

Lecture 1 Introduction

Computer Vision: understand the meaning of images with computer

Applications: measurement device, recognize semantic content, image manipulation, robotic...

Lecture 2 Image Basics and Filters

Pinhole Camera (aperture: size of pinhole), Digital Camera

Resolution: a sampling parameter (DPI: dots per inch)

Color: psychological property, not physical property Linear color spaces: defined by 3 primaries, RGB; Nonlinear color spaces: HSV (Hue, Saturation, Value)

Histogram: distribution of gray levels in the image

Noise type: Salt and pepper (random black and white); Impulse (random white); Gaussian

How to denoise? Correlation filter: $G[i, j] = \sum_{u, v} H[u, v]F[i + u, j + v] = H \otimes F$

Boundary issues: 1. Output size full / same / valid 2. Extrapolate out of bounds

Gaussian filter: approximation of 2d Gaussian function $h(u, v) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right)$ (normalize), size of kernel and σ control the amount of smoothing; remove high-frequency components

Median filter: remove spikes, edge preserving, good for impulse, salt and pepper noise

Convolution: flip the filter in both dimensions, apply cross-correlation (commutative)

$$G[i, j] = \sum_{u, v} H[u, v]F[i - u, j - v] = H \star F$$

Filter application: as templates for template matching (obvious maximum)

Lecture 3 Edges

Origin of edges: normal / depth / color / illumination discontinuity

Derivative THM: $\frac{\partial}{\partial x}(h * f) = \frac{\partial h}{\partial x} * f$ Finding edges = finding peak of derivatives

Seam carving: energy of a point $E(i, j) = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$, define vertical seam s as a 8-connected path, minimize energy of the seam $E(s) = \sum_{(i, j) \in s} E(i, j)$ using DP (preserve most interesting)

Edge detector: good detection, good localization, single response

Sobel Edge Detector: Sobel operator $G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$, $G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$, gradient

magnitude $G = \sqrt{G_x^2 + G_y^2}$, gradient direction $\theta = \arctan\left(\frac{G_y}{G_x}\right)$

Problem: poor localization; thresholding value favors certain directions over others

Canny Edge Detector: 1. compute magnitude G : derivative of Gaussian (DoG) filter, $G = \sqrt{G_x^2 + G_y^2}$ (large σ for large scale edges, small σ for fine features)

2. Non-maximum suppression: maximum of its neighbor in the gradient direction (8 candidates)

3. Hysteresis thresholding: < low, not edge; > high, strong edge; middle, edge if connect to strong

Learning-based Edge Detector: Challenge: combine boundary cues like brightness, color, texture

Goal: Learn posterior probability $p_b(x, y, \theta)$ from local information using human segmentations

Hough transform: find pixels that make up straight lines from edge

1. Quantize linear parameter space (a, b) by dividing into cells

2. For each pair of points $(x_1, y_1), (x_2, y_2)$ detected as edge, find (a', b') in the parameter space

3. Increment the cell (a', b') by 1, find the cell with the maximum value

Lecture 4 Keypoints and Descriptors

Image Features: Tracking, Shot analysis, Scene understanding, Instance matching, Pattern finding

Keypoint: detection: where to extract; description: what to extract; matching: how to compare

Desired local features: Repeatability, Saliency (distinctive), Compactness and efficiency, Locality

Harris corner detector: corner as distinctive interest points; $M = \sum w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$; compute

cornerness score $f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$, threshold $f > t$; non-maximum suppression (local maxima in 3x3)

Property: rotation invariant but not scale invariant!

Automatic scale selection: find solution that gives local maximum of f in both position and scale

"blob" detector -- Laplacian of Gaussian (LoG) $\nabla^2 G = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) G$

Characteristic scale: produce peak of Laplacian response

1. compute $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$, $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
2. $(x, y, \sigma) = \operatorname{argmin} \nabla_{\text{norm}}^2 L = \operatorname{argmin} \sigma^2 (\nabla^2 G) * I \approx \operatorname{argmin} D(x, y, \rho)$ where DoG $D(x, y, \rho) = (G(x, y, k\rho) - G(x, y, \rho)) * I(x, y)$ for $\rho = \{\sigma, \dots, k^{s-1}\sigma\}$, $k = 2^{1/s}$ is approximation of Log
3. NMS pruning (local maximum compared to 26 neighbors)

SIFT (Scale Invariant Feature Transform): Fast and efficient, real-time, invariant to rotation and scale, partially invariant to illumination, viewpoint, occlusion, clutter.

1. Do scale invariant point detection, record scale ρ , take the Gaussian-blurred image $I(x, y, \rho)$
2. Compute gradient $G_x = \frac{\partial I}{\partial x}$, $G_y = \frac{\partial I}{\partial y}$ in the neighborhood (size of neighbor in proportion to ρ)
3. Compute the dominant orientation of each keypoint: weight $|\nabla I(x, y)| \cdot G_{1.5\rho}(d)$
4. Compute 128 dim descriptor (neighbor into 4×4 grid, 8 relative orientations per grid cell)
5. Post processing: normalize, clip to 0.2, normalize

Lecture 5 Feature Matching and Transformation Fitting

How to avoid ambiguous match? Threshold ratio $\alpha = \text{dist_to_best} / \text{dist_to_second_best}$

Feature-based transformation fitting: extract feature \rightarrow compute match \rightarrow hypothesize & verify

Linear regression: fitting model to features, $\min \sum_i \text{residual}(x_i, M)$

Alignment: fitting transformation between pair of features, $\min \sum_i \text{residual}(T(x_i), x'_i)$

Image alignment: direct (pixel-based) alignment, feature-based alignment

General transformation: affine (2×3), projective (homography 3×3 , 8 parameters)

RANSAC: repeat draw s points randomly, fit model, count inliers, return answer with most inliers

Pros: simple & general Cons: tune params, not work in low inlier ratios

Image Mosaic: find homography by alignment, warp image, blend images

Lecture 6 Camera Models

Images: capture geometric and photometric information, complex 3D-2D relationships

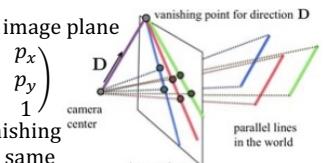
3D to 2D projection: use a linear 3D to 2D projection matrix (perspective, orthographic, scaled orthographic, paraperspective) Approx using camera models (pinhole model)

Modeling Projection: XOY as image plane, Z axis called optical / principal axis

Principal point: intersection of Z axis and image plane

Focal length: distance from the pinhole / principal point to the image plane

Under homogeneous coordinates: intrinsic matrix $K = \begin{pmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{pmatrix}$



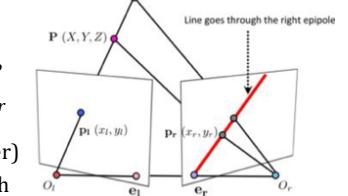
Projection Properties: 1. All parallel lines intersect at same vanishing

point (camera center \rightarrow vanishing point is D) 2. All lines on the same

3D plane have the same vanishing line 3. Parallel planes have the same vanishing line.

Orthographic Projection: simple drop Z coordinate, parallel lines still be parallel.

Projection Equations: $q = \begin{pmatrix} ax \\ ay \\ a \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$, $P = K(R | t)$, $R = (u^T, v^T, w^T)$, t is translation vector



Lecture 7 Depth from Stereo

Problem: Two calibrated cameras, diff by T , how to find depth?

Parallel case: 1. For each point $p_l = (x_l, y_l)$ in left image, find $p_r = (x_r, y_r = y_l)$ in right image (Operate on patch: SSD (Sum of Squared Differences); Normalized Correlation; Train a classifier)

2. Compute disparity $d = x_l - x_r$, depth $Z = \frac{fT}{d}$, f is focal length

Note: 1. larger patch less detail but smooth 2. Better with energy minimization on the top (MRF)

General case: For each point p_l , search p_r on epipolar line l_r

Note: Map from p_l to l_r , can be described by a single 3×3 matrix F (fundamental matrix)

$l_r = e_r \times p_r = e_r \times (H_\pi p_l) = F p_l$ (a line denoted as (a, b, c) is cross product of two points)

Simple solution: $p_r^T l_r = p_r^T F p_l = 0$, $O(1)$ (p_l, p_r) correspondence is enough

What can we do after getting F ?

1. Rectify: Align epipolar lines to be horizontal, find H rectify both images to be parallel

2. Compute relative pose: projection matrices $P_l = [I | 0]$, $P_r = [[e_r] \times F | e_r]$, where $e_r^T F = 0$.

Structure from motion: min reprojection err $E(P, X) = \sum_{i,j} \text{dist}(x_{ij}, P_i X_j)$ via bundle adjustment

Lecture 8 Classification

Problem: Semantic Gap

Challenges: viewpoint variation, background clutter, illumination, occlusion, deformation, intraclass variation

k Nearest Neighbor: cross validation, pixel distance not informative

Linear Classifier: 3 viewpoints (algebraic, visual, geometric: hyperplanes cutting up space)

Visual Bags of Words: originate from texture recognition, bag-of-words models

1. Extract features using regular grid / interest point detector

2. Learn vocabulary using k-means clustering

3. Quantize features to vocabulary, represent image as histogram of visual words (bag of words)

Usage: Treat as feature vector for standard classifier / Cluster BoW vector over image collection

Pyramid + Bag of Words: locally orderless representation at several levels of spatial resolution

Lecture 9 CNN

History: Perceptron, Adaline, Backpropagation, Neocognitron, LeNet, RBM, DBN, AlexNet

CNN out shape: $W_2 = (W_1 - F + 2P)/S + 1, H_2 = (H_1 - F + 2P)/S + 1, \# \text{ of params } F^2 C_1 C_2 + C_2$

Pooling out shape: $W_2 = (W_1 - F)/S + 1, H_2 = (H_1 - F)/S + 1, \# \text{ of params } 0$

Xavier initialization: $W_{ij} = \mathcal{N}(0, \sigma^2 = 1/d_{in})$ **Transfer Learning:** freeze conv, train FC layers

Optimizer: SGD (w/ momentum), Adam, RMSprop **Learning Rate Scheduling**

Improve test error: early stop, ensemble, regularization (dropout, weight decay, data augment)

Lecture 10 Object Detection

3 approaches: find interest points; sliding windows; generate region proposals and classify

HOG Detector: Train: 1. Scan images at all scales and locations

2. Extract features over windows (compute gradients → divide into 8x8 pixels → 9-dim feature vector) → Contrast normalize over overlapping spatial blocks ($\times 4$)

3. Train classifier on latent (bootstrapping: resample negative training images as hard examples)

Inference: scan and extract features from all windows, threshold, NMS by picking highest score

Eval: 1. $\alpha = \text{intersection} / \text{union} > 0.5$ & Multiple detections are considered right only once

2. Precision = correct / all predicted, Recall = correct / all ground truth

Average Precision = area under the precision-recall curve

DPM Model (Deformable Part-based Model): Motivation: objects are composed of parts

Penalty for shift with quadratic function $a(x - v_x)^2 + b(x - v_x) + c(y - v_y)^2 + d(y - v_y)$

$$\text{Score}(I, p_0) = \max_{p_i} \sum_{i=0} F_i \cdot \text{HOG}(I, p_i) - \sum_{i=1} w^i \cdot (dx, dy, dx^2, dy^2)$$

In HOG feature pyramid, compute cross-correlation and deformation cost for each placement

Train: part positions are not annotated; we can only use latent SVM

R-CNN: extract region proposals ($\sim 2k$) → compute CNN features → classify regions

Train: pretrain (and finetune) CNN → train linear predictor for detection

Faster R-CNN: Region Proposal Network to generate proposals → RoI pooling → classification

Other method: YOLO (generate class prob via Conv), SSD (Single Shot MultiBox Detector)

Lecture 11 Segmentation

Group in vision: Goal: gather features that belongs together / obtain representations

Top down (pixels belong together because they are from the same object) vs.

bottom up (pixels belong together because they are similar)

Goal of segmentation: separate image into coherent objects, group together similar-looking pixel

Clustering algorithms: 1. Unsupervised learning 2. Detect patterns in unlabeled data

K-means on intensity (sensitive to init centers / outliers, don't support probabilistic clustering)

Expectation Maximization: soft assignment of points to clusters, update cluster parameters

Segmentation as clustering: intensity, color, intensity + position, texture ...

Mean shift algorithm: seek modes or local maxima of density in feature space

Procedure: repeat shifting center to mass center of current window, merge near windows

Pros: not assume shape on clusters, one parameter, find multiple nodes, general

Cons: Need to select window size; don't scale well to high-dim feature space

Graph-based normalized cuts: node for every pixel, complete graph with affinity weight

$$\text{measuring similarity } \left(\text{e.g. } \exp\left(-\frac{\|x-y\|^2}{2\sigma_d^2}\right) \right)$$

break graph into segments = delete links cross between segments = solve min-cut problem

Actually, minimize normalized cut $Ncut(A, B) = \frac{cut(A, B)}{\text{assoc}(A, V)} + \frac{cut(A, B)}{\text{assoc}(B, V)}$

Pros: general, flexible to weight function, no need to model \mathcal{D}_{data}

Cons: computation expensive, preference for balanced partitions

Semantic Segmentation: 1. Sliding window + CNN (inefficient, no shared features between overlapping patches)

2. Fully CNN: only conv w/o downsampling, predict all pixels at once (convolutions on original image resolution is expensive)

3. Downsample + Upsample: Max unpooling (use indices from max pooling to upsample), Transpose conv

4. Others: U-Net, Dilated Conv, Spatial Pyramid Pooling, DeepLab, Smooth using CRF

Mask R-CNN: CNN + RPN → RoI Align → mask prediction FCN (Classification loss + Bounding-box regression loss)

Lecture 12 Optical Flow

Def: the apparent motion of brightness patterns in the image

Goal: recover image motion at each pixel from optical flow

Assumption: brightness constancy, small motion, spatial coherence

Brightness constancy: $I(x, y, t) = I(x + u_x, y + u_y, t + 1)$, i.e. $\nabla I \cdot u + \frac{\partial I}{\partial t} = 0$

Component of the flow perpendicular to the gradient is not observable (Aperture problem: due to partial observation, only flow component in the gradient direction is observable)

Lucas-Kanade flow: Assume flow is constant in a small window

Solve $Ax = b$ where $A = (I_x \ I_y), b = -I_t$, using least squares, $x = (A^T A)^{-1} A^T b$

$(A^T A)^{-1}$ is good when eigenvalues are not too small / well-conditioned, i.e. corners

Iterative Refinement: Under small motion assumption, $I(x + u_x, y + u_y, t + 1) - I(x, y, t) = 0$ has higher order terms, a polynomial root finding problem, solve using Newton's method.

Iterative Lucas-Kanade Algorithm: estimate velocity at each pixel, update image, repeat

Pyramid for large motion: reduce resolution, compute & upsample flow (coarse-to-fine estimate)

Horn-Schunck method: Flow is a minimizer of global energy function

$$E = \int \left[(I_x u_x + I_y u_y + I_t)^2 + \alpha^2 (\|\nabla u_x\|^2 + \|\nabla u_y\|^2) \right] dx dy \\ I_x (\nabla I \cdot u + I_t) - \alpha^2 \Delta u_x = 0 \quad I_y (\nabla I \cdot u + I_t) - \alpha^2 \Delta u_y = 0$$

Estimate Laplacian $\Delta u_x \approx \bar{u}_x - u_x$ Iterative solution: update u using the computed Δu

Michael Black method: replace quadratic regularization by $\sum_{n \in N(S)} (u_{x,s} - u_{x,n})^2 + (u_{y,s} - u_{y,n})^2$

FlowNet 1.0: correlation layer $c(x_1, x_2) = \sum_o f_1(x_1 + o, f_2(x_2 + o))$, upsampling layer

FlowNet 2.0: pretrain on chairs then finetune on Things3D; stacked architecture; sub-network specializing on small motions; fusion architecture

Motion segmentation: Goal: break into "layers", each has coherent (affine) motion

Affine motion: $u_x = ax + by + c, u_y = dx + ey + f$, solve each window by minimizing

$\sum (\nabla I \cdot u + I_t)^2$, cluster using K-means in affine params space

Lecture 13 Activity Recognition

Datasets: 1. Video Classification: UCF101 (YouTube video, 101 action classes), HMDB (Human Motion Database, 51 action classes, 6849 videos), Sports-1M (487 classes), YouTube-8M

2. Atomic Actions: Charades (RGB + Optical Flow features, action classification & sentence prediction), Atomic Visual Actions (AVA, 3s segments, fine-grained pose and object interactions, but no annotations about objects), Moments in Time (MIT, 3s, sound, balanced but single label)

3. Video description: M-VAD and MPII-MD, LSMDC (movie description, retrieval, and QA)

Ability: sequence modeling, temporal reasoning, focus on action recognition

Pre-DL: 1. Local feature: HOG + HOF (Histogram of Optical Flow) 2. Trajectory: Motion Boundary Histograms, dense trajectories 3. Ways to aggregate features: Bags of Visual Words, Fisher Vector

Representing Motion: Optical Flow / Trajectory Stacking

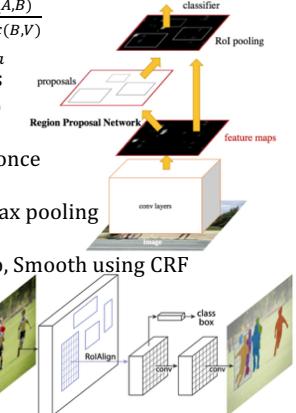
DL: 1. Single Stream Network: direct conv, lack motion modeling 2. Two Stream Network:

one for spatial stream, one for temporal stream (optical flow), fuse at the end

Follow-up of Two Stream Network: TDD, Beyond-Short-Snippet, Two-stream Fusion, TSN, TLE

3D CNNs: P3D, ECO, Non-local, SlowFast, X3D

Motion modeling: Rank Pooling, Compressed Videos, Hidden TSN, Trajectory Conv, TSM, TEA



Component of the flow perpendicular to the gradient is not observable (Aperture problem: due to partial observation, only flow component in the gradient direction is observable)

Lucas-Kanade flow: Assume flow is constant in a small window

Solve $Ax = b$ where $A = (I_x \ I_y), b = -I_t$, using least squares, $x = (A^T A)^{-1} A^T b$

$(A^T A)^{-1}$ is good when eigenvalues are not too small / well-conditioned, i.e. corners

Iterative Refinement: Under small motion assumption, $I(x + u_x, y + u_y, t + 1) - I(x, y, t) = 0$ has higher order terms, a polynomial root finding problem, solve using Newton's method.

Iterative Lucas-Kanade Algorithm: estimate velocity at each pixel, update image, repeat

Pyramid for large motion: reduce resolution, compute & upsample flow (coarse-to-fine estimate)

Horn-Schunck method: Flow is a minimizer of global energy function

$$E = \int \left[(I_x u_x + I_y u_y + I_t)^2 + \alpha^2 (\|\nabla u_x\|^2 + \|\nabla u_y\|^2) \right] dx dy \\ I_x (\nabla I \cdot u + I_t) - \alpha^2 \Delta u_x = 0 \quad I_y (\nabla I \cdot u + I_t) - \alpha^2 \Delta u_y = 0$$

Estimate Laplacian $\Delta u_x \approx \bar{u}_x - u_x$ Iterative solution: update u using the computed Δu

Michael Black method: replace quadratic regularization by $\sum_{n \in N(S)} (u_{x,s} - u_{x,n})^2 + (u_{y,s} - u_{y,n})^2$

FlowNet 1.0: correlation layer $c(x_1, x_2) = \sum_o f_1(x_1 + o, f_2(x_2 + o))$, upsampling layer

FlowNet 2.0: pretrain on chairs then finetune on Things3D; stacked architecture; sub-network specializing on small motions; fusion architecture

Motion segmentation: Goal: break into "layers", each has coherent (affine) motion

Affine motion: $u_x = ax + by + c, u_y = dx + ey + f$, solve each window by minimizing

$\sum (\nabla I \cdot u + I_t)^2$

cluster using K-means in affine params space

Lecture 13 Activity Recognition

Datasets: 1. Video Classification: UCF101 (YouTube video, 101 action classes), HMDB (Human Motion Database, 51 action classes, 6849 videos), Sports-1M (487 classes), YouTube-8M

2. Atomic Actions: Charades (RGB + Optical Flow features, action classification & sentence prediction), Atomic Visual Actions (AVA, 3s segments, fine-grained pose and object interactions, but no annotations about objects), Moments in Time (MIT, 3s, sound, balanced but single label)

3. Video description: M-VAD and MPII-MD, LSMDC (movie description, retrieval, and QA)

Ability: sequence modeling, temporal reasoning, focus on action recognition

Pre-DL: 1. Local feature: HOG + HOF (Histogram of Optical Flow) 2. Trajectory: Motion Boundary Histograms, dense trajectories 3. Ways to aggregate features: Bags of Visual Words, Fisher Vector

Representing Motion: Optical Flow / Trajectory Stacking

DL: 1. Single Stream Network: direct conv, lack motion modeling 2. Two Stream Network:

one for spatial stream, one for temporal stream (optical flow), fuse at the end

Follow-up of Two Stream Network: TDD, Beyond-Short-Snippet, Two-stream Fusion, TSN, TLE

3D CNNs: P3D, ECO, Non-local, SlowFast, X3D

Motion modeling: Rank Pooling, Compressed Videos, Hidden TSN, Trajectory Conv, TSM, TEA