sum \sum W_{i,j} I(i,j) + b\right) = \sigma(W^T x + b)$.
 - Learns multiple filters

- Convolve the filter with the image (slide over the image spatially, computing dot products)
- The number of neurons in a layer
 - Determined by depth and stride
 - **Depth**: controls number of neurons that connect to the same region of the input layer
 - **Depth column**: a set of neurons connected to the same region
 - **Stride**: controls spatial density
 - Affected by zero-padding
 - With padding, we can achieve no shrinking
 - Shrinking quickly doesn't work well

Convolutional neural network

- Learning a hierarchy of filters
- As we go deeper in the network, filters learn and respond to increasingly specialized structures

Pooling layer

- Make detection spatially invariant (insensitive to position of the object in the image)
- By pooling response over a spatial locations we gain robustness to position variations
- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently
- Max/avg pooling

Terminology

- **Network structure**: number and types of layers, forms of activation functions, dimensionality of each layer and connections (defines computational graph)
 - Generally kept fixed, requires some knowledge of the problem and NN to sensibly set
 - Deeper is better
- **Loss function**: objective function being optimized
 - Requires knowledge of the nature of the problem
 - Softmax, cross entropy
- **Parameters**: trainable parameters of the network, including weights/biases of linear/fc layers, parameters of the activation functions
 - Optimized using SGD or similar
- **Hyper-parameters**: parameters, including for optimization, that are not optimized directly as part of training

Multivariate regression

- Input: feature vector $x \in \mathbb{R}^n$.
- Output: output vector $y \in \mathbb{R}^m$.
- Neural network (input + hidden) $f(x, \theta) : \mathbb{R}^n \rightarrow \mathbb{R}^k$.
 - With tanh: $f \in [-1,1]$.
 - With ReLU: $f \geq 0$.
- Network output layer:
 - $\hat{y} = g(x; W, b) = Wf(x; \theta) + b : \mathbb{R}^k \rightarrow \mathbb{R}^m$.
- Loss: $L(y, \hat{y}) = |y - \hat{y}|^2$. Similarity between two distributions

Binary classification

- Input + hidden:
 - With sigmoid: $f \in [0,1]$.
 - Threshold hidden output $\hat{y} = 1$ if $f > 0.5$.
 - Problem: not differentiable, probabilistic interpretation maybe desirable
- Output
 - interpret sigmoid output as probability: $p(y = 1) = f(x; \theta)$
 - Can interpret the score as the log-odds of $y = 1$
- Loss: Loss for sigmoid: $L = -y \log f - (1 - y) \log(1 - f)$.

Multiclass classification

- Input + hidden: ReLU
- Output: softmax
- Loss: $L = -\sum y_i \log y_i$.

Training

- Initialize parameters of all layers
- For a fixed number of iterations or until convergence
 - Form mini-batch of examples (randomly chosen from a training dataset)
 - Compute forward pass to make predictions for every example and compute the loss
 - Recursively calling forward for each intermediate layer along computational graph
 - Compute backwards pass to compute the gradient of the loss with respect to each parameter for each example
 - Traversing computing graph in reverse order and composing intermediate gradients using chain rule
 - Update parameters of all layers, by taking a step in the negative average gradient direction

Inference/prediction

- Compute forward pass with optimized parameters on test examples

Monitoring learning

- Visualizing the training loss
- Big gap: overfitting
 - Increase regularization
- No gap: underfitting
 - Increase model capacity

Convolutional layer summary

- Input volume: $W_i \times H_i \times D_i$.
 - For mini batch: $N \times W_i \times H_i \times D_i$.
- Hyperparameters
 - Number of filters: $K \in \{32,64,128,256,512\}$.
 - Spatial extent of filters $F \in \{1,3,5, \dots\}$.
 - Stride of application $S \in \{1,2\}$.
 - Zero padding: $P \in \{0,1,2\}$.
- Output volume: $W_o \times H_o \times D_o$.
 - For mini batch: $N \times W_o \times H_o \times D_o$.
 - $W_o = \frac{W_i - 2\lfloor F/2 \rfloor + 2P}{S} + 1$.
 - $H_o = \frac{H_i - 2\lfloor F/2 \rfloor + 2P}{S} + 1$.
- # learnable parameters: $K(FFD_i + 1)$.

FC layer of CNNs

- Stretch input to linear $k \times 1$ vector.
- Each neuron looks at the full input volume and activate

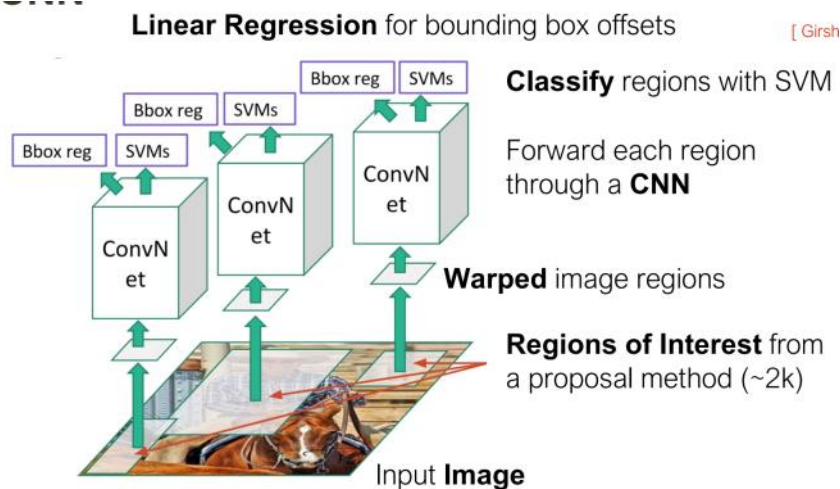
Pooling layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently
- No learned parameters
- Receptive field
 - Filter $K \times K$, stride 1.
 - Pool of size $P \times P$.
 - Each unit looks at $(P + K - 1)(P + K - 1)$ area.
- Summary
 - Input volume: $W_i \times H_i \times D_i$
 - Hyperparameters

- Spatial extent of filters: F
- Stride of application S
- Output volume $W_o \times H_o \times D_o$
 - $W_o = \frac{W_i - F}{S} + 1$.
 - $H_o = \frac{H_i - F}{S} + 1$.
 - $D_o = D_i$.
- # learnable parameters: 0

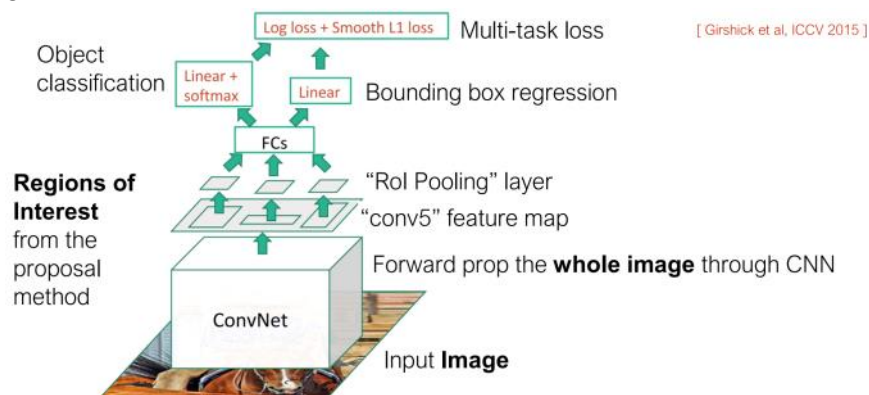
Computer vision applications

- **Categorization**
 - For each image, predict which category it belongs to out of a fixed set
- **Detection**
 - As a **regression** problem: Each image needs a different number of outputs
 - As a **classification** problem: apply CNN to many different crops in the image and CNN classifies each patch as object or background
 - Need to apply CNN to many patches in each image
 - **Region proposals**
 - Find image regions that are likely to contain objects
 - ◆ Works by looking at histogram distributions, region aspect ratio, closed contours, coherent color
 - Fast to run
 - Get true object regions to be in as few top K proposals as possible
 - R-CNN



- Extract promising candidate regions using an object proposals algorithm
- Resize each proposal window to the size of the input layer of a trained CNN
- Input each resized image patch to the CNN
- Final FC layer output can be used as input features to a trained SVM

- Fast R-CNN



- Performance dominated by region proposals

- **Segmentation**

- Semantic segmentation: label every pixel with a category label

- Sliding window: very inefficient, no reuse of computations for overlapping patches
- Fully convolutional CNNs
 - Design a network as a number of convolutional layers to make predictions for all pixels at once
 - Convolutions at the original image scale will be expensive
 - Design a network as a number of convolutional layers with down sampling and up sampling inside the network
 - Down sampling: pooling
 - Up sampling:
 - ◆ nearest neighbor
 - ◆ bed of nails: append zeros
 - ◆ Max un-pooling: remember which element was max when using max pooling, then use the positions when up sampling
- Instance segmentation
 - Differentiate category and instances

Neural image captioning

- Use RNNs

Grouping

2021年12月3日 12:59

Human vision

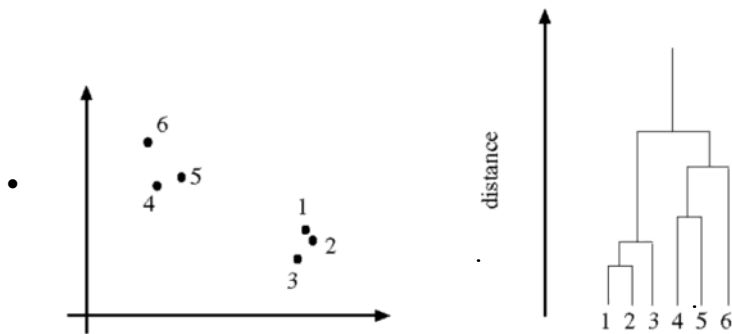
- Similarity
- Symmetry
- Common fate
- Proximity

Clustering

- Useful to **group** together **image regions** with similar appearance
 - Image compression
 - Approximate nearest neighbor search
 - Base unit for higher-level recognition tasks
 - Moving object detection in video sequences
 - Video summarization
- **Clustering** is a set of techniques to try to find components that belong together
 - Unsupervised learning
- Approaches
 - **Agglomerative** clustering
 - Each data point starts as a separate cluster. Clusters are recursively merged
 - Algorithm
 - Make each point a separate cluster
 - Until the clustering is satisfactory
 - Merge the two clusters with the smallest inter-cluster distance
 - **Divisive** clustering
 - The entire data set starts as a single cluster. Clusters are recursively split
 - Algorithm
 - Construct a single cluster containing all points
 - Until the clustering is satisfactory
 - Split the cluster that yields the two components with the largest inter-cluster distance
- **Inter-cluster** distance
 - Distance between the closest members of C_1 and C_2 .
 - Single-link clustering.
 - $\min d(a, b), a \in C_1, b \in C_2$.
 - Distance between the farthest members of C_1 and C_2 .
 - Complete-link clustering
 - $\max d(a, b), a \in C_1, b \in C_2$.
 - Average distances between members of C_1 and C_2 .
 - Group average clustering
 - $\frac{1}{|C_1||C_2|} \sum_{a \in C_1} \sum_{b \in C_2} d(a, b)$.

Dendrogram

- Generate a hierarchy of clusters and visualized with a dendrogram



K-means clustering

- Assume we know how many clusters there are in the data, denote by K .
- Each cluster is represented by a cluster center, mean
- Objective: minimize the representation (quantization) error in letting each data point be represented by some cluster center
 - $\min \sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{ith cluster}} |x_j - \mu_i|^2 \right\}$.
- Initialization: choose K random cluster centers
- Alternates between two steps
 - Assume the cluster centers are known. Assign each point to the closest cluster center
 - Assume the assignment of points to clusters is known. Compute the best center for each cluster, as the mean of the points assigned to the cluster
- Converges to a local minimum of the objective function
 - Results are initialization dependent
- Advantages
 - Always converges
 - Easy to implement
- Disadvantages
 - Number of classes K needs to be given as input
 - Algorithm doesn't always converge to the globally optimal solution
 - Limited to compact/spherical clusters

Segmentation by clustering

- Segmentation makes use of texture, corners, lines, geometry
- Agglomerative clustering
 - Represent the image as a weighted graph
 - Any pixels that are neighbors are connected by an edge
 - Each edge has a weight that measures the similarity between the pixels
 - Can be based on color, texture
 - Low weights-similar,
 - High weights-different
 - Segment the image by performing an agglomerative clustering guided by the graph
 - $d(C_1, C_2) = \min w(v_1, v_2)$.
 - **Internal difference**: largest weight in the minimum spanning tree of the cluster $M(C)$.
 - $int(C) = \max_{e \in M(C)} w(e)$.
 - For smaller clusters, this doesn't work
 - Add a term $\tau(C) = \frac{k}{|C|}$, we have $int(C) + \tau(C)$ as the difference.
 - $MInt(C_1, C_2) = \min \left(int(C_1) + \tau(C_1), int(C_2) + \tau(C_2) \right)$ for two clusters.

Algorithm: (Felzenszwalb and Huttenlocher, 2004)

Make each point a separate cluster.

Sort edges in order of non-decreasing weight so that $w(e_1) \leq w(e_2) \leq w(e_3) \dots \leq w(e_r)$

For $i = 1$ to r

 If both ends of e_i lie in the same cluster

 Do nothing

 • Else

 One end is in cluster C_l and the other is in cluster C_m

 If $d(C_l, C_m) \leq MIInt(C_l, C_m)$

 Merge C_l and C_m

Report the remaining set of clusters.