1 Intro Lecture

Vision pipeline: Pre-processing (noise-reduction, image scaling, color space conversion, gamma correction), selecting areas of interest (object detection, bkgd subtraction, feature/keypt extraction, image segmentation), precise preocessing of selected regions (obj. recognition, tracking, feature matching), decision making (motion analysis, match/no match, flag events).
Why so difficult: Vision is an inverse problem. For inv. probs, often have to appeal to prob. models or assumptions. Many vision probs are ill-posed and have non-unique sol'ns.
Well-posed problem: (1) A sol'n exists, (2) sol'n is unique, (3) sol'n behav. changes cont. w init. conds.
Imaging function:
$I = \mathcal{P}(\text{Geometry, Materials, Viewpt, Lighting, Atmospherics, } \epsilon)$
$\mathcal{P}$ is perspective projection. $(G, M, V) = \mathcal{P}^{-1}(I)$ is what we want.
A key issue: Geometry is 3D (maybe 4D), appearance is 2D projection of this geometry (depends only weakly on geometry).
Human vision: human retina is 576 megapixels, 20-bit dynamic range, foviated vision (2°in centre of view is super high res, rest is meh but brain fills it in).
Morevec's paradox: hard problems (intelligence tests, games) are easy, easy problems (basic perception, mobility) are hard.

2 Mathematical Foundations

Homogeneous Coordinates: $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathbb{P}^2$ where $\mathbb{P}^2 = \mathbb{R}^3 \setminus (0,0,0)$ is a **projective space**. Points with $\tilde{w} = 0$ are **points at infinity**. $\bar{\boldsymbol{x}} = (x, y, 1)$ denotes normalized form or **augmented vector**. $\mathbb{P}^2$ can be thought of as the set of all lines in $\mathbb{R}^3$ that pass through the origin. It is topologically equivalent to the unit sphere as antipodal points are equivalent (points infinitely far away meet at the centre of an image). Lines in $\mathbb{P}^2$ are planes in $\mathbb{R}^3$ passing through the origin.
Lines in 2D: $\tilde{\boldsymbol{l}} = (a, b, c)$ represents $\tilde{\boldsymbol{x}} \cdot \tilde{\boldsymbol{l}} = ax + by + c = 0$. Normalize as $\boldsymbol{l} = (\hat{n}_x, \hat{n}_y, d) = (\hat{\boldsymbol{n}}, d)$ with normal and distance to origin. $\tilde{\boldsymbol{l}}_1, \tilde{\boldsymbol{l}}_2$ intersect at $\tilde{\boldsymbol{l}}_1 \times \tilde{\boldsymbol{l}}_2$.
Skew-Symmetric Form: $\boldsymbol{u} \times \boldsymbol{v} = [\boldsymbol{u}]_\times \boldsymbol{v}$ where $[\boldsymbol{u}]_\times = -[\boldsymbol{u}]_\times^T$

$$[\boldsymbol{u}]_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

Homogeneous Transformations: For given $\boldsymbol{C}, \boldsymbol{t}$:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \quad \boldsymbol{H}^{-1} = \begin{bmatrix} \boldsymbol{C}^T & -\boldsymbol{C}^T\boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

2.1 Rotation Representations

Rotation Matrices: Orthonormal matrices, $\boldsymbol{C}\boldsymbol{C}^T = \boldsymbol{C}^T\boldsymbol{C} = \boldsymbol{I}$ where $\boldsymbol{C} \in SO(2)$ or $SO(3)$. In 2D, $\boldsymbol{C} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$.
Euler Angles: Begin with aligned frames, perform each rotation about a different intermediate frame axis.

$$\boldsymbol{C}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \quad \boldsymbol{C}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\boldsymbol{C}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Yaw-pitch-roll set of Euler angles: $\boldsymbol{C}(\psi, \theta, \phi) = \boldsymbol{C}(\psi)\boldsymbol{C}(\theta)\boldsymbol{C}(\phi) =$

$$\begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

Axis-Angle: Rotation by θ around $\hat{\boldsymbol{n}}$. Use **Rodriguez's formula** to convert to matrix:

$$\boldsymbol{C}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I}_3 + \sin\theta\,[\hat{\boldsymbol{n}}]_\times + (1 - \cos\theta)\,[\hat{\boldsymbol{n}}]_\times^2$$

Quaternion: $\boldsymbol{q} = q_0 + \mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \in \mathbb{H}$ are normalized to be unit length on $\mathbb{S}^3$, which forms a **double cover** of $SO(3)$. $\boldsymbol{q}$ is the same rotation as $-\boldsymbol{q}$. $\bar{\boldsymbol{q}}$ (conjugate) is the opposite rotation. Multiply as

$$\boldsymbol{p} \otimes \boldsymbol{q} = (p_0 + \mathbf{p}) \otimes (q_0 + \mathbf{q}) = p_0 q_0 - \mathbf{p}^T\mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}$$

Axis-angle: $\boldsymbol{q} = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \hat{\boldsymbol{n}}\sin(\theta/2) \end{bmatrix}$.
Rotation matrix: $\boldsymbol{C}(\boldsymbol{q}) = \boldsymbol{I}_3 + 2q_0\,[\mathbf{q}]_\times + 2\,[\mathbf{q}]_\times^2 =$

$$\begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\ 2q_0q_3 + 2q_1q_2 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_0q_1 + 2q_2q_3 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix}$$

2.2 Linear Transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	□
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	◇
similarity	$\begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	◇
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	▱
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	▱

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \end{bmatrix}_{3\times4}$	3	orientation	□
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}_{3\times4}$	6	lengths	◇
similarity	$\begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \end{bmatrix}_{3\times4}$	7	angles	◇
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{3\times4}$	12	parallelism	▱
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{4\times4}$	15	straight lines	▱

Examples: Camera intrinsic matrix is 2D affine; extrinsic matrix is 3D Euclidean. Perspective projection is 3D projective. Composition of projections is projective.

3 Probability and Estimation

3.1 Probability Definitions

Cumulative Distribution Function: $F_x(x) = \sum_{\bar{x}=-\infty}^{x} f_x(x)$
Marginalization and **Conditioning**:

$$f_x(x) = \sum_{y\in\mathcal{Y}} f_{xy}(x,y) \qquad f_{x|y}(x|y) = \frac{f_{xy}(x,y)}{f_y(y)}$$

For conditional probabilities:

$$f_{x|z}(x|z) = \sum_{y\in\mathcal{Y}} f_{xy|z}(x,y|z) \qquad f_{x|y,z}(x|y,z) = \frac{f_{xy|z}(x,y|z)}{f_{y|z}(y|z)}$$

Law of Total Probability: $f_x(x) = \sum_{y\in\mathcal{Y}} f_{x|y}(x|y) f_y(y)$
Bayes' Rule: $\underbrace{f_{x|y}(x|y)}_{\text{posterior}} = \overbrace{\underbrace{f_{y|x}(y|x)}_{\text{likelihood}}\underbrace{f_x(x)}_{\text{prior}}}^{\text{}} \Big/ \underbrace{f_y(y)}_{\text{evidence}}$
x and y are **independent** if $f(x|y) = f(x) \iff f(x,y) = f(x)f(y)$.
x and y are **conditionally independent** given z if

$$f(x|y,z) = f(x|z) \iff f(x,y|z) = f(x|z)f(y|z)$$

Expectation: $\boldsymbol{\mu}_x = \mathbb{E}[\boldsymbol{x}] = \sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x} f_x(\boldsymbol{x})$
Variance and Covariance:
$\boldsymbol{\Sigma}_x = \mathrm{Var}[\boldsymbol{x}] = \mathbb{E}[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^T] = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{x}]^T$
$\mathrm{Var}[x] = \sigma_x^2 = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$ in 1D. $\boldsymbol{\Sigma}$ is symmetric positive (semi-)definite.
Law of the Unconscious Statistician:

$$\mathcal{Y} = \{\, y \mid y = g(x), x \in \mathcal{X} \,\} \implies \mathbb{E}[y] = \mathbb{E}[g(x)]$$

Gaussian Distribution:

$$x \sim \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = 2\pi^{-\frac{n}{2}} \det\boldsymbol{\Sigma}^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

3.2 Linear Least Squares

Fit a model $y_i = f(x_i; \boldsymbol{\theta})$ where $\boldsymbol{\theta} = (m, b)$ is the vector of parameters we wish to determine. Matrix form: $y_i = \boldsymbol{J}(x_i)\boldsymbol{\theta}$ where $\boldsymbol{J}(x) = \frac{\partial f(x;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is the model **Jacobian** with respect to the parameters.
Minimize $E_{LS} = \sum_i \|\tilde{y}_i - f(x_i; \boldsymbol{\theta})\|^2$. Solution given by **normal equation**:

$$\hat{\boldsymbol{\theta}} = \left(\sum_i \boldsymbol{J}^T(x_i)\boldsymbol{J}(x_i)\right)^{-1} \left(\sum_i \boldsymbol{J}^T(x_i)\tilde{y}_i\right)$$

With varying uncertainties, $E_{WLS} = \sum_i \sigma_i^{-2}\|r_i\|^2$

3.3 Nonlinear Least Squares (Gauss-Newton)

We want to optimize $E(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{e}(\boldsymbol{x})^T\boldsymbol{e}(\boldsymbol{x})$ for general nonlinear $\boldsymbol{e}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}$. At each step linearize the system as $\boldsymbol{e}(\boldsymbol{x}) \approx \boldsymbol{e}(\boldsymbol{x}_{op}) + \boldsymbol{J}_e\delta\boldsymbol{x}$ where $\boldsymbol{J}_e = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\big|_{\boldsymbol{x}_{op}}$.
Solve for optimal update $\delta\boldsymbol{x}^* = -(\boldsymbol{J}_e^T\boldsymbol{J}_e)^{-1}\boldsymbol{J}_e^T\boldsymbol{e}(\boldsymbol{x}_{op})$ and update operating point $\boldsymbol{x}_{op} \leftarrow \boldsymbol{x}_{op} + \delta\boldsymbol{x}^*$.
For multiple measurements and weighted errors,

$$\delta\boldsymbol{x}^* = -\left(\sum_{i=1}^N \boldsymbol{J}_{e_i}^T\boldsymbol{W}_i\boldsymbol{J}_{e_i}\right)^{-1} \left(\sum_{i=1}^N \boldsymbol{J}_{e_i}^T\boldsymbol{W}_i\boldsymbol{e}_i(\boldsymbol{x}_{op})\right)$$

Weights can be inverses of covariance matrices for each measurement.

3.4 Random Sample Consensus (RANSAC)

1. Determine the smallest number of data points required to fit the model.
2. Draw the smallest possible subset to fit the model.
3. Check the number of points in the whole dataset that are within some threshold of the model prediction (inliers).
4. If we have enough points within the threshold, terminate and re-fit to the inliers.
5. Repeat from step 2 until we reached the max number of iterations.

With probability w of drawing an inlier and n points to fit the model, probability of drawing only inliers is $b = w^n$. Expected number of trials to get an outlier-free draw is $1/b = w^{-n}$.
To get with probability z, at least one outlier-free draw:

$$z = 1 - (1 - w^n)^k \qquad k = \frac{\log(1-z)}{\log(1-b)}$$

4 Image Formation and Optics

4.1 Pinhole Camera Model

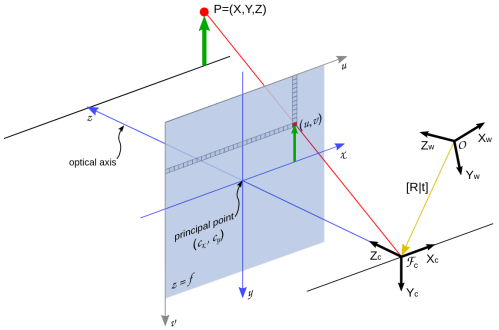
Basic Projective Map: Normalized by focal length,

$$\bar{\boldsymbol{x}} = \mathcal{P}_z(\boldsymbol{p}) = \boldsymbol{p}/p_z = \begin{bmatrix} x/z & y/z & 1 \end{bmatrix}^T$$

Intrinsic Matrix: For focal lengths f_x, f_y, center c_x, c_y (pixels), skew s:

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} = \boldsymbol{K}\frac{\boldsymbol{p}}{z}$$

This projects 3D to pixel coords. Note one can also normalize after projection. One can obtain a ray of points that project to the same pixel by multiplying the augmented pixel vector by $\boldsymbol{K}^{-1}$.



FoV, Sensor Width, Focal Length: $\tan\frac{\theta}{2} = \frac{W}{2f}$
Extrinsic Matrix: $\boldsymbol{E}_{CW} \in SE(3)$ contains the world frame to camera frame transform. $\boldsymbol{E}_{CW} = \boldsymbol{E}_{WC}^{-1}$ where $\boldsymbol{E}_{WC}$ is the camera pose in world frame.
Projective Matrix: $\tilde{\boldsymbol{P}} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{C} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \tilde{\boldsymbol{K}}\boldsymbol{E}$. Project a world point to image plane as $\tilde{\boldsymbol{x}}_s = \tilde{\boldsymbol{P}}\tilde{\boldsymbol{p}}_w$, then normalize by the third element.

4.2 Distortion & Optical Effects

Plumb Bob Distortion Model: From normalized image coordinates to distorted coordinates, where $r = \sqrt{x_n^2 + y_n^2}$ and x_d, y_d normalized by f_x, f_y:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} 2\tau_1 x_n y_n + \tau_2(r^2 + 2x_n^2) \\ 2\tau_2 x_n y_n + \tau_1(r^2 + 2y_n^2) \end{bmatrix}$$

Note distortion is applied after extrinsics & projective map, but before doing intrinsics.
Radial Distortion: Barrel (corner points pushed in), pincushion (corner points pulled out), or mustache (combination).
Tangential Distortion: Pixels shifted in one direction (further pixels shifted more), due to the lens and imaging plane not being parallel.
Unwarping: Undoing distortion by precomputing distorted location of each pixel, then using bilinear interpolation.
Chromatic Aberration: Differences in refraction due to wavelength cause slightly different focuses & magnifications (i.e. colours are misaligned).
Vignetting: Brightness decreases towards the edge of the image, caused by lens foreshortening, i.e. for rays at higher angles, the size of the lens is effectively smaller, so less light comes in.
Aliasing: Artifacts (e.g. Moire patterns) caused by spatial sampling.
Bayer Mosaics: Imaging sensors have more green pixels to look natural.

5 Image Operations & Transformations

Bias/Gain (Contrast/Brightness): $g(i,j) = af(i,j) + b$
Gamma Correction: $g(i,j) = f(i,j)^{\frac{1}{\gamma}}$
Histogram Equalization: $T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$ where $T(r_k)$ is the CDF and $p_r(r_j)$ is the PDF (normalized histogram) as a function of pixel intensity. Each pixel value is mapped to the proportion of pixels that it's brighter than, resulting in a flat-ish histogram.
Linear Filtering: $g(i,j) = f * h = \sum_{k,l} f(i-k, j-l)h(k,l)$ where $h(k,l)$ is the **kernel**. Note how the kernel is flipped. Separable kernels can be expressed as $\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T$ and is equivalent to convolution by $\boldsymbol{v}$ then $\boldsymbol{h}$.

Isotropic Gaussian Kernel: Separable kernel given by

$$G(x,y;\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Laplacian of Gaussian:

$$\nabla^2 G(x,y;\sigma) = \left(\frac{x^2+y^2+2\sigma^2}{\sigma^4}\right) G(x,y;\sigma)$$

A band-pass filter, can be used for edge/blob detection. Kernel shape like an upside down sombrero (ring of positive values, high negative value in the centre.)

Integral Image: Sum of pixels up to (i,j), can be computed in linear time and used to compute box filters in linear time.

$$\begin{aligned} s(i,j) &= \sum_{k=0}^{i}\sum_{l=0}^{j} f(k,l) \\ &= s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j) \end{aligned}$$

Nonlinear Filters: Simplest examples include min/max (erosion/dilation) or median filter (removes shot noise).

Bilateral Filter: Soft-rejects pixels whose value differs too much from the centre; removes noise while preserving edges, useful for image upscaling.

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$\begin{aligned} w(i,j,k,l) &= d(i,j,k,l) r(i,j,k,l) \\ &= \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}\right) \exp\left(-\frac{\|f(i,j)-f(k,l)\|^2}{2\sigma_r^2}\right) \end{aligned}$$

Geometric Transform: Applies to domain, $g(\boldsymbol{x}) = f(\boldsymbol{h}(\boldsymbol{x}))$. Better to compute the inverse transformation and map each pixel in the output to a location in the input, and bilinearly interpolate.

Regularization: For ill-conditioned problems. Combine smoothness penalty (penalize large gradients) with data penalty.

$$\varepsilon = \iint (f(x,y) - d(x,y))^2 \,\mathrm{d}x\,\mathrm{d}y + \lambda \iint f_x^2(x,y) + f_y^2(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

6 Image Features

6.1 Feature Identification and Description

Features should be salient (distinctive), local, repeatable (can be found in other images) and compact.

Weighted Summed Squared Difference: Images I_0, I_1, displacement $\boldsymbol{u}$

$$E_{WSSD}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)(I_1(\boldsymbol{x}_i+\boldsymbol{u}) - I_0(\boldsymbol{x}_i))^2$$

For $I_0 = I_1$ this is **autocorrelation** for a small $\Delta\boldsymbol{u}$; high autocorrelation means the area is distinctive. $E_{AC}(\Delta\boldsymbol{u}) \approx \Delta\boldsymbol{u}^T \boldsymbol{A} \Delta\boldsymbol{u}$. As convolution, $\boldsymbol{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$.

Harris Corner Detector: $\det \boldsymbol{A} - \alpha \operatorname{tr} \boldsymbol{A}^2 = \lambda_0\lambda_1 - \alpha(\lambda_0+\lambda_1)^2$. Largest when both eigenvalues are large, indicating distinctiveness in both directions, i.e. a corner.

SIFT: Scale and rotation invariant, capable but slow. Construct scale space representation by repeated blurring and downsampling. Find keypoints with Difference of Gaussians (similar to LoG). Reject keypoints using Hessians similar to Harris. Assign each keypoint an orientation using gradients of points around the keypoint in a histogram. Assemble final descriptor using 4x4 grid of subpatches, each subpatch having a gradient orientation histogram.

SURF: Like SIFT but faster. Uses box filters instead of Gaussians (can use integral images). Smaller descriptor size (half as SIFT, 64-dimensional vector).

FAST: Very fast but not scale or rotation invariant. For each pixel look at the 16 pixels in a ring around it, if there are 12 contiguous pixels that are brighter or darker than the centre it is a feature. Use non-maximum suppression (suppress non-maximum features in an area around each maximum).

BRISK: Sample pairs of points in a circular pattern around the centre (Gaussian applied around each sampled point); compare the pairs and generate one bit in the feature for each pair.

BRIEF: Draw points randomly (one of several distributions) around the centre, compare them similarly to BRISK to form descriptor. Sampling pattern is decided once and used consistently across images. 512-bit features. Fast but not scale/rotation invariant and sensitive to noise.

ORB: Combines FAST and BRIEF to get a fast and reliable feature. Downsample similarly to SIFT and apply FAST at all scales to find keypoints. Compute orientation using intensity centroid. Use rotated BRIEF (rotate sampling pattern or image subpatch) to form descriptor.

Learning-Based Detectors: Often use classic features to generate training data for supervised methods, or image + pose sequences for unsupervised methods. LIFT is one such method using 3 CNNs to detect, estimate orientation, and generate descriptor, trained using contrastive learning.

6.2 Feature Matching

For features without descriptor, use **sum-of-squared-differences** in a local patch as distance function. Vector-based descriptors can use **Euclidean distance**. Binary descriptors can use **Hamming distance** (number of differing bits). Use RANSAC to reject outliers.

6.3 Quantifying Matching Performance

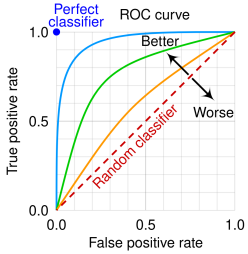
Confusion Matrix: To quantify matching performance. Table with true positives, false positives, false negatives, true negatives.

Recall/True Positive Rate: $TPR = TP/(TP+FN)$

False Positive Rate: $FPR = FP/(FP+TN)$

Precision/Positive Predictive Value: $PPV = TP/(TP+FP)$

Accuracy: $ACC = (TP+TN)/(TP+FN+TN+FP)$



Receiver Operating Characteristic: Plotting TP against FP for different values of thresholds.

Average Precision (AP): Area under the ROC curve.

k-D Tree: Multi-dimensional binary tree allowing for quick nearest neighbour queries. Built in $O(kn\log n)$ with $O(n)$ space and nearest neighbours in $O(\log n)$ time. When building, split along the median of the current dimension at each level and cycle through dimensions until there is at most 1 data point in each partition. For nearest neighbours:

1. Find the node in the same partition as the query point by searching down the tree, and set the current min distance.
2. Go back up the tree and check the following, until the root node:
   (a) Check the node against the current best and update if necessary.
   (b) Check if the other branch can possibly contain a point closer to the query point than the current best, using the distance between the query point and the partitioning point, along the partitioning axis; if the current best is less than this, eliminate the subtree, otherwise recurse down the subtree.

Kanade-Lucas-Tomasi (KLT) Tracker: Predict camera motion to narrow search space and match within a small area using gradient information. Feature selection is similar to Harris. In every frame, add new features, and discard features that have grown too dissimilar to when they were first added (based on RMS residual).

7 Camera and Object Pose Estimation

Perspective-n-Point Problem: Given 3D points and where they project to in 2D, estimate intrinsic and extrinsic parameters.

Direct Linear Transform (DLT): Directly estimating entries of the projection matrix $\boldsymbol{P}$; each point gives 2 equations

$$\begin{aligned} x_i &= \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}} \\ y_i &= \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}} \end{aligned}$$

Then recover $\boldsymbol{K}$ and $\boldsymbol{E}$ by QR factorization into upper-triangular $\boldsymbol{K} = \boldsymbol{R}$ and orthogonal $\boldsymbol{Q}$. This is an approximate solution since often the resulting matrices are not valid (particularly rotation) due to lack of constraints.

Iterative Solution: Minimize projection error $\boldsymbol{e}_i = \boldsymbol{x}_i - f(\boldsymbol{p}_i; \boldsymbol{C}, \boldsymbol{t}, \boldsymbol{K})$ using nonlinear least squares (special attention to $\boldsymbol{C}$ constraints).

Wahba Problem: Find $\boldsymbol{C}$ between two frames given $\boldsymbol{u}_i, \boldsymbol{v}_i$ in the frames. $\boldsymbol{e} = \boldsymbol{C}\boldsymbol{u}_i - \boldsymbol{v}_i$.

Using Euler angles, $\boldsymbol{J}_{e_i} = \frac{\partial \boldsymbol{C}\boldsymbol{u}_i}{\partial \boldsymbol{\theta}}$. Each column:

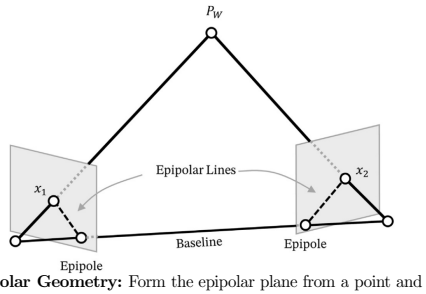
$$\begin{aligned} \frac{\partial \boldsymbol{C}\boldsymbol{u}_i}{\partial\theta_3} &= [\mathbf{1}_3]_\times \boldsymbol{C}_3(\theta_3)\boldsymbol{C}_2(\theta_2)\boldsymbol{C}_1(\theta_1)\boldsymbol{u}_i \\ \frac{\partial \boldsymbol{C}\boldsymbol{u}_i}{\partial\theta_2} &= \boldsymbol{C}_3(\theta_3)[\mathbf{1}_2]_\times \boldsymbol{C}_2(\theta_2)\boldsymbol{C}_1(\theta_1)\boldsymbol{u}_i \\ \frac{\partial \boldsymbol{C}\boldsymbol{u}_i}{\partial\theta_1} &= \boldsymbol{C}_3(\theta_3)\boldsymbol{C}_2(\theta_2)[\mathbf{1}_1]_\times \boldsymbol{C}_1(\theta_1)\boldsymbol{u}_i \end{aligned}$$

This breaks down at gimbal lock since Jacobian becomes zero.

Using axis-angle/matrix, $\boldsymbol{J}_{e_i} = -[\boldsymbol{C}_{op}\boldsymbol{u}_i]_\times$. For small rotations $\boldsymbol{C}(\delta\boldsymbol{\phi}) \approx 1 + [\delta\boldsymbol{\phi}]_\times$. Update operating point as $\boldsymbol{C}_{op} \leftarrow \boldsymbol{C}(\delta\boldsymbol{\phi}^*)\boldsymbol{C}_{op}$.

8 Stereo Vision

8.1 Stereo Geometry



Epipolar Geometry: Form the epipolar plane from a point and the cameras' optical centres; the intersection of the epipolar plane with each imaging plane forms an epipolar line, which is where correspondences lie.

Stereo Rectification: Transforming and unwarping the images so the imaging planes become parallel, at the same depth, and rotationally aligned (fronto-parallel or standard rectified geometry). After rectification all epipolar lines are horizontal.

Disparity: Difference in x values of a point between cameras. $Z = \frac{fb}{d}$ where b is baseline and d is disparity. Larger disparity indicates a point is closer.

Stereo Camera Model: $\begin{bmatrix} x_0 \\ y_0 \\ d \end{bmatrix} = \boldsymbol{f}(\boldsymbol{p}_c) = \frac{f}{Z}\begin{bmatrix} X \\ Y \\ b \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \\ 0 \end{bmatrix}$

Baseline: Distance between cameras. Larger baseline improves accuracy, but makes matching more difficult as overlapping FOV decreases and foreshortening is increased (oblique objects appear as different sizes). Wider baseline also makes errors more Gaussian-like (shorter tail); for short baseline, using Gaussian errors biases the points to be closer.

8.2 Stereo Matching

Local Methods: Sliding a small window along the epipolar line, without using global info. Computes similarity within window, e.g. sum-of-squared/absolute-differences, or normalized cross-correlation or gradient based. Larger window sizes result in smoother and less noisy images but loses detail.

Learning-Based Methods: Often using CNNs to combine local and global aggregation. Produces smoother depth maps. Often trained on synthetic datasets.

8.3 Invertible Stereo Models

Combined Stereo Model: Forward model produces pixel locations in both cameras. Frame origin is halfway between two cameras.

Forward: $f(\boldsymbol{p}) = \boldsymbol{M}\frac{1}{z}\boldsymbol{p} = \begin{bmatrix} f_u & 0 & c_u & f_u\frac{b}{2} \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u\frac{b}{2} \\ 0 & f_v & c_v & 0 \end{bmatrix} \frac{1}{z}\boldsymbol{p} = \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix}$.

Inverse model: $\boldsymbol{g}(\boldsymbol{y}) = \frac{b}{u_l - u_r}\begin{bmatrix} \frac{f_u}{f_v}\left(\frac{1}{2}(u_l+u_r) - c_u\right) \\ \frac{f_u}{f_v}\left(\frac{1}{2}(v_l+v_r) - c_v\right) \\ f_u \end{bmatrix}$.

Jacobians: forward $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \boldsymbol{M}\frac{1}{p_3}\begin{bmatrix} 1 & 0 & -p_1/p_3 & 0 \\ 0 & 1 & -p_2/p_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -p_4/p_3 & 1 \end{bmatrix}$, inverse $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} = \frac{b}{(u_l-u_r)^2}\begin{bmatrix} -u_r + c_u & 0 & u_l - c_u & 0 \\ -\frac{f_u}{f_v}\left(\frac{1}{2}(v_l+v_r) - c_v\right) & \frac{f_u}{2f_v} & \frac{f_u}{f_v}\left(\frac{1}{2}(v_l+v_r) - c_v\right) & \frac{f_u}{2f_v} \\ -f_u & 0 & f_u & 0 \end{bmatrix}$

Pixel-Disparity Stereo Model: Forward model produces a pixel location and disparity. Frame origin is at the left camera.

Forward: $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{p}) = \boldsymbol{M}\frac{1}{p_3}\boldsymbol{p} = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & f_u b \end{bmatrix} \frac{1}{p_3}\boldsymbol{p} = \begin{bmatrix} u_l \\ v_l \\ d \end{bmatrix}$.

Inverse model: $\boldsymbol{g}(\boldsymbol{y}) = \frac{b}{d}\begin{bmatrix} u_l - c_u \\ \frac{f_u}{f_v}(v_l - c_v) \\ f_u \end{bmatrix}$.

Forward Jacobian: $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \frac{1}{p_3}\boldsymbol{M}\begin{bmatrix} 1 & 0 & -p_1/p_3 & 0 \\ 0 & 1 & -p_2/p_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -p_4/p_3 & 1 \end{bmatrix}$

Inverse Jacobian: $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} = \frac{b}{d^2}\begin{bmatrix} d & 0 & c_u - u_l \\ 0 & \frac{f_u}{f_v}d & \frac{f_u}{f_v}(c_v - v_l) \\ 0 & 0 & -f_u \end{bmatrix}$

9 Camera Calibration

Calibration: Characterizing camera properties, including geometric (determining distortion and $\boldsymbol{K}$; most common) or photometric (determining optical effects and getting consistent brightnesses; harder and less common).

For geometric, given image coordinates $\boldsymbol{x}_{ij}$ for point i in observed from pose j and target points $\boldsymbol{p}_i$, recover all parameters in $\boldsymbol{x}_{ij} = f(\boldsymbol{p}_i; \boldsymbol{C}_j, \boldsymbol{t}_j, \boldsymbol{K}, \boldsymbol{\kappa}, \boldsymbol{\tau})$.

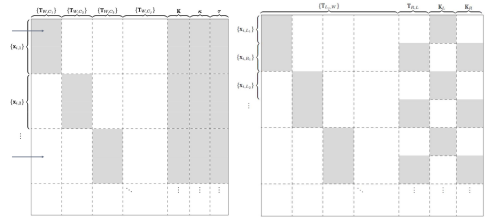
Calibration Targets: Most commonly checkerboard (N-planes approach) due to ease of detection, precision of corners, and invariance to foreshortening.

Solution: Use nonlinear least squares (DLT has no distortion and fails for coplanar points). Parameter θ contains calibration parameters and all poses. Minimize reprojection error:

$$E(\delta\boldsymbol{\theta}) = \sum_{ij}\left\| \frac{\partial f}{\partial \boldsymbol{C}_j}\delta\boldsymbol{C}_j + \frac{\partial f}{\partial \boldsymbol{t}_j}\delta\boldsymbol{t}_j + \frac{\partial f}{\partial \boldsymbol{K}}\delta\boldsymbol{K} + \frac{\partial f}{\partial \boldsymbol{\kappa}}\delta\boldsymbol{\kappa} + \frac{\partial f}{\partial \boldsymbol{\tau}}\delta\boldsymbol{\tau} - \boldsymbol{r}_{ij} \right\|^2$$

Stereo Calibration: Additionally solve for $\boldsymbol{T}_{R,L}$, the rigid transform from left camera to right camera. $\tilde{\boldsymbol{P}}_L = \tilde{\boldsymbol{K}}\boldsymbol{T}_{L,W}$, $\tilde{\boldsymbol{P}}_R = \tilde{\boldsymbol{K}}\boldsymbol{T}_{R,L}\boldsymbol{T}_{L,W}$. Parameter vector has both camera calibration parameters, left camera pose for each image, and transform between cameras.

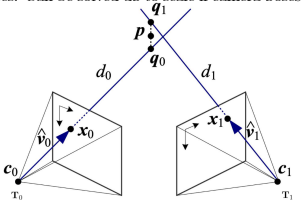
Jacobian Structure: Sparse, only the extrinsics for pose j affects $\boldsymbol{x}_{ij}$, but $\boldsymbol{K}, \boldsymbol{\kappa}, \boldsymbol{\tau}$ affects all measurements made with that camera. Can be exploited using sparse Cholesky factorization.



10 Sparse SfM, Visual Odometry, and SLAM

10.1 Structure from Motion

Structure from Motion: Determining 3D positions of landmark points (structure) and camera poses (motion) from image feature correspondences. Can be solved up to scale if camera poses are unknown.



Triangulation: Given $\boldsymbol{x}_j, \boldsymbol{K}_j$ and pose $\boldsymbol{C}_j, \boldsymbol{c}_j$ for each camera, find the point $\boldsymbol{p}$ closest to all of the rays from each camera. Each ray has unnormalized direction $\boldsymbol{v}_j = \boldsymbol{C}_j^{-1}\boldsymbol{K}_j^{-1}\tilde{\boldsymbol{x}}_j$. Optimal distance $d_j = \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{p} - \boldsymbol{c}_j)$ minimizes error $\|\boldsymbol{c}_j + d_j\hat{\boldsymbol{v}}_j - \boldsymbol{p}\|^2$, results in optimal point $\boldsymbol{q}_j = \boldsymbol{c}_j + (\hat{\boldsymbol{v}}_j\hat{\boldsymbol{v}}_j^T)(\boldsymbol{p} - \boldsymbol{c}_j)$. Least squares to minimize distance between $\boldsymbol{p}$ and each $\boldsymbol{q}_j$.

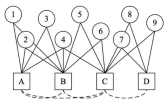
Bundle Adjustment: Using NLS to jointly optimize the 3D structure (through triangulation) and camera poses (and possibly calibration).

Optimizations: Naive NLS is cubic in number of parameters. Exploit sparse Jacobian and Hessian ($\boldsymbol{J}^T\boldsymbol{J}$) structure. Order parameters by grouping relevant parameters and putting global parameters last. Use sliding windows to optimize a fixed horizon or select keyframes to optimize fewer frames for very large problems (SLAM). Use Kalman filters for incremental solutions.

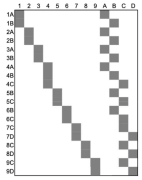
Parametrizations: Should be close to linear so NLS is locally quadratic. Use homogeneous coordinates $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w})$ for landmark points to handle points at infinity. For rotations use quaternions or matrices.

Jacobian/Hessian Structure: Jacobian has a row for each observation (each feature in each camera), a column for each unique feature's 3D location, and one for each camera's parameters. Dense block between observation and feature when the feature appears in that observation, between observation and camera when the observation comes from that camera. Hessians are somewhat diagonal since 3D feature locations don't affect each other.

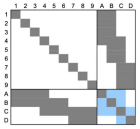
Robust Estimators: Solution is sensitive to outliers, so in addition to RANSAC, cost of outliers should be downweighted. Instead of L2, use M-estimators or cost functions that become linear or flatten out for large errors to weigh outliers less. Might reduce region of convergence.



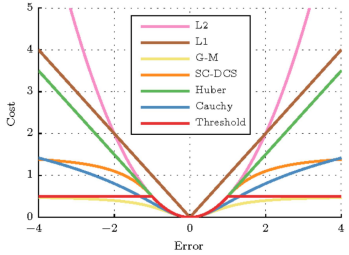
Network Structure



Jacobian Matrix



Hessian Matrix



10.2 Incremental SfM/Visual Odometry

Visual Odometry: Given images captured in frame a and frame b, find $\boldsymbol{T}_{ba}$. Use where high accuracy short-term localization is needed and no external options, e.g. indoors, Mars, or high wheel slip environments.

Taxonomy: *Monocular* (non-invertible observation model) vs. *stereo* (invertible); *feature-based/sparse* (extract & match features) vs. *intensity-based* (directly match image intensities).

VO Pipeline: Stereo dewarp and rectify, detect features and stereo match, match features with previous frame, outlier rejection, finally solve for pose transform with NLS, potentially with other sensors.

Point Cloud Alignment: Given $\boldsymbol{p}_a^j, \boldsymbol{p}_b^j$, minimize cost

$$E(\boldsymbol{C}_{ba}, \boldsymbol{r}_a) = \frac{1}{2}\sum_{j=1}^{J}\left(\boldsymbol{p}_b^j - \boldsymbol{C}_{ba}(\boldsymbol{p}_a^j - \boldsymbol{r}_a)\right)^T \boldsymbol{\Sigma}^j \left(\boldsymbol{p}_b^j - \boldsymbol{C}_{ba}(\boldsymbol{p}_a^j - \boldsymbol{r}_a)\right)$$

with matrix weights $\boldsymbol{\Sigma}^j = \left(\boldsymbol{G}_b^j\boldsymbol{R}_b^j\boldsymbol{G}_b^{jT} + \boldsymbol{C}_{ba}\boldsymbol{G}_a^j\boldsymbol{R}_a^j\boldsymbol{G}_a^{jT}\boldsymbol{C}_{ba}^T\right)^{-1}$ where $\boldsymbol{G}_b^j = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{f}(\boldsymbol{p}_b^j)}, \boldsymbol{G}_a^j = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{f}(\boldsymbol{p}_a^j)}$, and $\boldsymbol{R}_b^j, \boldsymbol{R}_a^j$ are pixel measurement covariances. Closed-form solution for scalar weights:

1. Align centroids: $\boldsymbol{p}_a = \frac{\sum_{j=1}^J w^j \boldsymbol{p}_a^j}{\sum_{j=1}^J w^j}, \boldsymbol{p}_b = \frac{\sum_{j=1}^J w^j \boldsymbol{p}_b^j}{\sum_{j=1}^J w^j}$
2. Outer product: $\boldsymbol{W}_{ba} = \frac{1}{\sum_{j-1}^J w^j}\sum_{j=1}^{J} w^j(\boldsymbol{p}_b^j - \boldsymbol{p}_b)(\boldsymbol{p}_a^j - \boldsymbol{p}_a)^T$
3. Compute SVD $\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T = \boldsymbol{W}_{ba}$, then compute rotation $\boldsymbol{C}_{ba} = \boldsymbol{V}\begin{bmatrix}1&0&0\\0&1&0\\0&0&\det(\boldsymbol{U})\det(\boldsymbol{V})\end{bmatrix}\boldsymbol{U}^T$, and translation $\boldsymbol{r}_a = -\boldsymbol{C}_{ba}^T\boldsymbol{p}_b + \boldsymbol{p}_a$.

11 Optical Flow

Optical Flow: Estimating a motion vector for every pixel in the image from a series of images, resulting in an optical flow field. Ill-posed problem due to the aperture problem.

Brightness Constancy Assumption:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

i.e. the corresponding pixel in the next image has the same brightness, where $(\Delta x, \Delta y)$ is the flow vector we want.

Flow Equation:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0 \iff I_x v_x + I_y v_y = \Delta I^T \boldsymbol{v} = -I_t$$

obtained from Taylor expansion of brightness constancy.

Lucas-Kanade Method: Assuming each small patch has the same flow and uses only local information. Matrix equation for each patch:

$$\begin{bmatrix} I_x(\boldsymbol{p}_1) & I_y(\boldsymbol{p}_1) \\ \vdots & \vdots \\ I_x(\boldsymbol{p}_N) & I_y(\boldsymbol{p}_N) \end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix} = -\begin{bmatrix} I_t(\boldsymbol{p}_1) \\ \vdots \\ I_t(\boldsymbol{p}_N)\end{bmatrix} \iff \boldsymbol{A}\boldsymbol{d} = \boldsymbol{b}$$

$$\begin{bmatrix}\sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix} = -\begin{bmatrix}\sum I_x I_t \\ \sum I_y I_t\end{bmatrix} \iff (\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{d} = \boldsymbol{A}^T\boldsymbol{b}$$

Horn-Schunck Method: Uses a smoothness regularization term, so large changes in flow vector between adjacent pixels is penalized:

$$E = \iint [(I_x u + I_y v + I_t)^2 + \alpha^2(\|\Delta u\|^2 + \|\Delta v\|^2)]\,\mathrm{d}x\,\mathrm{d}y$$

where α is the regularization constant and $\Delta u, \Delta v$ are flow vector gradients. Practically we take the derivative and convert to summation, then use NLS to solve. Can help fill in textureless regions, at the cost of blurring boundaries.

Control Applications: Objects further away move slower, so we can get relative depth info. A simple navigation rule is to steer so that optical flows of 2 cameras on two sides are equal, to centre the robot between two walls.

12 Dense/Photometric Mapping

Dense Mapping: Match images and produce a pose between frames based on image intensities only, without extracting features, and using (almost) every pixel to construct the map, i.e. solving pixel correspondence and camera pose estimation at once, instead of decoupling them like in sparse VO.

Comparison to Sparse Mapping/VO:

1. More robust to common failure modes due to inclusion of global information, e.g. blur, self-similar textures, defocus
2. Uses almost all information in the image (as opposed to only pixels around features)
3. Dense maps enable scene understanding and path planning
4. Implicit data association by matching pixel intensities
5. Very high dimensionality (10^5 to 10^6 pixels per image), computationally expensive
6. Contains a lot more local minima (due to implicit data association), sensitive to initialization (works only for small camera movements)

Dense Map Representations: Surface-based: meshes, splines (fitting splines to points), height/depth maps; Point-based: point clouds, surfels (like voxels but for a small 2D surface patch), Gaussian splats (blob with centre, covariance and colour); Volume-based: voxel grids (e.g. octrees for adaptive resolution), signed distance functions.

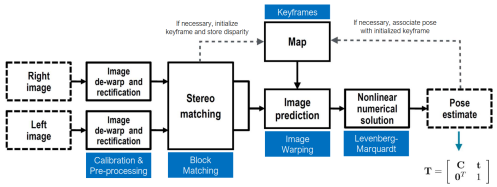


Image Warping: Given the location of a pixel in frame $k-1$ and pose transformation $\boldsymbol{T}_{k,k-1}$, $\begin{bmatrix}u_k\\v_k\\d_k\end{bmatrix} = \boldsymbol{f}\left(\boldsymbol{T}_{k,k-1}\boldsymbol{g}\left(\begin{bmatrix}u_{k-1}\\v_{k-1}\\d_{k-1}\end{bmatrix}\right)\right)$.

Photometric Error: $e_i = I_{k-1}(\boldsymbol{u}_{k-1}^i) - I_k'(\boldsymbol{u}_k^i)$ where I_k' is interpolated from I_k, indexed with continuous coordinates $\boldsymbol{u}_k^i$, and $\boldsymbol{y}_k^i = \begin{bmatrix}\boldsymbol{u}_k^i\\d_k^i\end{bmatrix} = \boldsymbol{f}(\boldsymbol{T}_{k,k-1}\boldsymbol{g}(\boldsymbol{y}_{k-1}^i))$ from image warping. Solve for optimal pose transform $\hat{\boldsymbol{T}}_{k,k-1} = \underset{\boldsymbol{T}_{k,k-1}}{\arg\min}\sum_{i=1}^{N}\left(\frac{1}{\sigma}e_i\right)^2$.

Disparity-Included Error: $\boldsymbol{e}_i = \begin{bmatrix} I_{k-1}(\boldsymbol{u}_{k-1}^i) - I_k'(\boldsymbol{u}_k^i) \\ d_k^i - \bar{D}_k'(\boldsymbol{u}_k^i) \\ d_{k-1}^i - \bar{d}_{k-1}^i\end{bmatrix}$ to optimize keyframe disparity for better mapping. d_{k-1} is the disparity to be optimized, $\bar{d}_{k-1}$ is the observed disparity (from stereo matching) in the previous frame, and $\bar{D}_k'$ is the interpolated observed disparity in the current frame. Use matrix weights for optimization.

Coarse-to-Fine Optimization: Starting with a blurred and subsampled image, then repeatedly optimizing while increasing resolution, using the previous optimization result as the initial guess for the next iteration. First iteration can use IMU/odometry. This smoothes out local minima and increases region of convergence.

Selective Pixel Matching: Performing matching/optimization only over "informative" pixels, based on gradient magnitude. Divide image into a grid, and for each cell take only the pixel with the largest gradient magnitude for optimization.

Apply Photometric Calibration: Dense mapping is affected by gamma, exposure, vignetting, etc., so photometric calibration can be used to build a model correcting these effects.

13 Classical Image Segmentation

Segmentation Types: *Coarse* (bounding boxes for objects) vs. *fine* (splines or pixel-wise masks/labels to describe boundaries between objects). *Semantic*: segmenting based on object types; *instance*: segmenting based on object types and different instances of the same type; *panoptic*: segmenting instances of countable foreground objects, and regions of uncountable background "stuff".

Thresholding: Simplest method; applying a threshold to image intensity values (or to specific channels in some colour space), then finding connected components.

Active Contours: An energy-minimization approach to fit a spline $\boldsymbol{f}(s) = (u(s), v(s))$ to minimize the energy consisting of the smoothness cost $\mathcal{E}_{\text{int}} = \int \alpha(s)\|\boldsymbol{f}_s(s)\|^2 + \beta(s)\|\boldsymbol{f}_{ss}(s)\|^2\,\mathrm{d}s$ and image energy $\mathcal{E}_{\text{image}} = w_{\text{line}}\mathcal{E}_{\text{line}} + w_{\text{edge}}\mathcal{E}_{\text{edge}} + w_{\text{term}}\mathcal{E}_{\text{term}}$ where $\mathcal{E}_{\text{edge}} = \sum_i -\|\nabla I(\boldsymbol{f}(i))\|^2$. The terms attract the spline to dark ridges, strong gradients, and line terminations respectively.

B-Splines: $\boldsymbol{f}(s) = \sum_k B_k(s)\boldsymbol{x}_k$, where $B_k(s)$ are k basis functions and $\boldsymbol{x}_k$ are control points (i.e. parameters); generalizations of Bezier curves.

Dynamic contours: Model evolving contours with discrete linear dynamics: $\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{w}_t$. Account for multi-modal uncertainty with particle filtering. Known as Conditional Density Propagation/CONDENSATION.

Split-and-Merge/Watershed: Starting from seed points, perform watershed segmentation: Interpret the image intensities as "heights" and fill the local minima with "water", i.e. propagate outward from the minima until we hit a high-intensity boundary. Assumes that the boundaries are similar in intensity. Locality constraints can be applied to close off the regions.

Mean-Shift Algorithm: Cluster pixels in colour space (e.g. LUV) by considering each pixel as a sample from a probability distribution with multiple means, using kernel density estimation to find the PDF, where each mode is a segment.

1. For each mode $\boldsymbol{y}$, start with some initial guess $\boldsymbol{y}_0$.
2. $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \boldsymbol{m}(\boldsymbol{y}_k) = \frac{\sum_i \boldsymbol{x}_i G(\boldsymbol{y}_k - \boldsymbol{x}_i)}{\sum_i G(\boldsymbol{y}_k - \boldsymbol{x}_i)}$ where $\boldsymbol{m}(\boldsymbol{y}_k)$ is the mean-shift, G is the derivative of the kernel density estimator.
3. Repeat until convergence, $\|\boldsymbol{m}(\boldsymbol{y}_k)\| < \epsilon$.

Initialize a mode at each input point, and iterate until all pixels have converged to a mode; then each distinct mode will be a segment.

Kernel Density Estimator: $f(\boldsymbol{x}) = \sum_i k\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{h^2}\right)$ for samples $\boldsymbol{x}_i$, mean $\boldsymbol{x}$, bandwidth (spread) h, and kernel function $k(r)$. Gaussian kernel $k_N(r) = \exp\left(-\frac{1}{2}r\right)$ is commonly used.

Graph Cuts: Construct a graph from the image and cut it into regions. Nodes are pixels. Edges are weighted using an affinity function based on salient properties, e.g. pixel distance, intensity difference, colour difference, texture metrics (e.g. by convolution). Can make minimum cuts (using the max-flow/min-cut algorithm, efficient but tends to over-segment), or normalized cuts (cost of the cut normalized by the size of segments, NP-hard but can be approximated).

Markov Random Fields Formulation: Segmentation label for each pixel are the "true" pixel states x_i, and the image itself is the observed "evidence" y_i. Minimize energy $E(x, y) = \sum_i \varphi(x_i, y_i) + \sum_{ij}\psi(x_i, x_j)$ where φ encodes the segmentation class and ψ encodes smoothness. Can be formulated as a min-cut problem for binary (foreground/background) labelling.

GrabCut: Segments using a user-supplied bounding box and an MRF model. A Gaussian mixture model (GMM) is fit to the foreground and background using the bounding box. The MRF energy is defined based on the GMM, and min-cut is applied to classify into foreground and background. Repeat until convergence, fitting a new GMM each time with the new foreground-background segmentation.

14 Visual Servoing

Controlling a robot (usually a manipulator) relative to a target of interest, by tracking visual features. The camera can be mounted on the robot (**eye-in-hand**) or at a fixed external point. For eye-in-hand there is a risk of moving too quickly and losing sight of the target.

14.1 Position-Based Visual Servoing

Using known calibration and target geometry (i.e. 3D positions of feature points) to estimate target pose, and control the robot in task space (e.g. $SE(3)$) to move to a desired pose $\boldsymbol{T}_{TE}^{*}$ relative to the target. Estimate target pose by minimizing the error function $\boldsymbol{e}_i = \boldsymbol{x}_i - \boldsymbol{f}(\boldsymbol{p}_i; \boldsymbol{C}, \boldsymbol{t}, \boldsymbol{K})$. Requires depth to be estimated or approximated based on target size. Commonly done with fiducial markers like ArUco or AprilTag, with motion estimation using Kalman filtering or particle filtering, either in an inertial frame or a frame relative to the moving camera. Controller can be *end-point open-loop* (using inverse kinematics to move the arm directly to the desired location, often used for external camera) or *eye-in-hand closed-loop* (setting the end-effector velocity based on pose error).

Limitations: No guarantee target image feats will stay in FOV. Can resolve with extra constraints, finite horizon planing, & numerical optimization. Assumes scale/depth knowledge from a known target. Can also solve with approx depths or locking scale on initialization.

14.2 Image-Based Visual Servoing

For hand-in-eye configurations, directly tracking the image feature positions in image space, without explicitly computing the target pose, usually using a linear controller in image space.

Use the camera model Jacobian to solve for the desired camera velocity vector, given the difference between actual $\boldsymbol{p}_i$ and desired $\boldsymbol{p}_i^d$ feature points for at least 3 non-colinear points:

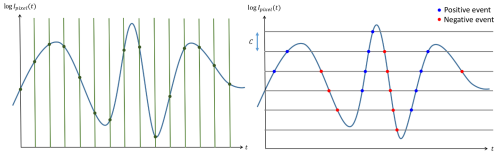
$$\dot{\boldsymbol{p}} = \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{f}{Z} & 0 & \frac{u}{Z} & \frac{uv}{f} & -\frac{f^2+u^2}{f} & v \\ 0 & -\frac{f}{Z} & \frac{v}{Z} & \frac{f^2+v^2}{f} & -\frac{uv}{f} & -u \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \boldsymbol{J}_p \boldsymbol{\nu}$$

$$\begin{bmatrix} \dot{\boldsymbol{p}}_1 \\ \vdots \\ \dot{\boldsymbol{p}}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{p_1} \\ \vdots \\ \boldsymbol{J}_{p_n} \end{bmatrix} \boldsymbol{\nu} \implies \boldsymbol{\nu} = \lambda \begin{bmatrix} \boldsymbol{J}_{p_1} \\ \vdots \\ \boldsymbol{J}_{p_n} \end{bmatrix}^{\dagger} \begin{bmatrix} \boldsymbol{p}_1^d - \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_n^d - \boldsymbol{p}^n \end{bmatrix}$$

Limitations: Since camera motion is calculated implicitly, can fail to converge for cases where movement is ambiguous, e.g. rotation by π for rectangular set of feature pts. Requires feat. depth estimates but usually robust to them.

15 Alternative Imaging Systems

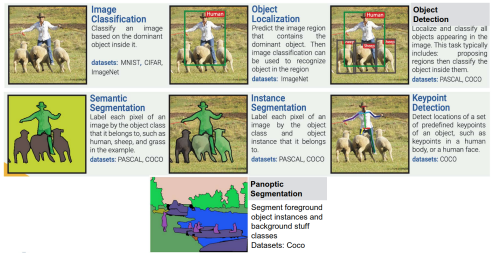
Event Camera: Measures light intensity variations in a scene, asynchronously for every pixel. Instead of outputting intensity values, an event is triggered when $L(x, y, t + \Delta t) - L(x, y, t) = \pm C$ where $L(x, y, t) = \log I(x, y, t)$. Each event is a tuple $(t, x, y, \text{sign}\frac{dI}{dt})$. For a point moving with velocity $\boldsymbol{u} = (u, v)$, events are generated where $-\Delta L \cdot \boldsymbol{u} = \pm C$, i.e. events are triggered along edges in the image when movement is perpendicular to an edge.



Advantages include much higher dynamic range (140 dB), very low latency/high framerate (1 MHz), negligible motion blur, low bandwidth use, and very low power consumption. Applications include capturing very fast motions and removing motion blur when combined with traditional camera.

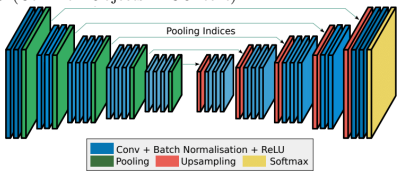
Thermal Camera: Measures the infrared spectrum instead of visible light. Can see with no light since objects emit their own infrared radiation, can see through fog, smoke, etc, and determine temperature of objects. Not photometrically consistent over time due to the camera heating itself, non-uniform sensor responses, and spatial artifacts, which requires calibration to remove. Can be combined with conventional cameras, especially in low-light scenarios where thermal cameras lead to many more feature matches.

16 Learning-Based Methods



16.1 Segmentation

Datasets: KITTI (only 200 training/test images), TUM SceneFlow (synthetic), City Scapes (large-scale instance-level segmentations), COCO (Common Objects in COntext).



Architecture: Commonly based on U-Net or SegNet architectures, first convolutions and downsampling until a bottleneck layer, then upsampling to recover an image with the original resolution. In U-Net upsampling is done through upconvolutions, and there are skip connections between the upsampling and downsampling layers. In SegNet upsampling is done through max unpooling and there are no skip connections.

Basic Upsampling: Replacing each cell with the value of its nearest neighbour, or leaving cells without a value as zero (bed-of-nails).

Max-Unpooling: Remembering which indices had the max value during max pooling, then restoring the values at these indices during upsampling with a bed of nails.

Learnable Upsampling/Upconvolution: Using a learnable kernel, multiplied by each input value and summed in the output, giving weighted copies of the filter.

16.2 Localization/Pose Estimation

PoseNet: A CNN (GoogLeNet) for 6-DoF camera relocalization from a single image. Loss function $\mathcal{L}(I) = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\| + \beta \left\| \hat{\boldsymbol{q}} - \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|} \right\|$. Accuracy is much worse than classical methods, but faster and uses less memory (represents map with fixed-size network).

SfMLearner: Unsupervised learning using photometric loss for monocular depth prediction and relative pose prediction. Consists of a depth CNN (U-Net) and pose/explainability network (similar to SegNet). During training, depth predictions and relative pose predictions are used to warp the current image (similar to dense/photometric mapping), and loss compares the prediction with the actual next image. The explainability mask filters out moving objects and occlusions; during training it also has a smoothness and regularization (encouraging nonzero predictions) term to prevent it from masking out all points.

Sun-BCNN: A Bayesian CNN (GoogLeNet) for predicting the angle of the sun (and confidence) in an image, which can be used in visual odometry to constrain the error to linear growth, without a dedicated sun sensor. Uses cosine distance loss $\mathcal{L} = 1 - \hat{\boldsymbol{s}}_k \cdot \boldsymbol{s}_k$.

Combining Classical and Learning Based Approaches:

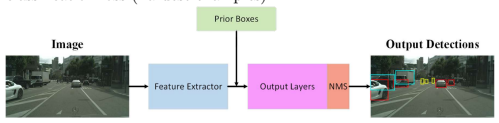
1. Correction: Use classical techniques to estimate a solution, then use a learning based method to apply a correction, e.g. correcting systematic bias in pose estimation.
2. Augmentation: Using learning based methods to get additional information, then using a classical method to predict the solution with the extra information, e.g. Sun-BCNN.
3. Initialization: Using learning methods to get an initial guess for the solution, then using classical methods to refine the solution accuracy, e.g. using PoseNet for initialization, then classic feature matching.

16.3 Object Localization and Detection

Nonmaximum Suppression: Used to prune overlapping network outputs. First sort all predictions by confidence score. Then for each box, starting with the highest confidence, include it in the final output set, and prune all remaining boxes (not in the output set) whose IoU with the box we just included is over some specified threshold. Note each class should be considered separately.

Bounding Box/Region Proposals: Tuples (x, y, w, h) and confidence, indicating a rectangular region. Can use simple L2 distance of the bounding box vectors as a similarity metric, or Intersection over Union (IoU) (ratio of overlap area over total area).

Region Proposal Methods: Identify potential regions (*anchors*) with an algorithm or region proposal network, then for each region run region run a detector network to calculate the class scores (or background), and a regression network to calculate a bounding box correction. Take all regions over a confidence threshold and apply NMS for final output. During training, loss is evaluated on all anchors instead of best ones. Take 3:1 ratio of negative to positive anchors to avoid bias towards negative samples, choosing negatives with highest classification loss (hardest examples).



R-CNN: Used classical algorithm for proposals, CNN and SVM to calculate class scores and regress bounding box corrections.

Fast R-CNN: Passing the entire image through a CNN to get a feature map first, then cropping the feature map for each proposal region.

Faster R-CNN: Using a learned region proposal network to generate proposals.

Single-Shot Methods (YOLO): Predicting bounding box coordinates and class probabilities in a single pass, without proposals. Break the image into $S \times S$ grid, where each cell predicts B possible bounding boxes and confidences and C class probabilities (including background). Outputs $S \times S \times (5B + C)$ values in total. Multiply bounding box confidence by class probability, take the highest combined confidence for each class and apply NMS to get final output.

Generally single-shot approaches are much faster and can run in real-time, but are slightly less accurate than proposal based approaches.

16.4 3D Object Detection

AVOD-FPN: Generate ground-anchored proposals from both an image feature map and a birds-eye-view LiDAR feature map, then fusing the two, performing detections in image and BEV LiDAR separately, then fusing the results again.

Categorical Depth Distribution Network: For monocular 3DOD, instead of generating a single depth for each pixel, generate a depth distribution represented as a histogram instead.

BayesOD: Generating probabilistic uncertainty estimates instead of just confidences from 3DOD so it can be fused with other sensors. Instead of NMS, each detection can be treated as a separate measurement for uncertainty estimation. The network can regress a variance directly. Can also use dropout on the network during inference and measure the difference between predictions.

Domain Adaptation: Transferring a detector trained on one dataset to another, e.g. adapting it for adverse weather. Can be done through a student-teacher architecture, where the teacher provides pseudo-labels on the target dataset to train the student with (taking only the most confident detections), and the student is taught to ignore dataset variations (by training it to produce the same output for an original image vs. a corrupted image) while retaining detections. Foundation models such as DINOv2 can be used in the teacher network due to their excellent domain transfer capabilities.

17 Application: Computer Vision on Mars

MER-DIMES: Used to estimate horizontal velocity during landing of MER to determine when to fire rockets. Inputs: radar distance to ground, monocular camera. Estimates motion over 3 frames by using 2 features and fusing with IMU/altitude. Pipeline: Downsampling, masking shadow, determine image overlap, Harris corners, correction for CCD effects, patch rectification, then feature matching over 2-scale pyramid.

Spirit and Opportunity: Used 3 sets of stereo, hazcams under solar panels for close-range obstacle detection, navcams on main mast for visual odometry (wheel odometry insufficient due to wheel slip and degradation of calibration), pancams for long-range panoramic imaging. Estimate traversibility by fitting planes to patches, checking for protrusions and tilt, then inflating an occupancy grid. Even more cameras on the larger Mars Science Laboratory.

Ingenuity Helicopter VO: Using a single camera, RANSAC for outlier rejection and shadow removal, matching using 12 FAST features, accounts for motion blur, lighting changes, feature-poor terrain.