Computer Vision

Instructors: Marc Pollefeys, Siyu Tang, Vittorio Ferrari

Lecture 1: Introduction & Pinhole Model

Computer Vision: automatic understanding of images and video.

Challenges: illumination, scale, deformation, viewpoint, occlusion, motion.

Homogeneous Coordinates

2D: $l^T x = (a,b,c)^T(x,y,1) = ax + by + c$.

Intersection of 2 lines: $x = l \times l'$

Joining 2 points: $l = x \times x'$

NB: $\times$ is the cross product (aka produit vectoriel)

$$x \times x' = [x]_\times x' = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix} x$$

$(x,y,0)$ is a point at infinity with direction (x,y).

$(0,0,c)$ is a line at infinity.

3D: $\pi^T X = (\pi_1,\pi_2,\pi_3,\pi_4)^T(X,Y,Z,1) = \sum \pi_i X_i$

Point on the plane: $X \in P \Leftrightarrow \pi^T X = 0$

Joining 3 points: $(X_1^T, X_2^T, X_3^T)^T \pi = 0$

Intersection of 3 planes: $(\pi_1^T, \pi_2^T, \pi_3^T)^T X = 0$

Lines:

$$W = (X_1^T, X_2^T)^T \text{ or } \lambda X_1 + \mu X_2$$
$$W^* = (\pi_1^T, \pi_2^T)^T \text{ or } \lambda \pi_1 + \mu \pi_2$$
$$W^* W^T = W W^{*T} = 0_{2\times 2}$$

Plane by line and point: $M\pi = \begin{bmatrix} W \\ X^T \end{bmatrix} \pi = 0$

Intersection of line and plane: $MX = \begin{bmatrix} W^* \\ \pi^T \end{bmatrix} X = 0$

2D Projective Transformations

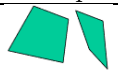
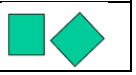
An invertible mapping $h: P^2 \to P^2$ such that 3 points x_1, x_2, x_3 lie on the same line if and only if $h(x_1), h(x_2), h(x_3)$ do.

Point: $x' = Hx$ Conic: $C' = H^{-T} C H^{-1}$

Line: $l' = H^{-T} l$ Dual conic: $C'^* = H C^* H^T$

Where $\det H \neq 0$

NB: same analogy for 3D points, planes, and quadrics.

Transformation	2D DoF	3D DoF	Example
Projective	8	15	
Affine	6	12	
Similarity	4	7	
Euclidian	3	6	

Fixed Points and Lines

$He = \lambda e$ (e is the same point as λe): eigenvectors.

For lines, take the eigenvectors of H^{-T}.

Affine Rectification: map back ideal points to infinity (after a projection).

Lecture 2: Camera Models and Calibration

Pinhole Camera Model: $(X,Y,Z) \mapsto (fX/Z, fY/Z)$

To avoid division by Z, we use homogeneous coordinates:

$$(X,Y,Z,1)^T \mapsto (fX, fY, Z)^T = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$\Rightarrow$ camera specific matrix + projection operation

$$x = PX = \text{diag}(f,f,1)[I|0]X$$

Principle Point Offset:

$$x = K[I|0]X = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} [I|0]X$$

Capture the rotation and translation from origin:

$$x = PX = K[R|t]X = KR[I|-C]X$$

Intrinsic camera parameters: $K = \begin{bmatrix} \alpha_x & s & p_x \\ & \alpha_y & p_y \\ & & 1 \end{bmatrix}$

Extrinsic camera parameters: R and t

Point: $x = PX$

Line: $l = PL = P(X_1 + \mu X_2) = x_1 + \mu x_2$

Line Back-projection: $\Pi = P^T l$

Compute the camera projection matrix given $X_i \leftrightarrow x_i$ using **D**irect **L**inear **T**ransform: we need 6 correspondences ($n \geq 6$).

Data Normalization:

1. Move center mass to origin.
2. Scale to yield order 1 values.

$$\tilde{x} = Tx = \begin{bmatrix} \sigma_{2D} & 0 & \bar{x} \\ 0 & \sigma_{2D} & \bar{y} \\ 0 & 0 & 1 \end{bmatrix}^{-1} x$$

$$\tilde{X} = UX = \begin{bmatrix} \sigma_{3D} & 0 & 0 & \bar{X} \\ 0 & \sigma_{3D} & 0 & \bar{Y} \\ 0 & 0 & \sigma_{3D} & \bar{Z} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} X$$

$x_i = PX_i \Leftrightarrow [x_i]_\times PX_i = 0$

$$\Leftrightarrow \begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \\ -y_i X_i^T & x_i X_i^T & 0^T \end{bmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = 0$$

$$\Leftrightarrow \begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \end{bmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = 0$$

$\Leftrightarrow A_i p = 0 \Leftrightarrow (A_1, \dots, A_n)^T p = 0 \Leftrightarrow Ap = 0$

Minimize geometric error:

$$\min_{P} \sum d(x_i, PX_i)^2$$

Gold Standard Algorithm

1. Linear solution: normalization + DLT
2. Minimization of geometric error
3. Denormalization: $P = T^{-1}\tilde{P}U$

Lecture 3: Local Features

Matching requirements

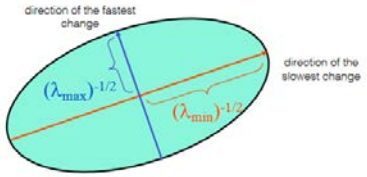
1. Detect the same points in both images.
2. Recognize the correspondence for each point.

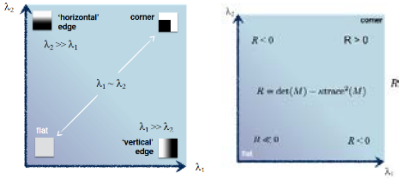
Key point Localization: Harris Detector

Shift a patch window W. Calculate the summed squared difference (SSD) before and after shifting.

$$\begin{aligned} E(u,v) &= \sum_{(x,y)\in W} \big(I(x+u,y+v) - I(x,y)\big)^2 \\ &\approx \sum_{(x,y)\in W} \left(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v\right)^2 \\ &= \sum_{(x,y)\in W} \left([I_x \quad I_y]\begin{bmatrix}u\\v\end{bmatrix}\right)^2 \\ &= \sum_{(x,y)\in W} [u \quad v]\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix} \end{aligned}$$

We calculate the eigenvalues λ_i and eigenvectors e_i of $M = \sum_{(x,y)} w(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2\end{bmatrix}$ by solving $\det(M - \lambda I) = 0$.





$$R = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det M - \kappa \mathrm{trace}^2(M)$$

Compare R to a lower bound threshold to detect the presence of corners.

The window can be uniform I or Gaussian $G(\sigma)$.

Rotation invariant: if the corner is rotated, the shape (i.e., the eigenvalues) remains the same.

Not scale invariant.

Scale Invariant Region Selection

Going from points to regions (descriptors) given two images of the same scene but with a large-scale difference.

Blob detector: used to detect uniform regions using Laplacian-of-Gaussian LoG which is expensive to compute. So, it's approximated by a Difference-of-Gaussians DoG.

Local Descriptors

How to describe detected points for matching?

Patch: pixel values, color histogram, spatial histograms.

Scale Invariant Feature Transform (SIFT)

1. Device the patch into 4x4 sub-patches (cells).
2. Compute histogram of gradient orientations (8 angle bins) for each cell.
3. Final descriptor of dimension 4x4x8.

Before comparing patches of across images, we normalize the orientation to a fixed orientation depending on the dominant angle of each patch.

Lecture 4: Optical Flow, Particle Filtering

Given two consecutive image frames, estimate the motion of each pixel.

Assumptions: color constancy and small motion.

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x,y,t)$$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$I_x u + I_y v + I_t = 0$$

Solve for u and v.

Lucas–Kanade method: constant flow for a 5x5 patch.

$$I_x(p_i)u + I_y(p_i)v = -I_t(p_i), \forall i \in \{1,\dots,25\}$$

$$Ax = b \Rightarrow A^TAx = A^Tb$$

$$\Leftrightarrow \begin{bmatrix} \sum_{p\in P} I_xI_x & \sum_{p\in P} I_xI_y \\ \sum_{p\in P} I_xI_y & \sum_{p\in P} I_yI_y\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix} = -\begin{bmatrix}\sum_{p\in P} I_xI_t \\ \sum_{p\in P} I_yI_t\end{bmatrix}$$

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

This method works best when λ_1 and λ_2 of A^TA are big, i.e., presence of corners.

Problem: when there are no corners in the image.

Horn–Schunck method: rigid objects/smoothness

Color constancy updated: for every pixel:

$$\min_{u,v} \big[I_xu_{ij} + I_yv_{ij} + I_t\big]^2$$

Smooth flow field:

$$\min_{u} \big[u_{i,j} - u_{i+1,j}\big]^2$$

Putting all together

$\min_{u,v} \sum_{i,j}[E_s(i,j) + \lambda E_d(i,j)]$ where:

Smoothness: $E_s(i,j) = \frac{1}{4}\Big[\big(u_{i,j} - u_{i+1,j}\big)^2 + \big(u_{i,j} - u_{i,j+1}\big)^2 + \big(v_{i,j} - v_{i+1,j}\big)^2 + \big(v_{i,j} - v_{i,j+1}\big)^2\Big]$

Brightness: $E_d(i,j) = \big[I_xu_{ij} + I_yv_{ij} + I_t\big]^2$

Weight: λ

Use gradient descent to solve the problem.

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x$$

$$\hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

new value — old average

Particle Filters

Predict

1. Particles are sampled according to their weights.
2. Apply motion model (velocity model) and add noise.

Update/Correct

1. State estimates are turned into observation using data from the image (histogram from a bound box).
2. Update particles: those having a state with similar histogram to the original will have bigger weights.

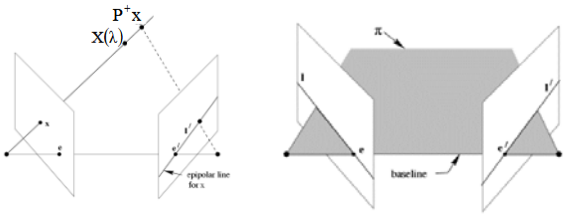
Lecture 5: Multiple View Geometry & Structure from Motion

Aim: How can multiple images of the same 3D scene be related?

Correspondence geometry: Given an image point x in the first image, how does this constrain the position of the corresponding point x' in the second image?

Epipolars e and e' = the intersection of baseline with image plane.

Epipolar plane π = plane containing baseline



Epipolar lines l and l' = intersection of epipolar plane with image.

The fundamental matrix F to map $x \mapsto l'$

Geometric derivation

$$x' = H_\pi x$$
$$l' = e' \times x' = [e']_\times H_\pi x = Fx$$
$$l = F^T x'$$

Algebraic derivation

We generate a 3D point $X(\lambda) = P^+x + \lambda C$ where P^+ is the pseudoinverse of P ($PP^+ = I$). The we project both C and $X(\lambda)$ on P'.

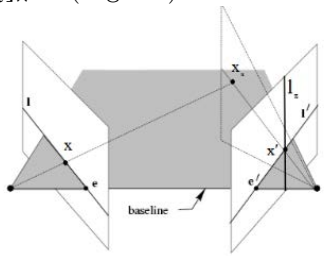
$$l' = P'C \times P'P^+x = [e']_\times P'P^+x = Fx$$

For every $x \leftrightarrow x'$, we have $x'^T Fx = x^T F^T x' = 0$ where F is the unique 3×3 rank 2 matrix satisfying that.

Relation between F and homographies

$[e']_\times H_\pi = F$, $l' = H_\pi^{-T} l$, $e' = H_\pi e$ (H_π given)

$x' = H_\pi x = [l_\pi]_\times Fx$ (F given)



Camera geometry (motion): Given a set of corresponding image points $\{x_i \leftrightarrow x_i'\}$, what are the cameras P and P' for the two views?

$$x = PX = (PH)(H^{-1}X) = \hat{P}\hat{X}$$
$$x' = P'X = (P'H)(H^{-1}X) = \hat{P}'\hat{X}$$

$(P, P') \mapsto F$ is unique

$F \mapsto (P, P')$ is NOT unique

We can choose the canonical way

$P = [I|0]$ and $P' = [M|m] = [[e']_\times F + e'v^T|\lambda e']$

Given a point x, we can limit x' searching to the corresponding epipolar line l'

$$m_2^T F m_1 = m_2^T l_2 = l_1^T m_1 = 0$$

(1:34:15)

Lecture 6: Model Fitting & Structure from Motion

Hough Transform: mapping between 2D space (tokens) and (θ, ρ) parameters space (votes). $p \mapsto l$ and $l \mapsto p$.

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

Line Fitting

Incremental Line Fitting: include a new point each time and fit a line, until the new added point is far from the lin.

K-means Line Fitting: randomly pick lines, associate points to line, and refit line.

Problem? Squared Error can be a source of bias in the presence of noise points. Because a quadratic function gives too much weight to outliners. Instead, we use $\rho(r, \sigma) = \frac{r^2}{\sigma^2 + r^2}$ where $\psi(r, \sigma) = \frac{\partial \rho}{\partial r} = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$.

RANSAC: randomly choose a subset, fit to it, anything close is signal, refit, redo.

Choose N (number of trials) so that at least one random sample is outliers-free (with a $p = 0.99$ for example).

$$(1 - (1 - e)^s)^N = 1 - p$$
$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

p: success probability and e: outliners ratio.

Cross-validation: use part of the data to fit a model and the rest to evaluate it.

Structure from Motion

Recovering 3D structure of the scene from camera motion.

Factorization

- Affine Projection: $\begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix} = \begin{bmatrix} P_i^x \\ P_i^y \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{bmatrix}$

$$\begin{bmatrix} x_{ij} - P_i^{x4} \\ y_{ij} - P_i^{y4} \end{bmatrix} = \begin{bmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{bmatrix} = \begin{bmatrix} \bar{P}_i^x \\ \bar{P}_i^y \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix}$$

$\tilde{x}_{ij} = x_{ij} - \sum_i x_{ij}$ and $\tilde{y}_{ij} = y_{ij} - \sum_i y_{ij}$

- Orthographic Projection: $\bar{m}_{ij} = \bar{P}_i \bar{M}_j$

Where $\bar{m}_{ij} = \begin{bmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{bmatrix}$, $\bar{P}_i = \begin{bmatrix} \bar{P}_i^x \\ \bar{P}_i^y \end{bmatrix}$ and $\bar{M}_j = \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix}$

Indexes: i for cameras and j for points.

All equations can be collected:

$$\bar{m} = (\bar{m}_{ij})_{1 \le i \le m, 1 \le j \le n} \in \mathbb{R}^{2m \times n}$$
$$\bar{P} = (\bar{P}_1, \dots, \bar{P}_m)^T \in \mathbb{R}^{2m \times 3}$$
$$\bar{M} = (\bar{M}_1, \dots, \bar{M}_n) \in \mathbb{R}^{3 \times n}$$

SVD-factorize $\bar{m} = U \Sigma V^T \Rightarrow \tilde{P} = U, \tilde{M} = \Sigma V^T$

(COLMAP: open source pipeline)

Refining structure and motion by minimizing Euclidean distances between 2D projected and observed points: $\min_{\hat{P}_k, \hat{M}_i} \sum_{k=1}^{m} \sum_{i=1}^{n} d(m_{ki}, \hat{P}_k \hat{M}_i)^2$

Sum over all the points i and all the images k.

Non-linear least-squares

Find P such that $X = f(P)$ by solving $\arg\min_P \|X - f(P)\|$

Newton iteration

$f(P_0 + \Delta) \approx f(P_0) + J\Delta$ where $J = \frac{\partial X}{\partial P}$

$$\|X - f(P_1)\| \approx \|X - f(P_0 + \Delta)\| = \|X - f(P_0) - J\Delta\| = \|e_0 - J\Delta\|$$

We project on the Jacobian to get a unique solution $\Rightarrow J^T J\Delta = J^T e_0 \Rightarrow \Delta = (J^T J)^{-1} J^T e_0$

Generally, $P_{i+1} = P_i + \Delta$ where $\Delta = (J^T J)^{-1} J^T e_0$.

Levenberg-Marquardt

We use $J^T J + \lambda \text{diag}(J^T J)$ to make it invertible.

Lecture 7: (Multiview) Stereo

For standard stereo geometry, we have a pure translation along x-axis:

$$F = [t]_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad Fx = \begin{bmatrix} 0 \\ 1 \\ -y \end{bmatrix} \quad F^T x' = \begin{bmatrix} 0 \\ 1 \\ -y' \end{bmatrix}$$

Stereo matching: for each pixel, look for the most similar pixel on the epipolar line.

Higher level feature matching (edges, etc.) can yield better results.

Constraints and Optimization

- Occlusions: some pixels may only appear in one image: mutual match testing.
- Ordering constraint: the matching pixels appear in the same order in both images.

Use Dynamic Programming (optimal path) to measure similarities.

(2:35:00)

Lecture 8: Specific Object Recognition

Recognition

There are up to 3,000 object categories.

Category vs. Instance (e.g. celebrity) recognition.

History: geometric era, appearance-based models (color histogram), sliding window, local features, parts-and-shape models, bags of features.

Present: local & global methods + context + DL

Specific Object Recognition

Basic idea: see how many key points are close to the other images' key points (very slow).

Scaling to Large Databases

Index local features: each patch has a descriptor i.e., a point in high-dimension space. Then see close points in feature space.

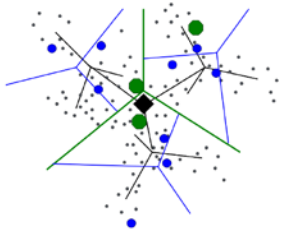
To reduce calculations, we assign one center for each cluster, and compare new features only with the cluster center → construct a vocabulary.

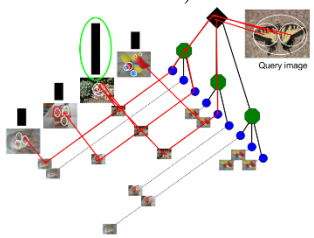
Fast lookup: inverted index: find all images containing a specific feature. Build a dictionary $\{w_i : (i_{i_1}, \dots, i_{i_n})\}$ where w are words and i images.

Given a query image containing a feature w, we can get all images i containing w.

How to choose vocabulary size?

Extract features from all training images and run multi-level k-means (e.g., 10 cluster and 6 levels).





How to measure the model performance?

Precision $= \frac{tp}{tp+fp} = \frac{relevent}{returned}$

Recall $= \frac{tp}{tp+fn} = \frac{relevent}{total\ relevent}$

Affine Transformation

Given a set of matching points $\{x_i, x_i'\}$ find the best parameters p such that $x_i' = f(x_i, p)$ where f is the transformation function (translation, rotation, aspect, affine, perspective, cylindrical).

Scaling: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

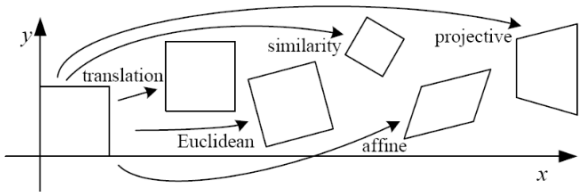
Shear: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Rotation: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

In general: $\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}$

Translation: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Affine Transformation: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$



How to determine the parameters of an AT?

Calculate: $\hat{p} = \arg\min \sum_i \|f(x_i, p) - x_i'\|^2$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}$$

And then, we stack all the points to get $b = Ax$. The problem is now $\hat{x} = \arg\min(Ax - b)$.

$$\hat{x} = (A^T A)^{-1} A^T b$$

Projective Transformation (Homography)

Skipped section: DLT and RANSAC.

Determining the homography matrix

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = 0$$

Solve with SVD.

Things to remember

Find key points & compute descriptors > match descriptors > vote for or fit transformation parameters > return object if # inliers > T.

Lecture 9: Recognition and Modeling Humans

People Recognition from Visual Input

Graph: nodes + edges

Decomposition of graph: partition of nodes

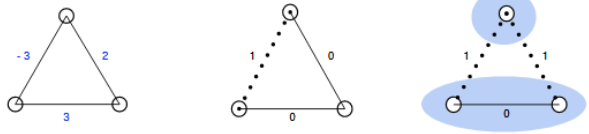
Multicut: set of edges being cut for the partition

Minimum Cost Multicut Problem

Objective: $\min_{x \in \{0,1\}^E} \sum_{e \in E} c_e x_e$

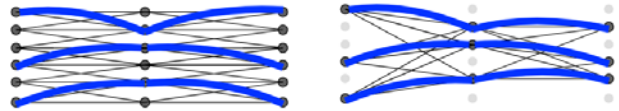
Constraint: $\forall C \in cc(G)\ \forall e \in C: x_e \leq \sum_{e' \in C \setminus \{e\}} x_{e'}$

If there is a cut in a cycle, it can't be a single cut.

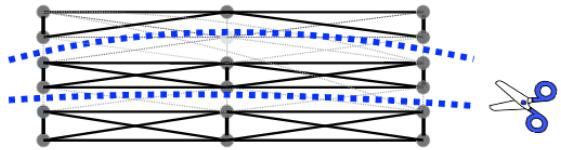


Graph-based modelling by:

- Finding disjoint paths in the graph modelling the detections (boundary boxes) in each frame over time. Merge tracks.



- Graph decomposition: keep all the detections and cut the graph.



Graph decomposition Formulation:

Objective: $\min_{x \in \{0,1\}^V, y \in \{0,1\}^E} \sum_{v \in V} c_v x_v + \sum_{e \in E} d_e y_e$.

Consistency: $\forall e = vw \in E: y_{vw} \leq x_v, y_{vw} \leq x_w$.

Transitivity:

$\forall C \in cc(G)\ \forall e \in C: (1 - y_e) \leq \sum_{e' \in C \setminus \{e\}} (1 - y_{e'})$.

DeepCut: joint edge and node labeling

y: edge variable: cut or not cut

x: node variable: the label of the node

The main idea is to take the node labels into consideration while performing edge cuts.

Generative Human Body Models

SMPL model example.

Lecture 10: Tracking

Follow the movements of something (point, region, template, part, object, segmentation, pose) or somebody.

Applications: autonomous driving, image editing, sports, AR/VR, customer tracking, counting in subway/airport.

Track a Point: points having the same color

$$E(h) = [I_0(x+h) - I_1(x)]^2 \approx [I_0(x) + hI_0'(x) - I_1(x)]^2$$
$$\frac{\partial E}{\partial h} = 2I_0'(x)[I_0(x) + hI_0'(x) - I_1(x)]$$
$$\frac{\partial E}{\partial h} = 0 \Rightarrow h = \frac{I_1(x) - I_0(x)}{I_0'(x)}$$

Problem 1: zero gradient: nonzero gradients in all directions are needed.
Problem 2: local minima: the frame rate should be faster than half-wavelength of the signal.

Track a Template
Problem: template might take a transformation
Lucas-Kanade Template Tracker
From $E(u,v) = \sum_{x,y}[I(x+u, y+v) - T(x,y)]^2$ which works only with translation motion
To $E(p) = \sum_{x,y}\left[I\big(W(x,p)\big) - T(x,y)\right]^2$
Where T is the template and W is a warp function that generalizes to other motions (affine, projective, etc.).
Using Taylor expansion: $\sum_x\left[I\big(W(x,p+\Delta p)\big) - T(x)\right]^2 = \sum_x\left[I\big(W(x,p)\big) + \nabla I\frac{\partial W}{\partial p}\Delta p - T(x)\right]^2$

Mean Shift to track a general region.
Non-parametric feature-space analysis technique to locate the maxima of a density function.
Choose a region of interest, calculate the center of mass, and update the region of interest.
Which feature space to use? Color pixels, grayscale, gradients.

Tracking by detection (use features)
Extract the features present in the reference image and try to find them in the test image. We use feature descriptor to describe nearby region for each feature.

Track with a model
Detect the object and perform space-time analysis.

Lecture 11: Image Segmentation

Identify group of pixels that belongs together.

Segmentation as Clustering

K-Means: Use K-Means to form clusters of pixels depending on their values (grayscale or (R,G,B)). We can consider spatial information, e.g., (R,G,B,x,y).

Gaussians: assume that points are generated by sampling a continuous function (generative model). K Gaussian blobs:

$$P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d|V_b|}}\exp -\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)$$

The likelihood of observing x

$$P(x|\theta) = \sum_{b=1}^{K}\alpha_b P(x|\theta_b)$$

where α_b is the probability to select blob b.

Model-free Clustering: Mean-Shift

- Choose features.
- Initialize windows at pixel locations.
- Start mean-shift.
- Merge windows ending at the same peak.

Hough Transforms

Use the structure of shapes to extract them in a parameter space of limited dimension.
A counter is incremented at each cell where a line pass. Then we can use mean-shift to detect peaks.

Interactive Segmentation with GraphCuts

Markov Random Fields: $\{y_i\}$ are the pixel features (color or other features) and $\{x_i\}$ are the hidden states (e.g., back- or foreground).

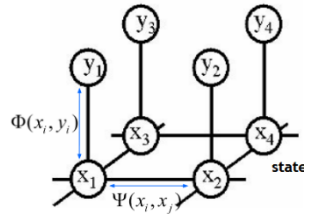
Field Joint Probability

$$P(x,y) = \prod_i \Phi(x_i, y_i)\prod_{i,j}\Psi(x_i, y_j)$$

Energy Function

$$E(x,y) = \log P(x,y)$$
$$= \sum_i \log\Phi(x_i,y_i) + \sum_{i,j}\log\Psi(x_i,y_j)$$
$$= \sum_i \phi(x_i,y_i) + \sum_{i,j}\psi(x_i,y_j)$$

ϕ: unary potentials: how likely is a pixel to be in a certain state.



ψ: pairwise potentials: how different is a pixel label from that of a neighbor one.

Learning-Based Approaches

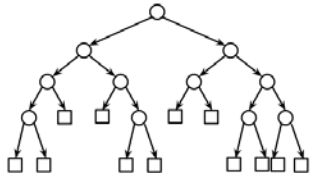
K-Nearest Neighbors

Pros
- Simple to implement and understand.
- The distance definition is flexible.

Cons
- Depends on the definition of K.
- Needs to keep the entire data in memory.
- For higher d, many samples are needed.

Random Forest

Binary decision trees



At each internal node, there is a binary question on the features extracted from the sample (e.g., image patch) ending in a leaf node.

A random forest consists of an ensemble of different decision trees. Once trained, a sample is put in on the top of every tree, and typically the class is the average of prediction.

Training
Binary stump: thresholding on one feature. During training, the feature k and the threshold τ are determined for each node.

The gain in class certainty

$$G(k,\tau) = H(\text{parent}) - \left|\frac{D_{rc}}{D_p}\right| H(\text{rc}) - \left|\frac{D_{lc}}{D_p}\right| H(\text{lc})$$

Where
- $D_n = \{(v_n, c_n) \in \text{Node } n\}$: training samples.
- $H(n) = \sum_i P(w_i) \log P(w_i)$: node entropy
- $P(w_i) = \sum_n^{|D_n|} \frac{\delta(c_n = w_i)}{|D_n|}$
- $k^*, \tau^* = \arg\max G(k,\tau)$

Stopping criterion: a child node is declared leaf node if one of the criteria is satisfied:
1. All its samples belong to one class.
2. The number of samples is below a threshold.

Pros
- Easy to implement.
- Efficient during testing.
- Can handle high dimensional space.

Cons
- Lots of parametric choices.
- Needs large amount of data.

Lecture 12: Object Class Recognition

Classification

Visual Words: map high-dimensional descriptors to words by quantizing the feature space. Then, determine which word to assign to a new image region by finding the closest cluster center.

Bag-of-Words: histogram of how many times each feature appeared in the image. To do categorization, we compare bag-of-words.
Learn a classifier: KNN, SVM.

Limitation of BoW: removes spatial layout: no spatial relationships between features.
→ This can be solved by Spatial Pyramid Representation: one histogram for each region.

Pros:
- Flexible to geometry transformations.
- Compact summary of the image content.
- Good results in practice.

Cons:
- Ignores geometry.
- Background and foreground mixed.
- How to generate the vocabulary?

CNNs: the features are learned.
Filtering: convolution (one feature map per filter)
Non-linearity: to model complex mappings.
Pooling: gives robustness to small shift.
Normalization: equalizes the features maps.

Detection

Sliding-window approaches
1. Obtain training data.
2. Define features.
3. Define classifier.

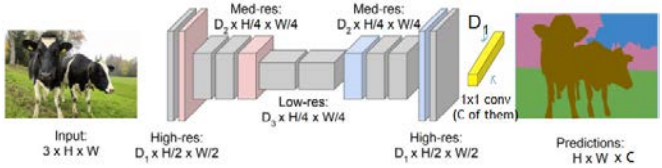
R-CNN
1. Extract region proposals
2. Warp each region into a square
3. Compute CNN features
4. Classify regions (e.g., SVM)

Fast R-CNN
Instead of feeding the region proposals to CNN, the input image is fed to the CNN to generate a convolutional feature map. Then, identify region of proposals, warp and classify them.
Why it's faster? Convolution is done once per image.

Implicit Shape Model
ISM is a codebook for each class along with a spatial probability distribution describing where each codebook entry could be found on the object.

Segmentation

Label each pixel of the image with a category label. This could be done using CNN: downsamling and upsampling.



--- END ---