Computer Vision

1 - Introduction and pinhole model

- What is computer vision, related disciplines, challenges
- Projective geometry
  - $x \times x' = [x]_\times x'$, where $[x]_\times = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix}$
  - Homogeneous coordinates (**only 2 DOF** for 2D)
  - Points and lines, ideal points and lines at infinity
    - Set of all equivalence relations in $\mathbb{R}^3 - (0,0,0)^T$ forms $\mathbb{P}^2$
    - Point-line intersection: $l^T x = 0$
    - Intersection of lines: $x = l \times l'$
    - Line joining two points: $l = x \times x'$
    - Intersection of parallel lines $l = (a,b,c)^T$ and $l' = (a,b,c')^T$ is $l \times l' = (b,-a,0)^T$
    - Ideal points: $(x_1, x_2, 0)^T$
    - Line at infinity: $l_\infty = (0,0,1)^T$
    - Duality of points and lines in $\mathbb{P}^2$: equations symmetric, homogeneous representations identical
    - Circular points: $I = (1,i,0)^T$ and $J = (1,-i,0)^T$
  - 3D points and lines, representing them by spans
    - Point-plane intersection: $\pi^T X = 0$
    - Ideal points: $(X_1, X_2, X_3, 0)^T$
    - Plane at infinity: $\pi_\infty = (0,0,0,1)^T$
    - Duality of points and planes
    - Lines have **4 DOF** in 3D
  - Conics: **5 DOF** (2D equation), tangent lines, dual conics
    - Conic equation: $x^T C x = 0$
    - Tangent line at a point: $l = Cx$, where $l^T C^* l = 0$ and $C^* = C^{-1}$ for full rank C
  - Degenerate conics ($(C^*)^* \neq C$)
    - $C = lm^T + ml^T$ (rank 2, two lines)
    - $C = ll^T$ (rank 1, repeated line)
  - Quadrics (**9 DOF**) and dual quadrics
    - $X^T Q X = 0$, where Q is 4x4 symmetric matrix
    - Tangent plane: $\pi = QX$, $\pi^T Q^* \pi = 0$, $Q^* = Q^{-1}$ (non-degenerate)
  - Projective transformations: $x' = Hx$ (8 DOF, H non-singular)
    - Projectivity = collineation = projective transformation = homography
    - Point transformation: $x' = Hx$
    - Line transformation: $l' = H^{-T} l$
    - Conic transformation: $C' = H^{-T} C H^{-1}$
    - Dual conic transformation: $C'^* = H C^* H^T$
  - Hierarchy of 2D transformations
    - Projective: **8 DOF**, invariants: concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio

        - Affine: **6 DOF**, invariants: parallelism, ratio of areas, ratio of lengths on parallel lines (e.g. midpoints), linear combinations of vectors (centroids), the line at infinity l_∞
        - Similarity: **4 DOF**, invariants: ratios of lengths, angles, the circular points (I, J), dual conic
        - Euclidean: **3 DOF**, invariants: lengths, areas
    - Circular points
        - $I = (1, i, 0)^T$, $J = (1, -i, 0)^T$
    - Hierarchy of 3D transformations
        - Projective: **15 DOF**, invariants: intersection and tangency
        - Affine: **12 DOF**, invariants: parallelism of planes, volume ratios, centroids, the plane at infinity π_∞
        - Similarity: **7 DOF**, invariants: angles, ratios of lengths, the absolute conic Ω_∞, absolute dual quadric Ω_∞^*
        - Euclidean: **6 DOF**, invariants: volume
- A (scalar) **invariant** of a geometric configuration is a function of the configuration whose value is unchanged by a particular transformation.
- **Isometries** are transformations of the plane $\mathbb{R}^2$ that preserve Euclidean distance.
    - If $\epsilon = 1$ then the isometry is orientation-preserving and is a Euclidean transformation (a composition of a translation and rotation).If $\epsilon = -1$ then the isometry reverses orientation, an example is the composition of a reflection.
    - A planar Euclidean transformation has **three degrees of freedom**, one for the rotation and two for the translation. Thus three parameters must be specified in order to define the transformation. The transformation can be computed from **two point correspondences**.
    - An isometry is orientation-preserving if the upper left hand 2 × 2 matrix has determinant 1.
- A **similarity** transformation (or more simply a similarity) is an isometry composed with an isotropic scaling.
    - A similarity transformation is also known as an equi-form transformation, because it preserves “shape” (form).
    - A planar similarity transformation has **four degrees of freedom**, the scaling accounting for one more degree of freedom than a Euclidean transformation. A similarity can be computed from **two point correspondences**.
    - The description metric structure implies that the structure is defined up to a similarity.
- An **affine transformation** (or more simply an affinity) is a non-singular linear transformation followed by a translation.
    - A planar affine transformation has **six degrees of freedom** corresponding to the six matrix elements. The transformation can be computed from **three point correspondences**.
    - The only “new” geometry, compared to a similarity, is the non-isotropic scaling. This accounts for the two extra degrees of freedom possessed by an affinity over a similarity. They are the angle ϕ specifying the scaling direction, and the ratio of the scaling parameters $\lambda_1 : \lambda_2$.
- A **projective transformation** is a general non-singular linear transformation of homogeneous coordinates. This generalizes an affine transformation, which is the composition of a general non-singular linear transformation of inhomogeneous coordinates and a translation.
    - The matrix has nine elements with only their ratio significant, so the transformation is **specified by eight parameters**. Note, it is not always possible to scale the matrix such

that v is unity since v might be zero. A projective transformation between two planes can be computed from **four point correspondences, with no three collinear on either plane**.
    - Unlike the case of affinities, it is not possible to distinguish between orientation preserving and orientation reversing projectivities in $\mathbb{P}^2$
- $H_E = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix}$, $H_S = \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix}$, $H_A = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix}$, $H_P = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v^T} & v \end{bmatrix}$ where R is rotation, A is non-singular

2 - Camera models and calibration

- Pinhole camera (perspective projection)
    - Distant objects appear smaller
    - Parallel lines meet (vanishing point)
- Geometric properties of projection
    - Points go to points
    - Lines go to lines
    - Planes go to whole image or half-plane
    - Polygons go to polygons
    - Degenerate cases:
        - Line through focal point yields a point
        - Plane through focal point yields a line
- For general projective camera: $x = PX$, where P is $3x4$ homogeneous camera projection matrix.
- The centre of projection is called the camera centre. It is also known as the optical centre. The line from the camera centre perpendicular to the image plane is called the principal axis or principal ray of the camera, and the point where the principal axis meets the image plane is called the principal point. The plane through the camera centre parallel to the image plane is called the principal plane of the camera.
- Pinhole model of central projection:
    - $(X, Y, Z)^T \longmapsto (fX/Z, fY/Z)^T$, where $P = diag(f, f, 1)[I|0]$
- 3D Projection
    - Perspective projection: $(X, Y, Z)^T \longmapsto (fX/Z, fY/Z)^T$ — Parallel lines appear to meet at a vanishing point; farther objects seem smaller
    - Weak perspective projection: $(X, Y, Z)^T \longmapsto (fX/Z_0, fY/Z_0)^T$ — Parallel projection (parallel lines remain parallel) + Scaling to simulate change in size due to object distance
    - Orthographic projection: $(X, Y, Z)^T \longmapsto (X, Y)^T$— Pure parallel projection. Highly simplified case where we even ignore the scaling due to distance
- Pinhole model with principal point offset:
    - $(X, Y, Z)^T \longmapsto (fX/Z + p_x, fY/Z + p_y)^T$, where $(p_x, p_y)^T$ are the coordinates of the principal point.
    - $P = K[I|0]$, $K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$, K is the calibration matrix (intrinsic parameters)
- Generalized pinhole model:
    - Camera rotation and translation

    - $P = KR[I|-C]$ (KR non singular) or $P = K[R|t]$ where $t = -RC$, $[R|t]$: extrinsic camera matrix
    - **9 degrees of freedom**: 3 for K (f, p_x, p_y), 3 for R, 3 for C
- CCD cameras:
    - $K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$ — where $\alpha_x = fm_x$, $\alpha_y = fm_y$, $x_0 = m_x p_x$, $y_0 = m_y p_y$
    - Additional possibility of having non-square pixels
    - α is focal length, m is number of pixels per unit direction, α_y/α_x is aspect ratio
    - CCD cameras have **10 degrees of freedom**
- Finite projective cameras:
    - $K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$
    - **11 degrees of freedom**, this is the same number of degrees of freedom as a 3×4 matrix defined up to an arbitrary scale
    - The skew parameter will be zero for most normal cameras. In realistic circumstances a non-zero skew might arise as a result of taking an image of an image, for example if a photograph is re-photographed, or a negative is enlarged
    - The set of camera matrices of finite projective cameras is identical with the set of homogeneous 3×4 matrices for which the left hand 3×3 submatrix is non-singular
- General projective cameras:
    - The final step in our hierarchy of projective cameras is to remove the non-singularity restriction on the left hand 3×3 submatrix
    - A general projective camera is one represented by an arbitrary homogeneous 3×4 matrix of rank
    - It has **11 degrees of freedom**
- Back-projection of points to rays:
    - Pseudo-inverse of P: $P^+ = P^T(PP^T)^{-1}$, for which $PP^+ = I$
    - Point P^+x lies on the ray because it projects to x, since $P(P^+x) = x$
    - The ray through C and P^+x: $X(\lambda) = P^+x + \lambda C$
- **Wikipedia**: In mathematics, a system of equations is considered overdetermined if there are more equations than unknowns.
- **Direct Linear Transform (DLT)**
    - $\lambda_i \mathbf{x_i} = P\mathbf{X_i}$, There are $3N$ equations and $11 + N$ unknowns, we need $3N \geq 11 + N \implies N \geq 6$ points to solve the problem.
    - $P = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix}$, where p_i are the rows of P. Equalities are $\mathbf{X_i}^T p_1 - \lambda_i x_i = 0$, $\mathbf{X_i}^T p_2 - \lambda_i y_i = 0$, $\mathbf{X_i}^T p_3 - \lambda_i = 0$
    -
- Decomposing the camera matrix
    - Finding the camera centre: The camera centre C is the point for which PC = 0. Numerically this right null-vector may be obtained from the SVD of P.
    - Finding the camera orientation and internal parameters: $P = [M|-MC] = K[R|-RC]$
        - We may easily find both K and R by decomposing M as M = KR using the RQ-decomposition

        - The ambiguity in the decomposition is removed by requiring that K have positive diagonal entries
    - Degenerate configurations:
        - The camera and points all lie on a twisted cubic.
        - The points all lie on the union of a plane and a single straight line containing the camera centre.
- Cameras at infinity:
    - We now turn to consider cameras with centre lying on the plane at infinity. This means that the left hand 3×3 block of the camera matrix P is singular. The camera centre may be found from $PC = 0$ just as with finite cameras.
    - It is called an affine camera because points at infinity are mapped to points at infinity.
- **Radial distortion**
    - Not modeled by the K-matrix. Cannot be removed by a projective mapping since lines are not mapped onto lines
    - Typically modeled as $x = x_d(1 + k_1 r^2 + k_2 r^4)$ and $y = y_d(1 + k_1 r^2 + k_2 r^4)$ where $r^2 = x_d^2 + y_d^2$, (x_d, y_d) are coordinates of distorted points w.r.t. image center, k_1 usually accounts for 90% of the distortion.
- A camera is called calibrated if the inner parameters K are known. If we normalize the coordinates in the image we get a normalized camera.
- One way to remove the projective ambiguity is to use calibrated cameras.
- Computation of camera matrix P (resectioning)
    - Minimal solution: Since the matrix P has 12 entries, and (ignoring scale) **11 degrees of freedom**, it is necessary to have 11 equations to solve for P. Since each point correspondence leads to two equations, at a **minimum** $5\frac{1}{2}$ **such correspondences** are required to solve for P
    - Over-determined solution: If the data is not exact, because of noise in the point coordinates, and $n \geq 6$ point correspondences are given, then there will not be an exact solution to the equations $Ap = 0$. As in the estimation of a homography a solution for P may be obtained by minimizing an algebraic or geometric error.
    - In the case of algebraic error the approach is to minimize $||Ap||$ subject to some normalization constraint. A possible constraint is $||p|| = 1 \rightarrow$ use SVD
    - Gold Standard Algorithm
        - Linear Solution (e.g. Normalization and Direct Linear Transform (DLT)) where normalization is $\tilde{x}_i = Tx_i$, $\widetilde{X}_i = UX_i$
        - Minimize geometric error (using an iterative algorithm e.g. Levenberg-Marquardt) $\sum_i d(\tilde{x}_i, \widetilde{P}\widetilde{X}_i)^2$
        - Denormalization: $P = T^{-1}\widetilde{P}U$
- Restricted camera estimation
    - Common assumptions are:
        - The skew s is zero.
        - The pixels are square: $\alpha x = \alpha y$.
        - The principal point (x_0, y_0) is known.
        - The complete camera calibration matrix K is known.
    - Can clamp the values to desired values after general DLT
    - Alternatively, can impose soft constraints and gradually increase weights
- Action of a projective camera
    - Forward projection of a line: $X(\mu) = P(A + \mu B) = a + \mu b$

    - Back-projection of a line: $\Pi = P^T l$ as $X^T P^T l = 0$
    - The lines for which $PL = 0$ pass through the camera centre
    - The effect of zooming by a factor k is to multiply the calibration matrix K on the right by $diag(k, k, 1)$.

3 - Feature extraction

- Feature:
    - Measured characteristic of (part of) a pattern / object
    - A feature should capture something discriminative about a well localizable patch of a surface
    - Localizable: shifting the patch a bit should make a big difference in terms of the underlying pattern
- Challenges
    - Viewpoint, illumination and background variation, occlusions
    - Scale change, occlusion, deformation, illumination change, blur...
- Considerations when selecting features
    - Not complete (not describing the pattern unambiguously)
    - Robust and easy extraction
    - Local instead of global
- **Characteristics of good features**:
    - Saliency: each feature is distinctive
    - Repeatability: the same feature can be found in several images despite geometric and photometric transformations
    - Compactness and efficiency: many fewer features than image pixels
    - Locality: a feature occupies a relatively small area of the image, robust to clutter and occlusion
- Corner detection
    - $E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \simeq \sum_{(x,y) \in W} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$, H is the gradient matrix
    - Harris corner detector: $R = det(H) - k[trace(H)]^2$, where $det(H) = \lambda_1 \lambda_2$ and $trace(H) = \lambda_1 + \lambda_2$
    - Alternatively, it is: $R = det(H)/trace(H)$
    - Shi-Tomasi corner detector: $R = min(\lambda_1, \lambda_2)$
    - Can apply non-maximum suppression afterwards.
- Feature descriptors
    - Requires invariance under geometric and photometric changes
    - Invariance under transformations implies invariance under the smallest encompassing group
    - As our local patches are by definition small, we expect that invariance under affine or even similarity transformations will do, also for general viewing conditions (if viewing changes are not too extreme)
    - Model for photometric change: linear part for intensity of illumination, non-linear part for specularities
- Normalized cross-correlation (NCC): It basically follows a matched filtering approach, correlating normalized intensities instead of the original intensity values. It is invariant to affine intensity changes.
- Patch: the shape of the patch should change with the viewpoint

- Parallelogram next to the edge corner (edge corners + affine moments)
    - Harris corner detection
    - Canny edge detection
    - Evaluate relative affine invariant parameter along two edges
    - Select parallelograms based on invariant extrema of function
    - Describe the pattern within the parallelogram with affine invariant moments
- Ellipses (intensity extrema + affine moments)
    - Search intensity extrema
    - Observe intensity profile along rays
    - Search maximum of invariant function f(t) along each ray
    - Connect local maxima
    - Fit ellipse (preserves affine covariance)
    - Double ellipse size (better descriptors)
    - Describe elliptical patch with moment invariants
- In practice different types of interest points are often combined
- **Wikipedia**: Moments
    - Image moment is a certain particular weighted average (moment) of the image pixels' intensities, or a function of such moments.
    - The central moments μ_{ij} of any order are, by construction, invariant with respect to translations.
    - Invariants η_{ij} with respect to both translation and scale can be constructed from central moments by dividing through a properly scaled zero-th central moment
    - As shown in the work of Hu, invariants with respect to translation, scale, and rotation can be constructed.
- Maximally Stable Extremal Regions (MSER)
    - Consecutive image thresholding by all thresholds: take regions at thresholds where the growth is slowest (happens when region is bounded by strong edges)
    - Region: a contiguous subset of image — connected component
    - Extremal Region R_i: all pixels are of equal or lower/higher intensity than i
    - Maximally Stable Extremal Region: a region R_i for which the relative change of area
    - Covariant with continuous deformations of images
    - Invariant to affine transformation of pixel intensities
    - Enumerated in O(n log log n), real-time computation
- Scale-Invariant Feature Transform (SIFT)
    - a carefully crafted interest point detector + descriptor
    - Based on intensity gradients (cf. our comment on photometric invariance) and invariants under similarities, not affine like so far.
    - Creating the scale space: the idea is to blur an image progressively, shrink it, blur the small image progressively and so on.
    - The creator of SIFT suggests that 4 octaves and 5 blur levels are ideal for the algorithm.
    - Approximate Laplacian of Gaussian (LoG) by Difference of Gaussian (DoG)
    - Compare pixels with 26 neighbors to find extrema, use Taylor expansion to find the sub-pixel maxima/minima
    - Contrast test (thresholding) and edge test (through Hessian) to eliminate keypoints
    - Create gradient orientation histogram weighted by gradient magnitudes, split keypoint if there are peaks 80% above the highest peak
    - Calculate a 8-bin histogram of gradients for each 4x4 sub-window in 16x16 patch around the keypoint, sub-window gradients are Gaussian weighted
    - Invariances: subtract keypoint rotation from each orientation, clamp feature values to 0.2 at max and then normalize the vectors

    - **Penn State**: Each keypoint has a center point (location), an orientation (rotation) and a radius (scale). At this point, we could try to correlate patches (after first normalizing to a canonical orientation and scale).
- **Wikipedia**: Speeded up robust features (SURF)
    - It is partly inspired by the scale-invariant feature transform (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.
    - To detect interest points, SURF uses an integer approximation of the determinant of Hessian blob detector, which can be computed with 3 integer operations using a precomputed integral image. Its feature descriptor is based on the sum of the Haar wavelet response around the point of interest. These can also be computed with the aid of the integral image.
- **Penn State**: Correlation based algorithms (dense set of correspondences) and feature based algorithms (sparse set of correspondences)
- Shape context
    - Algorithm:
        - Compute shape context descriptors for both sets of points. log-polar bins are used to compute the shape context.
        - Estimate cost matrix between two sets of descriptors
        - Use cost matrix to solve the correspondence problem between two sets of descriptors (e.g. with Hungarian algorithm) — minimize total cost of matching such that matching is one-to-one
        - From the correspondence, estimate a transformation ($T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$) from template to target points (e.g. with thin plate splines) and perform this transformation on the template points
        - Iterate over the steps
    - k-NN can be used for classification on the computed bending energies
    - Scale invariance is obtained by normalizing all radial distances by the mean distance α between all the point pairs in the shape, although median distance can also be used.
    - One can provide complete rotation invariance in shape contexts. One way is to measure angles at each point relative to the direction of the tangent at that point (since the points are chosen on edges). This results in a completely rotationally invariant descriptor. But of course this is not always desired since some local features lose their discriminative power if not measured relative to the same frame. Many applications in fact forbid rotation invariance e.g. distinguishing a "6" from a "9".
- R-HOG blocks are computed in dense grids at some single scale without orientation alignment, unlike SIFT.
- Color spaces:
    - RGB: red-green-blue [0-255]
    - CIE LAB: L = lightness [0-100], a* = green-red, b* = blue-yellow
    - HSL (HSV): H = hue, S = saturation, L = lightness

4 - Motion extraction, optical flow

- Motion is a basic cue, has many applications
- Optical Flow
    - Definition: apparent motion of brightness patterns

- Ideally, the optical flow is the projection of the three dimensional motion vectors on the image
- Two examples where following brightness patterns is misleading:
  - Untextured, rotating sphere
  - No motion, but changing lighting
- Our task is to figure out to which pixel in the next frame it moves. We assume these corresponding pixels have the same intensities as the pixels the scene points came from in the previous frame
- **Motion Field vs Optic Flow**
  - Motion Field: projection of 3D relative velocity vectors onto the 2D image plane.
  - Optic Flow: observed 2D displacements of brightness patterns in the image.
  - Motion field is what we want to know. Optic flow is what we can estimate. Sometimes optic flow is a good approximation to the unknown motion flow. We can then infer relative motion between the camera and objects in the world
- Optical flow constraint (Brightness Constancy) equation: $\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$, where $I(x, y, t)$ is brightness at (x, y) at time t
  - This equation states that if one were to track the image projections of a scene point through the video, it would not change its intensity. This tends to be true over short lapses of time.
  - $\frac{dI}{dt}$: Change of intensity when following a physical point through the images
  - $\frac{\partial I}{\partial t}$: Change of intensity when looking at the same pixel (x,y) through the images
  - Shorthand notation: $I_x u + I_y v + I_t = 0$, aperture problem: 1 equation but 2 unknowns
  - **Wikipedia**: The aperture problem: The motion direction of a contour is ambiguous, because the motion component parallel to the line cannot be inferred based on the visual input. This means that a variety of contours of different orientations moving at different speeds can cause identical responses in a motion sensitive neuron in the visual system.
  - **Penn State**: The component of the flow in the gradient direction is determined (called Normal Flow). The component of the flow parallel to an edge is unknown.
- Horn & Schunck algorithm
  - Breaking the spell via an additional smoothness constraint to be minimized besides the OF constraint equation term.
  - Minimize $u_x^2 + u_y^2 + v_x^2 + v_y^2$, alongside the OF constraint equation term.
  - Coupled partial derivative equations (PDEs) solved using iterative methods and finite differences (iteration i)
    - $\frac{\partial u}{\partial i} = \triangle u - \lambda(I_x u + I_y v + I_t)I_x$
    - $\frac{\partial v}{\partial i} = \triangle v - \lambda(I_x u + I_y v + I_t)I_y$
  - More than two frames allow for a better estimation of I_t
  - Information spreads from edge- and corner-type patterns
  - Errors at object boundaries
  - Example of regularization (selection principle for the solution of ill-posed problems by imposing an extra generic constraint, like here smoothness)
- Condensation filter
  - x_t: state vector, z_t: observation vector, w_t: noise in the system model, v_t: noise in measurement model

        - Prediction: $x_t = f_{t-1}(x_{t-1}, w_{t-1})$, based on the system model, f is the system transition function
        - Update: $z_t = h_t(x_t, v_t)$, based on the measurement model, h is the measurement function
        - The probability distribution is represented by a sample set with weights determining the sampling probability
        - In the limit (large N) equivalent to Bayesian tracker
        - Diversification through noise is important, as otherwise fewer and fewer different samples would survive
        - **Wikipedia**: Sequential Importance Resampling (SIR) filters with transition prior probability distribution as importance function are commonly known as bootstrap filter and condensation algorithm.
    - Comparison with Kalman filters
        - Condensation
            - Unrestricted system and noise models, multiple hypotheses
            - Discretization error, postprocessing for interpretation
        - Kalman-Bucy
            - Linear system models, Gaussian noise, unimodal
            - Exact solution, direct interpretation

Filter	State Space	Belief	Efficiency
Histogram Filters	Discrete	Multimodal	Exponential
Kalman Filters	Continuous	Unimodal	Quadratic
Particle Filters	Continuous	Multimodal	?

- Other approaches
    - Model-based tracking (application-specific)
        - active contours (discussed with segmentation)
        - analysis/synthesis schemes
    - Feature tracking (more generic)
        - corner tracking (shown when we discuss 3D)
        - blob/contour tracking
        - intensity profile tracking
        - region tracking

5 - Multiple-view geometry

- Epipoles
    - projection of center in other image
    - vanishing point of camera motion direction
    - intersection of baseline with image plane
- Fundamental matrix F
    - Geometric derivation: $x' = H_\pi x$, $l' = e' \times x' = [e']_\times H_\pi x = Fx$
    - Algebraic derivation: $X(\lambda) = P^+ x + \lambda C$, $l' = P'C \times P'P^+ x$, $F = [e']_\times P' P^+$ (does not hold true for $C = C'$)
    - Correspondence condition: $x'^T Fx = 0, \forall (x, x')$

    - Has **7 DOF, 3x3-1 (homogeneous)-1 (rank 2)**
    - If F is fundamental matrix for (P, P'), then F^T is fundamental matrix for (P',P)
    - For pure translation F only has **2 degrees of freedom**: $F = [e']_\times$ as $P = K[I|0]$ and $P' = K[I|t]$
    - Homographies: $e' = H_\pi e$, $l' = H_\pi^{-T} l$, $H = [e']_\times F$, **homogprahies have 8 DOF**
    - Canonical form: $P = [I|0]$ and $P' = [M|m]$ then $F = [e']_\times M$ where $e' = m$
    - Epipoles: $Fe = 0$, $F^T e' = 0$
    - F is a correlation, a projective map taking a point to a line. There is no inverse mapping, and F is not of full rank. For this reason, F is not a proper correlation (which would be invertible).
    - Note that in this case of pure translation $F = [e']_\times$ is skew-symmetric and has **only 2 degrees of freedom**, which correspond to the position of the epipole.
    - F only depends on projective properties of the cameras P, P'. If H is a 4 x 4 matrix representing a projective transformation of 3-space, then the fundamental matrices corresponding to the pairs of camera matrices $(P, P')$and $(PH, P'H)$ are the same.
    - A pair of camera matrices (P, P') uniquely determine a fundamental matrix F, but the converse is not true.
    - F allows reconstruction up to a projective transformation. Plane at infinity allows affine reconstruction. Calibrated cameras allow Euclidean reconstruction.
- Essential matrix E
    - The essential matrix is the specialization of the fundamental matrix to the case of normalized image coordinates.
    - Normalized coordinates: $\hat{x} = K^{-1}x = [R|t]X$, camera matrix $K^{-1}P = [R|t]$ is called a normalized camera
    - Defining equation: $\hat{x}'^T E\hat{x} = 0$, then $x'^T K'^{-T} E K^{-1} x = 0$, gives $E = K'^T F K$
    - For a pair of normalized camera matrices $P = [I|0]$ and $P' = [R|t]$, essential matrix has the form $E = [t]_\times R = R[R^T t]_\times$
    - The essential matrix, $E = [t]_\times R$, has only **five degrees of freedom**: both the rotation matrix R and the translation t have three degrees of freedom, but there is an overall scale ambiguity – like the fundamental matrix, the essential matrix is a homogeneous quantity.
    - The reduced number of degrees of freedom translates into extra constraints that are satisfied by an essential matrix, compared with a fundamental matrix: A 3×3 matrix is an essential matrix **if and only if two of its singular values are equal, and the third is zero**.
    - For a given essential matrix $E = Udiag(1, 1, 0)V^T$, and first camera matrix $P = [I|0]$, there are four possible choices for the second camera matrix P', namely $P' = [UWV^T| + \mathbf{u}_3]$, $P' = [UWV^T| - \mathbf{u}_3]$, $P' = [UW^TV^T| + \mathbf{u}_3]$, $P' = [UW^TV^T| - \mathbf{u}_3]$. Testing with a single point to determine if it is in front of both cameras is sufficient to decide between the four different solutions for the camera matrix P'. $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
    - Each point pair contributes with one constraining equation on the element in E. Since E has five degrees of freedom it should therefore be sufficient with only five point pairs to determine E. Though possible from a theoretical point of view, the practical implementation of this is not straightforward and relies on solving various non-linear equations.
- Computation of the Fundamental Matrix F

- Basic equations:
  $(x'x, x'y, x', y'x, y'y, y', x, y, 1) \cdot (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}) = 0$, for at least 7 (x, x')
- For a solution to exist, matrix A must have rank at most 8, and if the rank is exactly 8, then the solution is unique (up to scale), and can be found by linear methods – the solution is the generator of the right null-space of A.
- If the data is not exact, because of noise in the point coordinates, then the rank of A may be greater than 8 (in fact equal to 9, since A has 9 columns). In this case, one finds a least-squares solution.
- If the fundamental matrix is not singular then computed epipolar lines are not coincident.
- 8-point algorithm for computation of the fundamental matrix may be formulated as consisting of two steps:
  - Linear solution: A solution F is obtained from the vector f corresponding to the smallest singular value of A, for $A\mathbf{f} = 0$
  - Constraint enforcement. Replace F by F', the closest singular matrix to F under a Frobenius norm. This correction is done using the SVD.
- In the case where the matrix A has rank seven, it is still possible to solve for the fundamental matrix by making use of the singularity constraint. The most important case is when only 7 point correspondences are known
- **Penn State**
  - Unlike a homography, where each point correspondence contributes two constraints (rows in the linear system of equations), for estimating the essential/fundamental matrix, each point only contributes one constraint (row). [because the Longuet-Higgins / Epipolar constraint is a scalar eqn.] Thus need at least 8 points.
  - Eight point algorithm:
    - Construct the $m \times 9$ matrix A, for $m \geq 8$ point correspondences, find the SVD of A: $A = UDV^T$
    - The entries of F are the components of the column corresponding to the least singular value (eigenvector with the smallest eigenvalue)
    - Enforce rank 2 constraint: find SVD, set the smallest s.v. of F to 0, recompute F
- Polar rectification
  - Polar re-parameterization around epipoles, preserve length of epipolar lines, guarantees minimal image size, requires only oriented epipolar geometry
  - For traditional stereo applications the limitations of standard rectification algorithms are not so important. The main component of camera displacement is parallel to the images for classical stereo setups. The limited vergence keeps the epipoles far from the images.
- The normalized 8-point algorithm:
  - Normalization: Transform the image coordinates according to $\hat{x}_i = Tx_i$, $\hat{x}'_i = Tx'_i$, where T and T' are normalizing transformations consisting of a translation and scaling. The suggested normalization is a translation and scaling of each image so that the centroid of the reference points is at the origin of the coordinates and the RMS distance of the points from the origin is equal to $\sqrt{2}$. (Orders of magnitude difference between columns of data matrix)

        - Linear solution and constraint enforcement: gives $\hat{F}'$. Note that it is recommended that the singularity condition should be enforced before denormalization.
            - SVD from linearly computed F matrix (rank 3): $F = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} V^T$
            - Compute closest rank-2 approximation $min||F - F'||_F$: $F = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix} V^T$
        - Denormalization: Set $F = T'^T \hat{F}' T$ to obtain the original fundamental matrix.
    - **Hartley**: Linear estimation of projective transformation parameters from point correspondences often suffer from poor "conditioning" of the matrices involves. This means the solution is sensitive to noise in the points (even if there are no outliers).
    - Recommendations:
        - For a quick method, easy to implement, use the normalized 8-point algorithm. This often gives adequate results, and is ideal as a first step in other algorithms.
        - If more accuracy is desired, use the algebraic minimization method, either with or without iteration on the position of the epipole.
        - To be certain of getting the best results, if Gaussian noise is a viable assumption, implement the Gold Standard algorithm.
    - Let E be a 3×3 matrix with SVD given by $E = UDV^T$, where $D = diag(a, b, c)$ with $a \geq b \geq c$. Then the closest essential matrix E in Frobenius norm is given by $\hat{E} = U\hat{D}V^T$, where $\hat{D} = diag((a+b)/2, (a+b)/2, 0)$
- Structure Computation
    - Simple linear triangulation methods: As usual the estimated point does not exactly satisfy the geometric relations, and is not an optimal estimate
    - Practical problems: geometric distortion (can be compensated to some extent), dispersion of the light rays, discretization of the intensity in digital cameras (have to use interpolated values), the fact that the points are found through feature extractors
    - Triangulation: $\begin{bmatrix} P_3 x - P_1 \\ P_3 y - P_2 \\ P'_3 x' - P'_1 \\ P'_3 y' - P'_2 \end{bmatrix} X = 0$
    - Optimal 3D point in epipolar plane: select closest points on epipolar lines, obtain 3D point through exact triangulation, guarantees minimal projection error given this epipolar plane
    - Reconstruct matches in projective frame by minimizing the reprojection error: **has 3 DOF**
    - Non-iterative method: determine the epipolar plane for reconstruction (polynomial of degree 6, **has 1 DOF**, only works for two views)
- Image Rectification
    - Given a pair of stereo images, the intrinsic parameters of each camera, and the extrinsic parameters of the system (R and T), compute the image transformation that makes epipolar lines collinear and parallel to horizontal axis
    - In practice, it is convenient if image scanlines are the epipolar lines (the search is simplified to one dimension). Problem when epipole in (or close to) the image.

6 - Model fitting

- Hough transform: see Lecture 7.
- Line fitting can be a max likelihood, but choice of model is important (e.g. standard least squares vs total least squares)
- **Incremental line fitting**: walk along a curve, fit a line to runs of pixels along the curve, break the curve when residual is too large. Lines can be fit by k-means
- M-estimators
  - **Wikipedia**: In statistics, M-estimators are a broad class of extremum estimators for which the objective function is a sample average.
  - A quadratic function gives too much weight to outliers, instead use robust norm: $\rho(r, \sigma) = \frac{r^2}{\sigma^2 + r^2}$, influence function: $\psi = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$, choosing the scale is critical.
- **RANSAC** (Random Sample Consensus)
  - Choose a small subset uniformly at random, fit to that, anything close is signal and all others are noise, refit, do this many times and choose the best
  - How big a subset: smallest possible, what does close mean: depends on the problem, how many times: enough that we are likely to have a good model
  - What is a good line: one where the number of points is so big it is unlikely to be all outliers
  - Distance threshold (t): often chosen empirically, choose t so probability for inlier is α (e.g. 0.95)
  - **Penn State**: Multiple structures can also skew the results. (the fit procedure implicitly assumes there is only one instance of the model in the data).
  - Codimension: DOF of the error measure
    - For example, it is 1 for line fitting because error is the perpendicular distance. Similarly, it is 2 for point distance.
    - Codimension = 1 (line, F) $\rightarrow t^2 = 3.84\sigma^2$
    - Codimension = 2 (H, P) $\rightarrow t^2 = 5.99\sigma^2$
    - Codimension = 3 (T) $\rightarrow t^2 = 7.81\sigma^2$
  - How many samples? $p = 1 - (1 - (1 - e)^s)^N$
    - e: probability that a point is outlier
    - s: number of points in a sample
    - N: number of samples
    - p: desired probability that we get a good sample
  - Acceptable consensus set: $T = (1 - e)n$, Terminate when inlier ratio reaches expected ratio of inliers
  - Adaptively determining the number of samples
    - e is often unknown a priori, so assume worst case, e.g. 0.5, and adapt if more inliers are found
  - QDEGSAC and DEGENSAC for dealing with quasi-degenerate data
    - RANSAC gets confused by quasi-degenerate data
    - QDEGSAC is general and can deal with other robust estimation problems
    - QDEGSAC does not require to know which degeneracies can occur
  - Missing variables strategy: iterate until convergence
    - replace missing variables with expected values, given fixed values of parameters
    - fix missing variables, choose parameters to maximize likelihood given fixed values of missing parameters

        - Rule of thumb: it is better to fit to the better fitting points, within reason; if this is hard to do, then the model could be a problem
    - Issues with EM
        - Local maxima: can be a serious nuisance in some problems, no guarantee that we reached the right maximum
        - Starting: k-means to cluster the point is often a good idea
    - Model selection: Discounts (e.g. AIC, BIC, MDL), cross-validation, model averaging

7 - Image segmentation

- Semantic segmentation: recognizing, understanding what's in the image in pixel level
- Interactive segmentation: To impose constraints on the segmentation, interactive segmentation involves user interaction to indicate the “objectiveness” and thus to guide an accurate segmentation
- Grouping rules (Gestalt factors): similarity, common fate (similar motion), symmetry, proximity, closure, smoothness
    - Gestalt rules can never be translated into satisfactory algorithms.
- Examples of segmentation techniques
    - Clustering: thresholding, k-means, mean-shift
    - Edge-based: hough transform, dynamic path search
    - Learning: k-nearest neighbors, random forests, deep learning
- Feature space: intensity (1D), color (3D), filter bank responses (e.g. 24D) for texture similarity, intensity + position
- Thresholding
    - For high contrasts between objects and background
    - If area of object(s) and background is known: place threshold at intensity that yields appropriate proportions
    - Several techniques that include edge information:
        - maximize sum of gradient magnitudes for the pixels with threshold intensity
        - use a histogram taking only low gradient magnitude pixels into account
    - The Otsu criterion:
        - Tries to find a threshold that yields maximally homogeneous intensity areas for values above and below the threshold value
        - Minimize within-group variance of intensity: determine threshold to minimize $p_1\sigma_1^2 + p_2\sigma_2^2$, where p is fraction of pixels, σ variance of intensities
    - **Pros**: simplest algorithm ever, serious bandwidth reduction (binary output), simplification of further processing, availability of cheap and super-fast hardware
    - **Cons**: generally it won’t provide a satisfying segmentation
    - Binary enhancement:
        - Mathematical morphology basics: operations on binary images, shift-invariant, non-linear, based on neighboring pixels
        - Erosion: turn pixel into background as soon as there is a single background pixel around, Dilation: same for foreground
        - Opening: erosion, then dilation. Closing: dilation, then erosion.
        - Rank order op.: $i_1 \leq i_2 \leq \ldots \leq i_N$, $i_t = f_t(i_1, \ldots, i_N)$, Erosion: $i_t = i_1$, Dilation: $i_t = i_N$, Median: $i_t = i_{(N+1)/2}$

        - **Remarks**
            - Result not identical to noise-free original
            - Erosion + dilation $\neq$ dilation + erosion
            - Use same neighbourhood for both steps
            - Noise in background can be reduced by reversed operation (opening and closing)
            - Reminder: median filtering useful for edge preserving smoothing
        - **Wikipedia**: Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise.
- k-means clustering
    - Best cluster centers are those that minimize SSD between all points and their nearest cluster center
    - k-means will always converge to some solution, can be a local minimum (not always the global minimum of the objective function)
    - **Pros**: simple, fast to compute, converges to local minimum of within-cluster squared error
    - **Cons**: setting k (automatically choose e.g. based on low segment variance), sensitive to initial centers, sensitive to outliers, detects spherical clusters only
- Mean-shift
    - Finding modes in a histogram, mode = local maximum of a given distribution
    - Iterative Mode Search in the feature space: initialize random seed center and window, calculate the mean of W, shift the search window to the mean, repeat
    - Cluster: all data points in the attraction basin of a mode, Attraction basin: the region for which all trajectories lead to the same mode
    - Mean-shift clustering: Initialize windows at feature values for individual pixels, start mean-shift from each window until convergence, merge windows that end up near the same “peak” or mode
    - **Pros**:
        - General, application-independent tool, robust to outliers
        - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
        - Just a single parameter (window size h), finds variable number of modes given the same h
    - **Cons**:
        - Output depends on window size h, window size h (bandwidth) selection is not trivial
        - Computationally rather expensive and doesn’t scale well with dimension of feature space (sparsity problems…)
- EM Segmentation
    - We assume that the colors originate from a probability density given by a Gaussian Mixture Model (GMM)
    - We can maximize the expected value of the complete log-likelihood and iterate expectation and maximization steps
    - Here, we assume the number of segments K is known
- Edge-based segmentation
    - Clustering is not enough, edges can help
    - Convolution not enough: gaps or too thick (gradient magnitude), unnaturally closed (Laplacian)

    - Zero-crossing: jump across objects, edges in homogeneous areas
    - **Edinburgh**: The zero crossing detector looks for places in the Laplacian of an image where the value of the Laplacian passes through zero— i.e. points where the Laplacian changes sign. Such points often occur at 'edges' in images — i.e. points where the intensity of the image changes rapidly, but they also occur at places that are not as easy to associate with edges. Often zero crossings are found in regions of very low gradient where the intensity gradient wobbles up and down around zero.
    - Difficulties — Hough transform: add prior info about edge shape, Dynamic path search: add human initialization
- Hough transform
    - **Wikipedia**: The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.
    - Uses the symmetry of shapes to extract them in a parameter space of limited dimension
    - Straight line equation: $y = ax + b$, for a fixed point: $b = (-x)a + y$
    - Scrutinize all points of interest (all candidate edge points), draw the line in (a, b) parameter space
    - Problem: unbounded parameter domain, vertical lines require infinite a, alternative representation: $x \cos\theta + y \sin\theta = \rho$, each point will add a cosine function in the (θ, ρ) parameter space
    - How many lines: count the peaks in the Hough array — Who belongs to which line: tag the votes
    - Hardly ever satisfactory in practice, because problems with noise and cell size defeat it
    - Difficulties: How big should the cells be? (too big and we cannot distinguish between quite different lines; too small and noise causes lines to be missed)
    - **Remarks**: time consuming, robust (noise points unlikely to contribute consistently to any single bin), peak detection is difficult, spurious peaks due to uniform noise
    - Robustness of peak detection increased by weighting contributions (votes that are proportional with intensity gradient magnitude)
- Dynamic path search
    - Guided by a user-supplied cost function, find optimal path having lower cost and satisfying constraints (e.g. given endpoints)
    - Useful in interactive applications (e.g. medical) or when environment is constrained
    - Nomenclature:
        - A graph consists of nodes (pixels) connected by arcs (steps), nodes connected by steps are parents and successors
        - Identifying a node's successors is expansion of that node, a tree is a graph with 1 parent for the nodes (our arcs are undirected)
        - Often the arcs are assigned a cost, a sequence of nodes is a path, usually path cost = $\sum$ arc costs
    - Cost function incorporates problem-specific information: penalize changes in the edge direction, penalize the inclusion of pixels with low intensity gradient, etc. The structure of the cost function to a large extent determines how difficult the optimization is.
    - **Wikipedia**: Best-first search is a search algorithm which explores a graph by expanding the most promising node chosen according to a specified rule.
    - Best-first search

        - Uses problem-specific information to guide the process selectively
        - Returns the optimal solution if applied properly
    - BF*, Z, Z*, etc.
    - **Redblobgames**: Greedy Best First Search: Use the estimated distance to the goal for the priority queue ordering. The location closest to the goal will be explored first.
    - **Redblobgames**: Dijkstra's Algorithm works well to find the shortest path, but it wastes time exploring in directions that aren't promising. Greedy Best First Search explores in promising directions but it may not find the shortest path. The A* algorithm uses both the actual distance from the start and the estimated distance to the goal.
    - A* algorithm
        - **Wikipedia**: At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A* selects the path that minimizes $f(n) = g(n) + h(n)$, where n is the next node on the path, g(n) is the cost of the path from the start node to n, and h(n) is a heuristic function that estimates the cost of the cheapest path from n to the goal.
        - **Wikipedia**: If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal.
- Learning
    - Objects $\rightarrow$ measurements $\rightarrow$ features $\rightarrow$ object classes
    - Training data: application dependent, very valuable, can be very expensive, the "ground-truth" segmentations are often done manually or with a semiautomatic algorithm
    - k-NN
        - The mapping is defined through the labels of the K-nearest neighbors
        - **Pros**: very simple to implement and understand, distance definition is flexible, efficient implementations possible for approx. NNs
        - **Cons**: highly depends on the definitions and K, need to keep the entire data in memory for distance computations, for high dimensional problems need a lot of training samples for accuracy
    - Random forests
        - Random forest: consists of an ensemble of different decision trees — hence forest
        - Training: deciding on the (binary) node tests, deciding on when to stop splitting
        - The goal is to find a node test that maximally reduces the uncertainty of class membership for samples ending up in the child nodes, compared to at their parent node. Uncertainty can be quantified through the entropy of the class distribution at a node.
        - Another aspect of the randomness is that different trees can be trained with different, random subsets of the training samples
        - **Pros**: easy to implement, very efficient during testing, can easily use diverse features, can handle high dimensional spaces
        - **Cons**: lots of parametric choices, needs large number of data, training can take time
    - Deep learning
        - Better integration of segmentation and recognition, yields superior results, needs lots and lots of training data
        - One of the biggest strengths of deep learning is that **it also designs the features** to be used by the classifier

8 - Stereo and MVS

- Standard stereo geometry: $F = [t]_\times$, $\frac{B}{Z} = \frac{d}{f}$, B is baseline, d is disparity, Z is depth, f is focal length
- $d = -\frac{Bf}{Z}$ Disparity is inversely proportional to depth. $\triangle Z = -\frac{Z^2}{Bf}\triangle d$
- Correspondence problem: determining which pixel in the right image corresponds to which pixel in the left image.
- Stereo matching: search along the epipolar line, look for the most similar pixel (e.g. using SSD, SAD, NCC, rank transform, census transform)
- **When will the similarity constraint fails?**
    - Textureless surfaces, occlusions, repetition, non-Lambertian surfaces, specularities
- **Disparity Space Image**
    - The DSI for one row represents pairwise match scores between patches along that row in the left and right image.
    - DSI and Scanline Consistency: Assigning disparities to all pixels in left scanline now amounts to finding a connected path through the DSI.
    - Ordering constraint: It is common to impose an ordering constraint on the path. Intuitively, the path is not allowed to “double back” on itself.
    - Cox et.al. Stereo Matching: Dynamic programming over costs w.r.t. ordering constraints.
    - Occlusion Filling: Fill in left occluded pixels with value from the nearest valid pixel preceding it in the scanline. Similarly, for right occluded, look for valid pixel to the right.
- Pixel dissimilarity
    - Boundary overreach (surface over-extension): The recovered object boundary turns out to be wrongly located away from the real one due to the window’s coverage beyond a boundary
    - Offset windows: equivalent to using min nearby cost, result: loss of depth accuracy. Use offset windows only where appropriate.
    - Compact windows: adapt windows size (based on average matching error per pixel, variance of matching error, window size — biased towards larger windows) and pick window that minimizes the cost
    - Rod-shaped filters: instead of square windows aggregate cost in rod-shaped shiftable windows
    - Locally adaptive support: apply weights to contributions of neighboring pixels according to similarity and proximity
- Challenges
    - Ill posed inverse problem: recover 3D structure from 2D information
    - Difficulties: uniform regions, half-occluded pixels
- Scene constraints
    - Similarity constraint, epipolar constraint, continuity constraint (disparity is piecewise smooth)
    - Ordering constraint: Points are not always on the same order on both epipolar lines
    - Uniqueness constraint: In an image pair each pixel has at most one corresponding pixel
- Image pair rectification: simplify stereo matching by warping the images
    - Apply projective transformation so that epipolar lines correspond to horizontal scanlines
    - Standard stereo setup: epipoles are at infinity and epipolar line are parallel

- Optimal baseline: too small = large depth error, too large = difficult search problem, more occlusions
- **Binocular stereo review**
  - The basic stereo matching algorithm: similarity constraint, no coherence enforced between matches
  - Non-local constraints: uniqueness, ordering, smoothness
  - Scanline stereo through dynamic programming
    - Coherent optimization of all matches along an entire scanline, independent optimization of different scanlines
    - Scanline stereo generates streaking artifacts
    - Can't use dynamic programming to find spatially coherent disparities/ correspondences on a 2D grid
  - Coherent 2D stereo: energy minimization, approximate solutions found via graph cuts
- **Multiple view stereo**
  - The third view can be used for verification. Use the sum of SSD scores to rank matches.
  - Pros: Using multiple images reduces the ambiguity of matching
  - Cons: Must choose a reference view, Occlusions become an issue for large baseline
  - Some solutions for visibility issues: multi-view linking, best of left or right, best k out of n
- **Stereo matching overview**
  - The main underlying assumption that allow to search for conjugate points is that image patches that are projection of the same surface patch are similar.
    - This may not be true because of occlusions, non-Lambertian lighting effect and perspective
  - All methods attempt to match pixels in one image with pixels in the other image by exploiting a number of constraints.
    - Local methods: use constraints on a small number of pixels surrounding the pixel of interest (e.g. block matching)
    - Global methods: use constraints on scan-lines or the whole image (e.g. dynamic programming, graph cuts)
  - Reliability-accuracy tradeoff: solutions are adaptive/shiftable windows, hierarchical approaches
  - Block matching and gradient-based methods are sensitive to depth discontinuities and uniform regions
  - Limit the correspondence search to reliable features in the images (e.g. Harris corners) $\rightarrow$ Matches are reliable but sparse
  - Segmentation based: first segment the images and then match the segmented regions $\rightarrow$ Produces dense maps but it is sensitive to the original segmentation
  - Dynamic programming: The original (2D) problem is decomposed in several simpler ones (1D), vertical coherence is lost (though it may be incorporated)
  - Graph cuts: The DSI becomes a graph; capacity of edges defined as a function of the cost of adjacent nodes, the min cut is analogous to the best path along a pair of scanlines determined by DP but extended to 3D
  - **Scharstein & Szeliski's taxonomy**: Matching cost computation (etc. SSD, SAD, NCC), cost aggregation (summing or averaging over a support region of DSI), disparity computation/optimization (local: winner-takes-all, global: energy minimization, dynamic programming etc.), disparity refinement (post-processing: sub-pixel, occlusion

detection)

9 - Structure-from-motion

- Goal: combine point correspondence information from multiple points over multiple frames to solve for scene structure and camera motion (structure from motion)
- **Ambiguity**:
  - In the general case (nothing is known) the ambiguity is expressed by an arbitrary affine or projective transformation
  - The ambiguity exists even for calibrated cameras. For calibrated cameras, the similarity ambiguity is the only ambiguity.
  - Scale ambiguity: it is impossible based on the images alone to estimate the absolute scale of the scene
- Factorization
  - Approach: numerically stable approach based on using SVD to "factor" matrix of observed point positions
- Data Association: connected components from unstructured images though feature extraction, feature matching and geometric verification
- **Structure-from-Motion**: from relative to absolute cameras and structure
  - Incremental SfM
    - Initialization: choose two non-panoramic views, triangulate inlier correspondences, bundle adjustment
    - Absolute camera registration: find 2D-3D correspondences, solve perspective-n-point problem
    - Outlier filtering: remove points with large reprojection error, remove points at infinity
    - **Loop is** image registration, triangulation, bundle adjustment, outlier filtering
  - Global SfM
    - Estimate global rotations: $\min_R ||R_{ij} - R_j R_i^T||$, filter relative rotations
    - Estimate global translations, filter relative translations
    - Triangulate and refine with bundle adjustment
  - Hierarchical SfM
    - Hierarchical clustering of scene graph
    - Reconstruct clusters independently (using incremental or global SfM)
    - Merge clusters using similarity transformations

Method	Efficiency	Robustness	Accuracy
Incremental	-	++	+
Global	+	+	+
Hierarchical	++	-	-

- Bundle adjustment
  - Non-linear refinement of structure and motion
  - Minimize projection error: $\min_{P,X} ||x - \pi(P, X)||$
  - **Wikipedia**: Given a set of images depicting a number of 3D points from different viewpoints, bundle adjustment can be defined as the problem of simultaneously

refining the 3D coordinates describing the scene geometry, the parameters of the relative motion, and the optical characteristics of the camera(s) employed to acquire the images, according to an optimality criterion involving the corresponding image projections of all points.
    - Huge problem but can be solved efficiently: Newton iteration, (Sparse) Levenberg-Marquardt
    - Why it is hard: many parameters, poorer conditioning (high correlation), potentially lots of outliers, gauge (coordinate) freedom
    - Lots of parameters: sparsity $\rightarrow$ only a few entries in Jacobian are non-zero, $J^T J$ needed for minimization (first order approximation of the Newton's approach)
    - Sparse bundle adjustment: eliminate dependence of camera / motion parameters on structure parameters
- Challenges (of crowdsourced data)
    - WTFs: watermarks, timestamps, frames (detect pure translations at image border)
    - Calibration: focal length unknown, EXIF inaccurate/missing, image distortion
    - Scale ambiguity: inherent in SfM, use GPS EXIF tags for geo registration, use semantics to infer scale
    - Dynamic objects: standard SfM formulation only for static objects
    - Repetitive structures: symmetries in man-made structures
        - Pre-processing: remove inconsistent graph edges
        - Post-processing: identify and correct duplicate structures
    - Illumination change: day-night matching difficult / not possible, leverage transition during dusk/dawn
    - Initial view selection: trade-off of triangulation angle and number of correspondences
    - Next-best view selection: maximize number of 2D-3D correspondences
    - Scalability: $O(N^3)$ complexity in bundle adjustment, inexact step algorithms (PCG) have $O(N)$ complexity
- Large-scale reconstruction: skeletal graphs

10 - Specific object recognition

- Model-based recognition
    - Relate image features to model features
    - Simplify the relation through invariance
    - Complexity of objects limited
    - Can deal with cluttered scenes
    - Hypothesize-and-verify process is slow
    - More recent example: invariant-based recognition of planar shapes
- Image-based (appearance based) recognition
    - Model equals all possible views (training is just taking pictures, no sophisticated models)
    - Acquiring representative data set is cumbersome, sometimes not possible at all
    - Can deal with complex objects (generic, any kind of object, 3D shapes)
    - Difficulties with variable backgrounds (need for a clean background)
- Scale invariant features
    - Laplacian of Gaussian for blob detection in 2D
    - Characteristic scale: peak of Laplacian response

    - Scale-normalized LoG: $\nabla^2_{norm} g = \sigma^2 (\frac{\sigma^2 g}{\sigma x^2} + \frac{\sigma^2 g}{\sigma y^2})$
- Affine invariant features: idea is to use local affine and photometric invariant features
    - Intensity based method: see Lecture 3
    - Taking f to be the characteristic function of a region (1 inside, 0 outside), moments of orders up to 2 allow to approximate the region by an ellipse.
    - The ellipse will have the same moments of orders up to 2 as the original region.
- MSER: see Lecture 3
- Invariance vs. covariance
    - Invariance: features(transform(image)) = features(image)
    - Covariance: features(transform(image)) = transform(features(image))
    - **Covariant detection, invariant description**
- Feature description
    - Affine normalization of ellipses to unit circles: problem of not having a unit transformation. Need to eliminate rotational ambiguity.
    - Simplest descriptor: vector of raw intensity values
        - Compare with SSD: Not invariant to intensity change
        - Compare with NCC: Invariant to affine intensity change
        - Small misalignments can affect the matching score a lot
    - SIFT descriptor: see Lecture 3
- Feature matching
    - Simple algorithm to generate putative matches
        - Compare all pairs of descriptors
        - Keep all pairs with distance below a threshold
    - Heuristic: compare distance of nearest neighbor to that of second nearest neighbor. Ratio of closest distance to second-closest distance will be high for features that are not distinctive. Threshold of 0.8 provides good separation.
- **Comparison of different approaches**
    - Pure model based
        - Compact model, can deal with clutter
        - Slow analysis-by-synthesis, models difficult to produce, for limited object classes
    - Pure appearance based
        - Large models, cannot deal with clutter
        - Efficient, models easy to produce, for wide classes of objects
    - Hybrid techniques
        - Rather compact model, can deal with clutter and partial occlusion
        - Efficient, model easy to produce (take images, fewer than in pure appearance based), for rather wide class of objects (almost as wide as in pure appearance based)
- Dealing with highly challenging imaging conditions
    - Usual approach is to extract features, compute and match descriptors, match and filter (e.g. RANSAC)
    - Difficulties: Loss of information (large scale change), few repeated object regions and less accurate shape (occlusion), many mismatches (extensive clutter)
    - **Image exploration through expansion and contraction**
        - Expansion: project via affine transformation of support, then refine

        - Contraction: sidedness constraint for all triples, remove matches with highest count (allows non-rigid deformations)
    - Power: a single correct match per smooth surface suffices, expansion and contraction help each other

- Scaling to large databases
    - **Fast nearest neighbor search via inverted indexes** (vocabulary trees): hierarchical partitioning of descriptor space
    - Matching complexity depends only on the depth of the tree, O(n) where n is the quantized words (leaves of the tree)
    - Disadvantages: quantization boundary artifacts, no geometric verification of object's presence, approximation depends on number of leaves and densities of space
    - Hypotheses receiving some minimal amount of votes can be subjected to more detailed geometric verification
    - Indexing with geometric invariants: when we don't have feature descriptors, we can take n-tuples of neighboring features and compute invariant features from their geometric configurations (example application: searching the sky)
- **Applications of invariant regions**
    - Wide baseline stereo
    - Image database retrieval
    - Tracking for augmented reality

11 - Shape from X

- Shape from silhouettes: **Visual Hull** (two colors / binary image)
    - The case of binary images: a voxel is photo-consistent if it lies inside the object's silhouette in all views
    - Finding the silhouette-consistent shape: Backproject each silhouette, Intersect backprojected volumes (generalized cones)
    - Volume Intersection (**voxel based**): Easy to implement, fast, accelerated via octrees, no concavities, requires silhouette extraction, reconstruction is not photo-consistent if texture information is available
    - Marching intersections: Split up object space into a 3D grid made of 3 sets of rays rather than voxels, merging them is reduced to 1D intersections along each ray
    - Polyhedral volume intersection: no voxelization artifacts, depends on discretization of outlines, numerical problems when polygons to be intersected are almost coplanar, does not take advantage of epipolar geometry
    - Image-based visual hulls:
        - Pick a pixel in the virtual view, Project corresponding visual ray into every other view to get a set of epipolar lines, Find intervals where epipolar lines overlap with silhouettes, Lift intervals back onto the 3D ray and find their intersection
        - Can work in real time, takes advantage of the epipolar geometry, need to recompute visual hull every time the virtual view is changed
    - Carved visual hulls: Visual hull gives a reasonable initial mesh that can be iteratively deformed, Need silhouette extraction, Have to compute a lot of points that don't lie on the object, Finding rims is difficult, The carving step can get caught in local minima, Possible solution: use sparse feature correspondences as initialization

    - Feature-based stereo matching: Robust to clutter and occlusion, Only find matches at reliable points, Can use invariant local features to deal with foreshortening, scale change, wide baselines, You only get a sparse cloud of points (or oriented patches), not a dense depth map or a complete surface
    - From feature matching to dense stereo: Extract features, Get a sparse set of initial matches, Iteratively expand matches to nearby locations, Use visibility constraints to filter out false matches, Perform surface reconstruction
- Multiple View Geometry of Silhouettes
    - Points on Silhouettes in 2 views do not correspond in general except for projected Frontier Points
    - Always at least 2 extremal frontier points per silhouette
    - 7 or more corresponding frontier points needed to compute epipolar geometry for general motion
    - Hard to find on single silhouette and possibly occluded
- **Voxel coloring**:
    - Goal: assign RGB values to voxels in V, photo-consistent with images
    - A photo-consistent scene is a scene that exactly reproduces your input images from the same camera viewpoints
    - You can't use your input cameras and images to tell the difference between a photo-consistent scene and the true scene
    - True scene $\subset$ Photo-Consistent Scenes $\subset$ All Scenes
- Voxel Coloring Approach (C unconstrained, viewpoint constraints)
    - Depth ordering: visit occluders first
    - Inward/outward looking camera configurations
    - Limitations: a view-independent depth ordering may not exist, need more powerful general-case algorithms
- **Space Carving Algorithm** (general case)
    - Algorithm:
        - Initialize to a volume V containing the true scene
        - Choose a voxel on the current surface, project to visible input images, carve if not photo-consistent
        - Repeat until convergence
    - Which shape do you get: the **Photo Hull** is the UNION of all photo-consistent scenes in V
        - It is a photo-consistent scene reconstruction
        - Tightest possible bound on the true scene
        - All inconsistent points are removed, a point on the true scene is never removed
    - Transparency not supported
- Multi-pass plane sweep
    - Basic space carving algorithm is complex, alternative solution
        - Easy to implement, converges quickly in practice
        - Efficient, can use texture mapping hardware
    - Algorithm
        - Sweep plane in each of 6 principle directions
        - Consider cameras on only one side of plane
        - Repeat until convergence
- Volumetric graph cuts
    - Inside is sink, outside is source, discretize middle volume

    - Assign photoconsistency cost to voxels
    - Cut = 3D surface, minimum cut = minimal 3D surface under photo-consistency metric
    - Protrusion problem: favouring bigger volumes that fill the visual hull (ballooning force)
- **Shape-from-X**: X = shading, texture, focus/defocus, specularities, shadows, multiple light sources (photometric stereo), etc.
- Shape from shading
    - **Wikipedia**: Lambertian reflectance is the property that defines an ideal "matte" or diffusely reflecting surface. The apparent brightness of a Lambertian surface to an observer is the same regardless of the observer's angle of view.
    - Relation between intensity and shape: reflectance map
    - Reflectance map and a known light source are not enough to uniquely determine each point
    - Solution: add more constraints (shape from shading)
        - Image irradiance constraint: image irradiance should match the reflectance map
        - Smoothness constraint: penalize rapid changes in surface orientation over the image
    - Solution: take more images (photometric stereo)
        - Solve the equation system for multiple lights
        - RGB images: get three sets of equations, one per color channel
    - Limitations:
        - Doesn't work for shiny things, semi-translucent things
        - Shadows, inter-reflections

12 - Object category recognition

- Problem definition
    - Specific objects vs object categories, two main tasks: classification and detection
- Visual word: A small patch on the image (array of pixels) which can carry any kind of interesting information in any feature space (color changes, texture changes ...etc.).
- Classification by bag-of-words
    - Visual words: Quantize via clustering, let cluster centers be the prototype “words”, Determine which word to assign to each new image region by finding the closest cluster center
    - Appearance of objects vary a lot within a category, but **appearance of local parts varies less**
    - Issues:
        - Sampling strategy (for object classes dense sampling works best)
        - Clustering / quantization algorithm (many options, often k-means works well enough)
        - What corpus provides features (universal vocabulary?): typically as close as possible to your application
        - Vocabulary size, number of words
    - Bag-of-words: Summarize entire image based on its distribution (histogram) of word occurrences
    - Any histogram comparison measure can be used here (e.g. normalized scalar product between their (possibly weighted) occurrence counts)

    - Classifier choices: nearest neighbor, neural networks, SVMs, boosting, Naive Bayes (assume that each feature is conditionally independent given the class)
    - The bag of words removes spatial layout: **this is both a strength and weakness**
    - **Middle ground**:
        - Visual "phrases" : frequently co-occurring words
        - Semi-local features : describe configuration, neighborhood
        - Add position to each feature descriptor
        - Count bags of words only within sub-grids of an image (Spatial Pyramid Representation)
    - **Pros**:
        - Flexible to geometry / deformations / viewpoint
        - Compact summary of image content
        - Provides vector representation for sets (bags to be precise)
        - Empirically good recognition results in practice
    - **Cons**:
        - Basic model ignores geometry – can verify afterwards, or embed within feature descriptors
        - Background and foreground mixed when bag covers whole image
        - Interest points or sampling: no guarantee to capture object-level parts
        - Optimal vocabulary formation remains unclear
- Classification by CNNs
    - Neural network with specialized connectivity structure
    - Filtering: Dependencies are local, Translation equivariance, Tied filter weights (few parameters), Stride 1,2,... (faster, less memory)
    - Rectified linear unit (ReLU): Simplifies backprop (max operation instead of exponentials), Makes learning faster, Avoids saturation issues
        - **Stackoverflow**: Two additional major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient (constant gradient)
        - **Stackoverflow**: Tend to blow up activation (there is no mechanism to constrain the output of the neuron), dying ReLU problem (alternatively, use leaky ReLU)
    - Overlapping pooling and max pooling (preferred over non-overlapping and sum pooling)
    - Normalization: 0 mean, 1 std, equalizes the feature maps
- Detection by sliding windows (Detection via classification)
    - Global appearance features (e.g. grayscale / color histograms, vector of pixel intensities):
        - Pixel-based representations sensitive to small shifts
        - Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation
    - Gradient-based representations (e.g. HoG):
        - Locally orderless: offers invariance to small shifts and rotations
        - Contrast-normalization: try to correct for variable illumination
    - Edge-based Representations (e.g. Pairs of Adjacent Segments (PAS)):
        - Subdivide window into tiles (more discriminative power)
        - Compute a separate bag-of-PAS for each tile
        - Contour features: better suited for classes defined by shape
    - Boosting (e.g. AdaBoost):

        - see AML notes
        - Rectangular filters: efficiently computable with integral image: any sum can be computed in constant time
    - **Pros**:
        - Simple detection protocol to implement
        - Good feature choices critical
        - Good detectors available (e.g. Viola-Jones, HoG, etc.)
    - **Cons**:
        - High computational complexity
        - With so many windows, false positive rate better be low
        - Typically need fully supervised training data (= bounding-boxes)
        - Some object do not fit a box well (diagonal bottle)
        - Sensitive to partial occlusion (unless in training data)
- **Proposal-based detection**
    - Uses low-level perceptual organization cues
    - Proposal mechanism can be category-independent
    - Proposal mechanism can be trained
    - A proposal generator: **objectness** (probability that a window covers an object)
        - Well-defined, closed boundary in space
        - Different appearance than its surroundings
        - Might be unique within the image (salient)
    - **Objectness cues**:
        - Density of salient pixels
        - Color contrast (window vs surrounding ring)
        - Superpixel straddling
    - Another proposal generator: **selective search**
        - Use hierarchical segmentation: start with small superpixels and merge based on diverse cues
        - **Book**: Like segmentation, we use the image structure to guide our sampling process. Like exhaustive search, we aim to capture all possible object locations.
    - **R-CNNs**:
        - Object proposals $\rightarrow$ warped image regions $\rightarrow$ forward each region through ConvNet $\rightarrow$ classify regions with SVMs
        - Pros: Accurate, any deep architecture can immediately be “plugged in”
        - Cons: Ad hoc training objectives, training is slow and takes a lot of disk space, inference (detection) is slow
        - **TowardsDataScience**: The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.
    - Fast R-CNN:
        - **TowardsDataScience**: Instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer.
        - Faster R-CNN: Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.

- Part-based models (Implicit Shape Models)
  - Represent objects by their parts, encode spatial structure (unlike bag-of-words)
  - Different Connectivity Structures: Constellation, Bag of Features, Star shape, etc.
  - **Pros**:
    - Works well for many different object categories (both rigid and articulated objects)
    - Flexible geometric model (can recombine parts seen on different training examples)
    - Learning from relatively few (50-100) training examples
    - Optimized for detection, good localization properties
  - **Cons**:
    - Needs supervised training data (object bounding boxes for detection, reference segmentations for top-down segmentations)
    - Only weak geometric constraints (result segmentations may contain superfluous body parts)
    - Purely representative model (no discriminative learning)

13 - Tracking

- **Tracking-by-detection**: detect object independently in each frame, associate detections over time into tracks
  - Properties
    - extreme case, often intermediate strategies are proposed
    - typically involves use of classifiers (generic over classes, or specific to an object)
    - **now the dominant modern paradigm**, many variants exist
    - more robust to sudden motions, less drift, even track over occlusions (given good appearance models)
    - often needs a separate training stage (more complex, need more data)
  - Temporal association by clustering
    - Clustering by clique partitioning: partition graph so as to maximize sum of intra-cluster weights
    - Each person in a different cluster
    - $\frac{A \cap B}{A \cup B} \in [0, 1]$: cue for same cluster
  - **Pros**:
    - Simple to understand, easy to build
    - Deals with tough imaging conditions, don't assume static background
  - **Cons**:
    - splits tracks when prolonged occlusions
    - only finds what detector trained on (no online adaptation)
    - holes in tracks due to missed detections (use only hard detections)
    - temporal association fragile on overlapping people (only cue is intersection area; no appearance)
    - needs whole shot (good for batch tracking)
  - A better version
    - Box overlap is a fragile affinity cue, add also appearance similarity (e.g. color histograms), point tracks passing by both boxes
    - Also can fill holes by interpolating within a track
    - Robust temporal association, even with overlapping people
- Online learning of appearance: **tracking by fast (re-) detection**

    - Properties
        - No training set fixed beforehand: **learn appearance of this specific object while it tracks it**
        - Requires an adaptive, robust and general tracker
        - Object detector: offline training (fixed training set, general object detector) -> object tracker: on-line boosting for feature selection (on-line update, specific object vs background)
        - When does it fail: drifting due to self-learning policy
    - Tracking by adaptive classifier: pro and contra (with respect to basic scheme)
        - no holes (search every frame)
        - only dependent on past frames -> good for online applications
        - can track anything (not only kind of objects detector trained on)
        - very robust, as explicitly models local background appearance
        - still cannot do long occlusions (actually drifts toward occluding object)
        - tracks one object at the time, independently
- Tracking-by-Detection with a Detector Confidence Particle Filter
- Including an articulated model
    - Learn relation between local feature and body parts, body parts related by kinematic constraints
- Tracking with stereo cameras
    - stereo camera -> estimate depth and ground-plane location
    - use it to reduce false-positives and reason in 3D
- **Keypoint-Based Object Detection**: Local features ensembles (Tracking by ensembles of local features)
    - Standard approach: keypoint detection -> patch rectification -> building descriptors -> descriptor matching
    - New approach: generate many samples of the keypoint's appearance under various perspective, lighting, noise... -> directly classify patches around keypoints in the test image
    - Pros:
        - fully re-initializing, as each frame is treated essentially independently
        - tracks over occlusions
        - only depend on past frames (good for online applications)
        - very fast (real-time on a modern PC)
    - Cons:
        - needs very textured objects
        - better with rigid objects (global geometry verification)
        - one target at the time
        - no adaptation: training set fixed beforehand
- **Temporal association based on local features**: entire objects and local features
    - Key idea: Track faces using viewpoint covariant regions, use these region tracks to resolve face temporal associations
    - Goal: develop very long and good quality tracks
        - Stage I: match regions detected in neighbouring frames
        - Stage II: repair tracks by region propagation
    - Connecting face detections temporally

        - Does not require contiguous detections (e.g. region tubes continue over pose changes)
        - Independent evidence -> no drift
    - Pros:
        - Tracks over missed detections and long occlusions
    - Cons:
        - Still only finds kind of objects the detector was trained on
        - Still holes (only hard detections)
        - Still depends on whole shot (no online)