Guest Lecture: M.Sc. Jonathon Luiten
RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

**Announcements**

- **Exams**
  - We are in the process of sending around the exam slot assignments.
  - If the assigned date doesn’t work for you, please contact us.

- **Exam Procedure**
  - Oral exams
  - Duration 30min
  - I will give you 4 questions and expect you to answer 3 of them.

- **Announcements (2)**
  - Today, we’ll summarize the most important points from the lecture.
    - It is an opportunity for you to ask questions...
    - ... or get additional explanations about certain topics.
    - So, please do ask.
  - Today’s slides are intended as an index for the lecture.
    - But they are not complete, won’t be sufficient as only tool.
    - Also look at the exercises – they often explain algorithms in detail.

**Content of the Lecture**

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis

**Recap: Gaussian Background Model**

- **Statistical model**
  - Value of a pixel represents a measurement of the radiance of the first object intersected by the pixel’s optical ray.
  - With a static background and static lighting, this value will be a constant affected by i.i.d. Gaussian noise.

- **Idea**
  - Model the background distribution of each pixel by a single Gaussian centered at the mean pixel value:
    \[ N(x|\mu, \Sigma) = \frac{1}{(2\pi)^d \det(\Sigma)^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]
  - Test if a newly observed pixel value has a high likelihood under this Gaussian model.

  ⇒ Automatic estimation of a sensitivity threshold for each pixel.

**Recap: Stauffer-Grimson Background Model**

- **Idea**
  - Model the distribution of each pixel by a mixture of \( K \) Gaussians
    \[ p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \]
  - Check every new pixel value against the existing \( K \) components until a match is found (pixel value within 2.5 \( \sigma_k \) of \( \mu_k \)).
  - If a match is found, adapt the corresponding component.
  - Else, replace the least probable component by a distribution with the new value as its mean and an initially high variance and low prior weight.
  - Order the components by the value of \( w_k/\sigma_k \) and select the best \( B \) components as the background model, where
    \[ B = \arg \min_0 \left( \sum_{k=0}^{2.5} \frac{w_k}{\sigma_k} > T \right) \]
Recap: Stauffer-Grimson Background Model

- **Online adaptation**
  - Instead of estimating the MoG using EM, use a simpler online adaptation, assigning each new value only to the matching component.
  - Let $M_k(t) = 1$ if component $k$ is the model that matched, else 0.
  - $x_k^{(t+1)} = (1 - \alpha)x_k^{(t)} + \alpha M_k(t)$
  - Adapt only the parameters for the matching component where (i.e., the update is weighted by the component likelihood)

[C. Stauffer, W.E.L. Grimson, CVPR'99]

Recap: Kernel Background Modeling

- **Nonparametric density estimation**
  - Estimate a pixel's background distribution using the kernel density estimator $K(x)$ as
  - Choose $K$ to be a Gaussian $N(0, \Sigma)$ with $\Sigma = \text{diag}(\sigma_i)$. Then
  - A pixel is considered foreground if $p(x^{(t)}) < \theta$ for a threshold $\theta$.
  - This can be computed very fast using lookup tables for the kernel function values, since all inputs are discrete values.
  - Additional speedup: partial evaluation of the sum usually sufficient

[A. Elgammal, D. Harwood, L. Davis, ECCV'00]

Content of the Lecture

- **Single-Object Tracking**
  - Background modeling
  - Template based tracking
  - Tracking by online classification
  - Tracking-by-detection
- **Bayesian Filtering**
- **Multi-Object Tracking**
- **Visual Odometry**
- **Visual SLAM & 3D Reconstruction**
- **Deep Learning for Video Analysis**

Recap: Estimating Optical Flow

- **Optical Flow**
  - Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ between them.
  - **Key assumptions**
    - **Brightness constancy**: projection of the same point looks the same in every frame.
    - **Small motion**: points do not move very far.
    - **Spatial coherence**: points move like their neighbors.

Recap: Lucas-Kanade Optical Flow

- **Use all pixels in a $K \times K$ window to get more equations.**
- **Least squares problem:**
  - $\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ I_x(p_2) & I_y(p_2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_x(p_1) \\ I_y(p_2) \end{bmatrix} \begin{bmatrix} 2x2 \\ 2x1 \end{bmatrix}$
  - $A \cdot d = b$
  - Minimum least squares solution given by solution of
  - $A^T b = (A^T A) d = A^T b$

Recap: Iterative LK Refinement

- **Estimate velocity at each pixel using one iteration of LK estimation.**
- **Warp one image toward the other using the estimated flow field.**
- **Refine estimate by repeating the process.**
- **Iterative procedure**
  - Results in subpixel accurate localization.
  - Converges for small displacements.
Repetition

We assume that an initial estimate of the image displacement \( y \) is known and iteratively solve for the parameters \( \Delta p \).

Compute steepest descent images:

\[ \nabla W = \begin{bmatrix} \frac{\partial W}{\partial p_1} \\ \frac{\partial W}{\partial p_2} \\ \vdots \\ \frac{\partial W}{\partial p_n} \end{bmatrix} \]

Increment parameters to solve for \( \Delta p \):

\[ \nabla I = \frac{\partial I}{\partial p} \]

Recap: Inverse Compositional LK Algorithm

• Iterate
   - Warp \( I \) to obtain \( I(W(\{x, y\}; p)) \)
   - Compute the error image \( I(W(\{x, y\}; p)) - I(W(\{x, y\}; \hat{p})) \)
   - Warp the gradient \( \nabla I \) with \( W(\{x, y\}; \hat{p}) \)
   - Evaluate \( \frac{\partial W}{\partial p} \) at \( (x, y); p \) (Jacobian)
   - Compute steepest descent images:
     \[ \nabla W = \begin{bmatrix} \frac{\partial W}{\partial p_1} \\ \frac{\partial W}{\partial p_2} \\ \vdots \\ \frac{\partial W}{\partial p_n} \end{bmatrix} \]
   - Compute Hessian matrix:
     \[ H = \sum_{x} \begin{bmatrix} \frac{\partial W}{\partial p_1} \\ \frac{\partial W}{\partial p_2} \\ \vdots \\ \frac{\partial W}{\partial p_n} \end{bmatrix} \begin{bmatrix} \frac{\partial W}{\partial p_1} & \frac{\partial W}{\partial p_2} & \cdots & \frac{\partial W}{\partial p_n} \end{bmatrix} \]
   - Compute \( \Delta p = H^{-1} \sum_{x} \begin{bmatrix} \frac{\partial W}{\partial p_1} \\ \frac{\partial W}{\partial p_2} \\ \vdots \\ \frac{\partial W}{\partial p_n} \end{bmatrix} \begin{bmatrix} I(W(\{x, y\})) - I(W(\{x, y\}; p)) \end{bmatrix} \)
   - Update the parameters \( p = p + \Delta p \)
   - Until \( \Delta p \) magnitude is negligible

Recap: General LK Image Registration

• Goal
   - Find the warping parameters \( p \) that minimize the sum-of-squares intensity difference between the template image \( I(x) \) and the warped input image \( I(W(x; p)) \).

   \[ \arg\min_p \sum_x |I(W(x; p)) - T(x)|^2 \]

   - We assume that an initial estimate of \( p \) is known and iteratively solve for increments to the parameters \( \Delta p \):

Recap: Coarse-to-fine Optical Flow Estimation

• Key to the derivation
  - Taylor expansion around \( \Delta p \)

\[ I(W(x; p + \Delta p)) \approx I(W(x; p)) + \nabla I(W(x; p)) \Delta p + O(\Delta p^2) \]

  \[ = I(W(x; p)) + \nabla I(W(x; p)) \Delta p \]

  \[ + \begin{bmatrix} \frac{\partial I}{\partial p_1} \\ \frac{\partial I}{\partial p_2} \\ \vdots \\ \frac{\partial I}{\partial p_n} \end{bmatrix} \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_n \end{bmatrix} \]

  \[ = I(W(x; p)) + \nabla I(W(x; p)) \Delta p \]

  \[ + \begin{bmatrix} \frac{\partial I}{\partial p_1} \\ \frac{\partial I}{\partial p_2} \\ \vdots \\ \frac{\partial I}{\partial p_n} \end{bmatrix} \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_n \end{bmatrix} \]

  \[ = I(W(x; p)) + \nabla I(W(x; p)) \Delta p \]

Gradient

Jacobian

Recap: Step-by-Step Derivation

Recap: Shi-Tomasi Feature Tracker (\( \rightarrow \)KLT)

• Idea
  - Find good features using eigenvalues of second-moment matrix
  - Key idea: “good” features to track are the ones that can be tracked reliably.

• Frame-to-frame tracking
  - Track with LK and a pure translation motion model.
  - More robust for small displacements, can be estimated from smaller neighborhoods (e.g., \( 5 \times 5 \) pixels).

• Checking consistency of tracks
  - Affine registration to the first observed feature instance.
  - Affine model is more accurate for larger displacements.
  - Comparing to the first frame helps to minimize drift.

Recap: Inverse Compositional LK Algorithm

Recap: Tracking as Online Classification


Recap: Tracking as Online Classification

Course Outline

- Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Tracking by online classification
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis

Recap:

AdaBoost – Algorithm

1. Initialization: Setting $w^{(i)} = \frac{1}{N}$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$ iterations:
   a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function
   $$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)$$
   where $I$ is $1$, if $A$ is true and $0$, otherwise.
   b) Estimate the weighted error of this classifier on $X$:
   $$\epsilon_m = \frac{1}{N} \sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)$$
   c) Calculate a weighting coefficient for $h_m(x)$:
   $$\alpha_m = \ln \left( \frac{1 - \epsilon_m}{\epsilon_m} \right)$$
   d) Update the weighting coefficients:
   $$w_n^{(m+1)} = w_n^{(m)} \exp \left( \alpha_m I(h_m(x_n) \neq t_n) \right)$$
From Offline to Online Boosting

- Main issue
  - Computing the weight distribution for the samples.
  - We do not know a priori the difficulty of a sample!
  - (Could already have seen the same sample before...)
- Idea of Online Boosting
  - Estimate the importance of a sample by propagating it through a set of weak classifiers.
  - This can be thought of as modeling the information gain w.r.t. the first $n$ classifiers and code it by the importance weight $\lambda$ for the $n+1$ classifier.
  - Proven [Oza]: Given the same training set, Online Boosting converges to the same weak classifiers as Offline Boosting in the limit of $N \to \infty$ iterations.


Recap: From Offline to Online Boosting

\begin{align*}
\text{off-line} & \quad \text{on-line} \\
\text{Given:} & \quad \text{Given:} \\
\text{set of labeled training samples} & \quad \text{ONE labeled training sample} \\
X = \{(x_1, y_1), \ldots, (x_N, y_N)\} & \quad \gamma = 1 \\
\text{weight distribution over them} & \quad \text{strong classifier to update} \\
D_0 = \frac{1}{N} & \quad \text{initial importance } \lambda = 1 \\
\text{for } n = 1 \text{ to } N & \quad \text{for } n = 1 \text{ to } N \\
\text{- train a weak classifier using} & \quad \text{- train the weak classifier using} \\
\text{samples and weight dist.} & \quad \text{samples and importance} \\
X, D_n & \quad \lambda(X, D_n, \gamma) \\
\text{- calculate error } \eta_n = f(x_n) & \quad \text{- update error estimation } \hat{c}_n \\
\text{- calculate weight } \alpha_n = \frac{1}{2} \text{exp}(\eta_n) & \quad \text{- update weight } \alpha_n = \frac{1}{2} \text{exp}(\eta_n) \\
\text{- update weight dist. } D_n & \quad \text{- update importance weight } \lambda \\
\lambda^{n+1}(x) = \text{sigm}(\sum_{i=1}^{N} \alpha_n \cdot \lambda^n_i(x)) & \quad \text{next} \\
\text{next} \\
\end{align*}

Recap: Online Boosting for Feature Selection

- Introducing “Selector”
  - Selects one feature from its local feature pool
  - $\mathcal{F} = \{h^{weak_1}, \ldots, h^{weak_m}\}$
  - $\mathcal{F} = \{f_1, \ldots, f_n\}$
  - $m = \text{arg min}_i \mathcal{F}(x)$

On-line boosting is performed on the Selectors and not on the weak classifiers directly.


Recap: Tracking by Online Classification

- Update classifier (tracker)
- from time $t$ to $t+1$
- Evaluate classifier on sub-patches
- Search region
- Create confidence map
- Search region

Recap: Drifting Due to Self-Learning Policy

- Not only does it drift, it also remains confident about it!
Recap: Tracking-by-Detection

- **Main ideas**
  - Apply a generic object detector to find objects of a certain class
  - Based on the detections, extract object appearance models
  - Link detections into trajectories

Recap: Sliding-Window Object Detection

- For sliding-window object detection, we need to:
  1. Obtain training data
  2. Define features
  3. Define a classifier

Recap: Histograms of Oriented Gradients (HOG)

- **Holistic object representation**
  - Localized gradient orientations

Recap: Object Detector Design

- In practice, the classifier often determines the design.
  - Types of features
  - Speedup strategies
- We looked at 3 state-of-the-art detector designs
  - Based on SVMs
  - Based on Boosting
  - Based on CNNs

Recap: Elements of Tracking

- **Detection**
  - Where are candidate objects?
- **Data association**
  - Which detection corresponds to which object?
- **Prediction**
  - Where will the tracked object be in the next time step?
Recap: Deformable Part-based Model (DPM)

• Multiscale model captures features at two resolutions

Score of object hypothesis is sum of filter scores minus deformation costs

Recap: DPM Hypothesis Score

\[
\text{score}(p_0, \ldots, p_n) = \sum_{l=0}^{n} F_l \cdot \phi(H, p_l) - \sum_{i=1}^{n} d_i \cdot (dx_i, dy_i),
\]

concatenation of HOG features and deformation parameters

Recap: Integral Channel Features

• Generalization of Haar Wavelet idea from Viola-Jones
  – Instead of only considering intensities, also take into account other feature channels (gradient orientations, color, texture).
  – Still efficiently represented as integral images.

P. Dollar, Z. Tu, P. Perona, S. Belongie. Integral Channel Features, BMVC'09.

Recap: Integral Channel Features

• Generalize also block computation
  – 1st order features:
    • Sum of pixels in rectangular region.
  – 2nd-order features:
    • Haar-like difference of sum-over-blocks
  – Generalized Haar:
    • More complex combinations of weighted rectangles
  – Histograms
    • Computed by evaluating local sums on quantized images.

Recap: VeryFast Detector

• Idea 1: Invert the template scale vs. image scale relation

1 model, 50 image scales

50 models, 1 image scale


Recap: VeryFast Detector

• Idea 2: Reduce training time by feature interpolation

50 models, 1 image scale

5 models, 1 image scale

• Shown to be possible for Integral Channel features
Recap: VeryFast Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + w_2 \cdot h_2 + \ldots + w_N \cdot h_N \]

Recap: Convolutional Neural Networks

- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end


Recap: Convolution Layers

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single \([1 \times 1 \times \text{depth}]\) depth column in output volume.

Recap: Activation Maps

- 5\times5 filters

Recap: Pooling Layers

- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations
Recap: Faster R-CNN

- One network, four losses
  - Remove dependence on external region proposal algorithm.
  - Instead, infer region proposals from same CNN.
  - Feature sharing
  - Joint training
  ⇒ Object detection in a single pass becomes possible.

Recap: Mask R-CNN


Recap: YOLO / SSD

- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)

Recap: Tracking as Inference

- Inference problem
  - The hidden state consists of the true parameters we care about, denoted \( X \).
  - The measurement is our noisy observation that results from the underlying state, denoted \( Y \).
  - At each time step, state changes (from \( X_{t-1} \) to \( X_t \)) and we get a new observation \( Y_t \).
- Our goal: recover most likely state \( X_t \) given
  - All observations seen so far.
  - Knowledge about dynamics of state transitions.

Recap: Tracking as Induction

- Base case:
  - Assume we have initial prior that predicts state in absence of any evidence: \( f(X_0) \)
  - At the first frame, correct this given the value of \( Y_0 = y_0 \)
- Given corrected estimate for frame \( t \):
  - Predict for frame \( t+1 \)
  - Correct for frame \( t+1 \)

Course Outline

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis
Recap: Prediction and Correction

• Prediction:

$$P(X_t | y_0, \ldots, y_{t-1}) = \int P(X_t | X_{t-1}) P(X_{t-1} | y_0, \ldots, y_{t-1}) \, dX_{t-1}$$

Dynamics model
Corrected estimate from previous step

• Correction:

$$P(X_t | y_0, \ldots, y_{t}) = \frac{P(X_t | X_{t-1}) P(X_{t-1} | y_0, \ldots, y_{t-1})}{P(y_0, \ldots, y_{t} | X_{t-1})}$$

Observation model
Predicted estimate

Recap: Measurement is linearly transformed state plus Gaussian noise

$$y_t \sim \mathcal{N}(M x_t, \Sigma_{y_t})$$

Recap: Linear Dynamic Models

• Dynamics model

- State undergoes linear transformation $D$, plus Gaussian noise

$$x_t \sim \mathcal{N}(D x_{t-1}, \Sigma_d)$$

• Observation model

- Measurement is linearly transformed state plus Gaussian noise

$$y_t \sim \mathcal{N}(M x_t, \Sigma_{y_t})$$

Recap: Constant Velocity (1D Points)

• State vector: position $p$ and velocity $v$

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix}$$

- (greek letters denote noise terms)

$$v_t = p_{t-1} + (\Delta t) v_{t-1} + \xi$$

- Measurement is position only

$$y_t = M x_t + \text{noise}$$

Recap: Constant Acceleration (1D Points)

• State vector: position $p$, velocity $v$, and acceleration $a$.

$$x_t = \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix}$$

- (greek letters denote noise terms)

$$a_t = v_{t-1} + \frac{(\Delta t)^2}{2} p_{t-1} + \epsilon$$

- Measurement is position only

$$y_t = M x_t + \text{noise}$$

Recap: General Motion Models

• Assuming we have differential equations for the motion

- E.g. for (undamped) periodic motion of a linear spring

$$\frac{d^2 p}{dt^2} = -p$$

- Substitute variables to transform this into linear system

$$p_1 = p, \quad p_2 = \frac{dp}{dt}, \quad p_3 = \frac{d^2 p}{dt^2}$$

- Then we have

$$x_t = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$p_1 = p_{t-1} + (\Delta t) v_{t-1} + \frac{(\Delta t)^2}{2} a_{t-1} + \epsilon$$

$$p_2 = p_{t-1} + (\Delta t) p_{t-1} + \xi$$

$$p_3 = -p_{t-1} + \zeta$$

Recap: The Kalman Filter

Know corrected state from previous time step, and all measurements up to the current one

- Predict distribution over next state.

Receive measurement
Know prediction of state, and next measurement

- Update distribution over current state.

Time update ("Predict")
Measurement update ("Correct")

Mean and std. dev.
of predicted state:

$$\mu_i, \sigma_i$$

Mean and std. dev.
of corrected state:

$$\mu_f, \sigma_f$$

Time advances: $t \rightarrow t+1$
Recap: General Kalman Filter (>1dim)

\[
\begin{align*}
\text{PREDICT} & : x_t^* = D_t x_{t-1} + \varepsilon_t \\
& \quad \Sigma_t^* = D_t \Sigma_{t-1} D_t^T + \Sigma_e
\end{align*}
\]

\[
\begin{align*}
\text{CORRECT} & : x_t = x_t^* + K_t (y_t - h(x_t^*)) \\
& \quad \Sigma_t = (I - K_t H_t) \Sigma_t^*
\end{align*}
\]

More weight on residual when measurement error covariance approaches 0.
Less weight on residual as a priori estimate error covariance approaches 0.

Extended Kalman Filter (EKF)

- Algorithm summary
  - Nonlinear model
    \[
    \begin{align*}
    x_t &= g(x_{t-1}) + \varepsilon_t \\
y_t &= h(x_t) + \delta_t
    \end{align*}
    \]
    \text{with the Jacobians}
  - Prediction step
    \[
    \begin{align*}
x_t^* &= g(x_{t-1}^*) \\
\Sigma_t^* &= G_t \Sigma_{t-1} G_t^T + \Sigma_{\delta_t}
\end{align*}
    \]
  - Correction step
    \[
    \begin{align*}
    K_t &= \Sigma_t^* H_t^T (H_t \Sigma_t^* H_t^T + \Sigma_{\delta_t})^{-1} \\
    x_t &= x_t^* + K_t (y_t - h(x_t^*)) \\
\Sigma_t &= (I - K_t H_t) \Sigma_t^*
\end{align*}
    \]

Recap: Propagation of General Densities

Recap: Factored Sampling

\[P(X_1, X_2, \ldots, X_n) = \frac{P(X_1 \mid X_2)P(X_2 \mid X_3, \ldots, X_n)}{\int P(Y_1 \mid X_1)P(X_1 \mid Y_1, \ldots, Y_n) dX_1}\]

Course Outline

- Single-Object Tracking
- Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis
Recap: Particle Filtering

- Many variations, one general concept:
  - Represent the posterior pdf by a set of randomly chosen weighted samples (particles)
  - Randomly Chosen = Monte Carlo (MC)
  - As the number of samples become very large – the characterization becomes an equivalent representation of the true pdf.

Background: Particle Filtering

- Objective:
  - Evaluate expectation of a function \( f(x) \) w.r.t. a probability distribution \( p(x) \):
    \[
    \mathbb{E}[f] = \int f(x)p(x)dx
    \]
- Monte Carlo Sampling idea
  - Draw \( L \) independent samples \( x^l \) with \( l = 1, \ldots, L \) from \( p(x) \).
  - This allows the expectation to be approximated by a finite sum
    \[
    f = \frac{1}{L} \sum_{l=1}^{L} f(x^l)
    \]
  - As long as the samples \( x^l \) are drawn independently from \( p(x) \), then
    \[
    \mathbb{E}[f] = \mathbb{E}[\hat{f}]
    \]

Background: Importance Sampling

- Idea
  - Use a proposal distribution \( q(x) \) from which it is easy to draw samples and which is close in shape to \( f \).
  - Express expectations in the form of a finite sum over samples \( \{x^l\} \) drawn from \( q(x) \):
    \[
    \mathbb{E}[f] = \int f(x)p(x)dx = \int f(x)p(x)q(x)dx
    \]
    \[
    \approx \frac{1}{L} \sum_{l=1}^{L} p(x^l)q(x^l) f(x^l)
    \]
  - with importance weights
    \[
    w_l = \frac{p(x^l)}{q(x^l)}
    \]

Recap: Sequential Importance Sampling

For a concrete algorithm, we need to define the importance density \( q(\cdot | \cdot) \).

Recap: Sequential Importance Sampling

function \( \{x_i^l, w_i^l\}_{i=1}^{N} = \text{SIS} \{x_{i-1}^l, w_{i-1}^l\}_{i=1}^{N}, y_i \)

\( \eta = 0 \)

for \( i = 1: N \)

\( x_i^l \sim q(x^l | x_{i-1}^l, y_i) \) \hspace{2cm} \text{Sample from proposal pdf}

\( w_i^l = w_{i-1}^l \frac{p(y_i | x^l) \cdot p(x^l | x_{i-1}^l)}{q(x^l | x_{i-1}^l, y_i)} \) \hspace{2cm} \text{Update weights}

\( \eta = \eta + w_i^l \) \hspace{2cm} \text{Update norm. factor}

end

for \( i = 1: N \)

\( w_i^l = w_i^l / \eta \) \hspace{2cm} \text{Normalize weights}

end

Recap: SIS Algorithm with Transitional Prior

function \( \{x_i^l, w_i^l\}_{i=1}^{N} = \text{SIS} \{x_{i-1}^l, w_{i-1}^l\}_{i=1}^{N}, y_i \)

\( \eta = 0 \)

for \( i = 1: N \)

\( x_i^l \sim p(x_i^l | x_{i-1}^l) \) \hspace{2cm} \text{Sample from proposal pdf}

\( w_i^l = w_{i-1}^l \frac{p(y_i | x_i^l)}{q(x_i^l | x_{i-1}^l, y_i)} \) \hspace{2cm} \text{Update weights}

\( \eta = \eta + w_i^l \) \hspace{2cm} \text{Update norm. factor}

end

for \( i = 1: N \)

\( w_i^l = w_i^l / \eta \) \hspace{2cm} \text{Normalize weights}

end
Recap: Resampling

- Degeneracy problem with SIS
  - After a few iterations, most particles have negligible weights.
  - Large computational effort for updating particles with very small contribution to $p(x_t | y_{1:t})$.

- Idea: Resampling
  - Eliminate particles with low importance weights and increase the number of particles with high importance weight.
  - The new set is generated by sampling with replacement from the discrete representation of $p(x_t | y_{1:t})$ such that $P(Y_w)$.

Recap: Efficient Resampling Approach

- From Arulampalam paper:
  
  - From Arulampalam paper:
    
    - Basic idea: choose one initial small random number; deterministically sample the rest by "crawling" up the cdf.
    
    - This is $O(N)$.

Recap: Generic Particle Filter

- Sample $[x_i, w_i]_{i=1}^N = \text{PP} \left( \left[ x_{i-1}, w_{i-1} \right]_{i=1}^N, y_t \right)$
- Apply SIS filtering $\left[ x_i, w_i \right]_{i=1}^N = \text{SIS} \left( \left[ x_{i-1}, w_{i-1} \right]_{i=1}^N, y_t \right)$

Recap: Sampling-Importance-Resampling (SIR)

- Calculate $N_{eff}$
  
  - If $N_{eff} < N_{thr}$:
    
    - Resample $\left[ x_i, w_i \right]_{i=1}^N = \text{RESAMPLE} \left( \left[ x_i, w_i \right]_{i=1}^N \right)$

Recap: Sampling-Importance-Resampling (SIR)

- We can also apply resampling selectively
  
  - Only resample when it is needed, i.e., $N_{eff}$ is too low.
  
  - Avoids drift when the tracked state is stationary.

Recap: Condensation Algorithm

- Start with weighted samples from previous time step
  - Sample and shift according to dynamics model
  - Spread due to randomness; this is predicted density $P(X_t | y_{t-1})$
  - Weight the samples according to observation density
  - Arrive at corrected density estimate $P(X_t | y_t)$


Basic idea: choose one initial small random number; deterministically sample the rest by "crawling" up the cdf. This is $O(N)$. 

Important property: Particles are distributed according to pdf from previous time step.

Important property: Particles are distributed according to posterior from this time step.
Recap: Motion Correspondence Ambiguities

1. Predictions may not be supported by measurements
   - Have the objects ceased to exist, or are they simply occluded?
2. There may be unexpected measurements
   - Newly visible objects, or just noise?
3. More than one measurement may match a prediction
   - Which measurement is the correct one (what about the others)?
4. A measurement may match to multiple predictions
   - Which object shall the measurement be assigned to?

Recap: Track-Splitting Filter

• Idea
  - Instead of assigning the measurement that is currently closest, as in the NN algorithm, select the sequence of measurements that minimizes the total Mahalanobis distance over some interval!
  - Form a track tree for the different association decisions
  - Modified log-likelihood provides the merit of a particular node in the track tree.
  - Cost of calculating this is low, since most terms are needed anyway for the Kalman filter.

• Problem
  - The track tree grows exponentially, may generate a very large number of possible tracks that need to be maintained.

Recap: Pruning Strategies

• In order to keep this feasible, need to apply pruning
  - Deleting unlikely tracks
    - May be accomplished by comparing the modified log-likelihood $\lambda(k)$, which has a $\chi^2$ distribution with $\nu$ degrees of freedom, with a threshold $\gamma$ (set according to $\chi^2$ distribution tables).
    - Problem for long tracks: modified log-likelihood gets dominated by old terms and responds very slowly to new ones.
    $\Rightarrow$ Use sliding window or exponential decay term.
  - Merging track nodes
    - If the state estimates of two track nodes are similar, merge them.
    - E.g., if both tracks validate identical subsequent measurements.
  - Only keeping the most likely $N$ tracks
    - Rank tracks based on their modified log-likelihood.
Recap: Multi-Hypothesis Tracking (MHT)

• Ideas
  – Instead of forming a track tree, keep a set of hypotheses that generate child hypotheses based on the associations.
  – Enforce exclusion constraints between tracks and measurements in the assignment.
  – After hypothesis generation, merge and prune the current hypothesis set.


Recap: Hypothesis Generation

• Create hypothesis matrix of the feasible associations

\[ \begin{bmatrix}
  \mathbf{X}_1 & \mathbf{X}_2 & \mathbf{X}_{fa} & \mathbf{X}_{nt} \\
  1 & 0 & 1 & 0 \\
  1 & 1 & 1 & 0 \\
  0 & 1 & 1 & 0 \\
  0 & 0 & 1 & 1 
\end{bmatrix} \]

• Interpretation
  – Columns represent tracked objects, rows encode measurements
  – A non-zero element at matrix position \((i,j)\) denotes that measurement \(y_i\) is contained in the validation region of track \(x_j\).
  – Extra column \(x_{fa}\) for association as false alarm.
  – Extra column \(x_{nt}\) for association as new track.

Recap: Assignments

\[
\begin{array}{cccc}
  Z_j & x_1 & x_2 & x_{fa} & x_{nt} \\
  y_1 & 0 & 0 & 1 & 0 \\
  y_2 & 1 & 0 & 0 & 0 \\
  y_3 & 0 & 1 & 0 & 0 \\
  y_4 & 0 & 0 & 1 & 1 \\
\end{array}
\]

• Impose constraints
  – A measurement can originate from only one object.
  ⇒ Any row has only a single non-zero value.
  – An object can have at most one associated measurement per time step.
  ⇒ Any column has only a single non-zero value, except for \(x_{fa}, x_{nt}\).

Recap: Calculating Hypothesis Probabilities

• Probabilistic formulation
  – It is straightforward to enumerate all possible assignments.
  – However, we also need to calculate the probability of each child hypothesis.
  – This is done recursively:

\[
p(\mathbf{Z}_j^{(k)} | \mathbf{Y}^{(k)}) = p(\mathbf{Z}_j^{(k)} | \mathbf{Y}^{(k-1)}) \cdot p(\mathbf{Z}_j^{(k-1)})
\]

Recap: Measurement Likelihood

• Use KF prediction
  – Assume that a measurement \(\mathbf{y}_i^{(k)}\) associated to a track \(\mathbf{x}_j\) has a Gaussian pdf centered around the measurement prediction \(\mathbf{\hat{y}}_j^{(k)}\) with innovation covariance \(\mathbf{S}_j^{(k)}\).
  – Further assume that the pdf of a measurement belonging to a new track or false alarm is uniform in the observation volume \(W\) (the sensor’s field-of-view) with probability \(W^{-1}\).

  Thus, the measurement likelihood can be expressed as

\[
p(\mathbf{y}_i^{(k)} | \mathbf{Z}_j^{(k)}, \mathbf{Y}^{(k-1)}) = \prod_{i=1}^{M} \mathcal{N}(\mathbf{y}_i^{(k)}; \mathbf{\hat{y}}_j^{(k)}, \mathbf{S}_j^{(k)}) \cdot W^{-(1-A)}
\]

Recap: Probability of an Assignment Set

\[
p(\mathbf{Z}_p^{(k)} | \mathbf{Y}^{(k)})
\]

• Composed of three terms
  1. Probability of the number of tracks \(N_{det}, N_{fal}, N_{new}\)
     • Assumption 1: \(N_{det}\) follows a Binomial distribution
     \[p(N_{det} | \mathbf{Z}_p^{(k-1)}) = \binom{N}{N_{det}} p_{det}^{N_{det}} (1 - p_{det})^{N - N_{det}}\]
     where \(N\) is the number of tracks in the parent hypothesis
     • Assumption 2: \(N_{fal}\) and \(N_{new}\) both follow a Poisson distribution with expected number of events \(\lambda_{fal}W\) and \(\lambda_{new}W\)

\[
p(N_{det}, N_{fal}, N_{new} | \mathbf{Z}_p^{(k-1)}) = \binom{N}{N_{det}} p_{det}^{N_{det}} (1 - p_{det})^{N - N_{det}} \cdot p(N_{fal}; \lambda_{fal}W) \cdot p(N_{new}; \lambda_{new}W)
\]
Recap: Probability of an Assignment Set

2. Probability of a specific assignment of measurements
   • Such that \( M_k = N_{\text{det}} + N_{\text{fal}} + N_{\text{new}} \) holds.
   • This is determined as \( \frac{1}{N_{\text{det}}} \), where the number of combinations is:
     \[
     \binom{M_k}{N_{\text{det}}} \binom{M_k - N_{\text{det}}}{N_{\text{fal}}} \binom{M_k - N_{\text{det}} - N_{\text{fal}}}{N_{\text{new}}}
     \]

3. Probability of a specific assignment of tracks
   • Given that a track can be either detected or not detected.
   • This is determined as \( \frac{1}{N_{\text{det}}} \), where the number of assignments is:
     \[
     \frac{N!}{(N - N_{\text{det}})!} \frac{N - N_{\text{det}}}{N_{\text{det}}}
     \]

⇒ When combining the different parts, many terms cancel out!

Recap: Linear Assignment Formulation

• Form a matrix of pairwise similarity scores
• Example: Similarity based on motion prediction
  - Predict motion for each trajectory and assign scores for each measurement based on inverse (Mahalanobis) distance, such that closer measurements get higher scores.

Recap: Linear Assignment Problem

• Formal definition
  - Maximize \( \sum_{i,j} a_{ij} z_{ij} \)
  - Subject to \( \sum_{j=1}^{M} z_{ij} = 1 \), \( i = 1, 2, \ldots, N \)
  - \( \sum_{i=1}^{N} z_{ij} = 1 \), \( j = 1, 2, \ldots, M \)
  - \( z_{ij} \in \{0, 1\} \)

⇒ The permutation matrix constraint ensures we can only match up one object from each row and column.

Recap: Optimal Solution

• Greedy Algorithm
  - Easy to program, quick to run, and yields “pretty good” solutions in practice.
  - But it often does not yield the optimal solution.

• Hungarian Algorithm
  - There is an algorithm called Kuhn-Munkres or “Hungarian” algorithm specifically developed to efficiently solve the linear assignment problem.
  - Reduces assignment problem to bipartite graph matching.
  - When starting from an \( N \times N \) matrix, it runs in \( O(N^3) \).

⇒ If you need LAP, you should use it.

Recap: Min-Cost Flow

• Conversion into flow graph
  - Transform weights into costs \( c_{ij} = a - b_{ij} \)
  - Add source/sink nodes with 0 cost.
  - Directed edges with a capacity of 1.

Course Outline

• Single-Object Tracking
• Bayesian Filtering
  - Kalman Filters, EKF
  - Particle Filters
• Multi-Object Tracking
  - Introduction
  - MHT
  - Network Flow Optimization
• Visual Odometry
• Visual SLAM & 3D Reconstruction
• Deep Learning for Video Analysis
Recap: Min-Cost Flow

- Conversion into flow graph
  - Pump \( N \) units of flow from source to sink.
  - Internal nodes pass on flow (\( \sum \) flow in = \( \sum \) flow out).
  \( \Rightarrow \) Find the optimal paths along which to ship the flow.

Recap: Using Network Flow for Tracking

- Complication 1
  - Tracks can start later than frame1 (and end earlier than frame4)
  \( \Rightarrow \) Connect the source and sink nodes to all intermediate nodes.

Recap: Network Flow Approach

Solution: Divide each detection into 2 nodes.

Recap: Min-Cost Formulation

- Objective Function
  \[ T^* = \arg\min_T \sum_i C_{m,i} f_{m,i} + \sum_i C_{i,out} f_{i,out} \]
  \[ + \sum_{i,j} C_{i,j} f_{i,j} \]

- subject to
  - Flow conservation at all nodes
    \[ f_{m,i} + \sum_j f_{j,i} = f_{i,out} + \sum_j f_{i,j} \forall i \]
  - Edge capacities
    \[ f_{i,j} \leq 1 \]
Min-Cost Formulation

- Objective Function
  $$T^* = \arg\min_T \sum_i C_{in,i}f_{in,i} + \sum_i C_{out,i}f_{out,i} + \sum_i C_{trans,i}f_{trans,i}$$
- Equivalent to Maximum A-Posteriori formulation
  $$T^* = \arg\max_T \prod_{T_e, T} P(T_e|T)$$

Course Outline

- Single-Object Tracking
- Bayesian Filtering
- Multi-Object Tracking
  - Introduction
  - MHT
- Network Flow Optimization
- Visual Odometry
  - Sparse interest-point based methods
  - Dense direct methods
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis

Recap: What is Visual Odometry?

Visual odometry (VO)...
- ... is a variant of tracking
  - Track motion (position and orientation) of the camera from its images
  - Only considers a limited set of recent images for real-time constraints
- ... also involves a data association problem
  - Motion is estimated from corresponding interest points or pixels in images, or by correspondences towards a local 3D reconstruction

Recap: Direct vs. Indirect Methods

- Direct methods
  - Formulate alignment objective in terms of photometric error (e.g., intensities)
    $$p(Y_j | Y_i, \xi) \rightarrow E(\xi) = \int p(\xi) | I_j(a) - I_i(\omega(a, \xi)) | da$$
- Indirect methods
  - Formulate alignment objective in terms of reprojection error of geometric primitives (e.g., points, lines)
    $$p(Y_j | Y_i, \xi) \rightarrow E(\xi) = \sum |y_{j,i} - \omega(y_{j,i}, \xi)|$$

Recap: Point-based Visual Odometry Pipeline

- Keypoint detection and local description (CV I)
- Robust keypoint matching (CV I)
- Motion estimation
  - 2D-to-2D: motion from 2D point correspondences
  - 2D-to-3D: motion from 2D points to local 3D map
  - 3D-to-3D: motion from 3D point correspondences (e.g., stereo, RGB-D)

Recap: Motion Estimation from Point Correspondences

- 2D-to-2D
  - Reproj. error:
    $$E(\mathbf{T}_{ji}) = \sum |y_{j,i} - \omega(T_{ji}y_{j,i})|^2$$
    - Introduced linear algorithm: 8-point
- 2D-to-3D
  - Reprojection error:
    $$E(\mathbf{T}_{ji}) = \sum |y_{j,i} - \omega(T_{ji}y_{j,i})|^2$$
    - Introduced linear algorithm: DLT PnP
- 3D-to-3D
  - Reprojection error:
    $$E(\mathbf{T}_{ji}) = \sum |y_{j,i} - \omega(T_{ji}y_{j,i})|^2$$
    - Introduced linear algorithm: Arun’s method
Recap: Eight-Point Algorithm for Essential Matrix Est.

- First proposed by Longuet and Higgins, 1981
- Algorithm:
  1. Rewrite epipolar constraints as a linear system of equations
     \[ y_i^T y_j = a_i^T E a_j = 0 \]
     \[ \Rightarrow A E = 0 \]
     \[ A = (a_1, \ldots, a_n)^T \]
     using Kronecker product \( a_i = y_i \otimes y_j \) and \( E = (e_1, e_2, \ldots, e_n)^T \)
  2. Apply singular value decomposition (SVD) on \( A \) and unstack the 9th column of \( V_x \) into \( E \).
  3. Project the approximate \( E \) into the (normalized) essential space: Determine the SVD of \( E = U \text{diag}(c_1, c_2, c_3) V^T \) with \( U, V \in \text{SO}(3) \) and replace the singular values \( c_2 \geq c_3 \geq c_1 \) with \( 1, 0, 0 \) to find
     \[ E = U \text{diag}(1, 1, 0) V^T \]

Algorithm: 2D-to-2D Visual Odometry

**Input:** image sequence \( I_{1:n} \)

**Output:** aggregated camera poses \( T_{1:n} \)

**Algorithm:**
1. For each current image \( I_k \):
   1. Extract and match keypoints between \( I_{k-1} \) and \( I_k \)
   2. Compute relative pose \( T_k^{k-1} \) from essential matrix between \( I_{k-1}, I_k \)
   3. Compute relative scale and rescale translation of \( T_k^{k-1} \) accordingly
   4. Aggregate camera pose by \( T_{k} = T_{k-1} T_k^{k-1} \)

Recap: Eight-Point Algorithm cont.

- Normalized essential matrix: \( \|E\| = \|c\| = 1 \)
- Linear algorithms exist that require only 6 points for general motion
- Non-linear 5-point algorithm with up to 10 (possibly complex) solutions
- Points need to be in „general position": certain degenerate configurations exists (e.g., all points on a plane)
- No translation, ideally: \( \|c\| = 0 \Rightarrow \|E\| = 0 \)
- But: for small translations, signal-to-noise ratio of image parallax may be problematic: „spurious“ pose estimate

Recap: Relative Scale Recovery

- Problem:
  - Each subsequent frame-pair gives another solution for the reconstruction scale
- Solution:
  - Triangulate overlapping points \( Y_{i-2}, Y_{i-1}, Y_i \) for current and last frame pair 
    \[ X_{i-2,i-1}, X_{i-1,i} \]
  - Rescale translation of current relative pose estimate to match the reconstruction scale with the distance ratio between corresponding point pairs
    \[ r_{ij} = \frac{|X_{i-2,i-1}|}{|X_{i-1,i}|} \]
  - Use mean or robust median over available pair ratios

Recap: Triangulation

- Goal: Reconstruct 3D point \( \hat{x} = (x, y, z, w)^T \in \mathbb{R}^4 \) from 2D image observations \( \{y_1, \ldots, y_N\} \) for known camera poses \( \{T_1, \ldots, T_N\} \)
  - Linear solution: Find 3D point such that reprojections equal its projections
    \[ y_i = \pi(T_i \hat{x}) \]
  - Each image provides one constraint \( y_i - y'_i = 0 \)
  - Leads to system of linear equations \( A \hat{x} = 0 \), two approaches:
    - Set \( -1 \) and solve nonhomogeneous system
    - Find nullspace of \( A \) using SVD (this is what we did in CV I)
  - Non-linear solution: Minimize least squares reprojection error (more accurate)
    \[ \min \hat{x} \left\{ \sum \|y_i - y'_i\|^2 \right\} \]
Recap: Direct Linear Transform for PnP

- Goal: determine projection matrix \( \mathbf{P} = (\mathbf{R} \mathbf{t}) \in \mathbb{R}^{3 \times 4} = (\mathbf{p}_1 \mathbf{p}_2 \mathbf{p}_3) \)
- Each 2D-to-3D point correspondence
  \( 3D: \mathbf{x} = (x, y, z, w) \in \mathbb{P}^3 \quad 2D: \mathbf{y} = (u', v', w') \in \mathbb{P}^2 \)
gives two constraints
  \[
  \begin{bmatrix}
  0 \\
  -w \mathbf{x}_2^T \\
  -w \mathbf{x}_3^T
  \end{bmatrix}
  \begin{bmatrix}
  \mathbf{p}_1^T \\
  \mathbf{p}_2^T \\
  \mathbf{p}_3^T
  \end{bmatrix} = 0
  \]
  through \( \mathbf{y} \times (\mathbf{p}_k) \neq 0 \)
- Form linear system of equation \( \mathbf{A} \mathbf{p} = 0 \) with \( \mathbf{p} := (\mathbf{p}_1^T \mathbf{p}_2^T \mathbf{p}_3^T) \in \mathbb{R}^3 \) from \( N \geq 6 \) correspondences
- Solve for \( \mathbf{p} \): determine unit singular vector of \( \mathbf{A} \) corresponding to its smallest eigenvalue

Recap: 3D Rigid-Body Motion from 3D-to-3D Matches

- [Arun et al., Least-squares fitting of two 3-d point sets, IEEE PAMI, 1987]
- Corresponding 3D points, \( N \geq 3 \)
  \( \mathbf{X}_{k-1} = \{ \mathbf{x}_{k-1,1}, \ldots, \mathbf{x}_{k-1,N} \} \quad \mathbf{X}_k = \{ \mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,N} \} \)
- Determine means of 3D point sets
  \[
  \mu_{k-1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_{k-1,i} \quad \mu_k = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_{k,i}
  \]
- Determine rotation from
  \[
  \mathbf{A} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_{k-1,i} - \mu_{k-1}) (\mathbf{x}_{k,i} - \mu_k)^T \quad \mathbf{A} = \mathbf{USV}^T \quad \mathbf{R}_{k-1} = \mathbf{VU}^T
  \]
- Determine translation as
  \[
  \mathbf{t}_{k-1} = \mu_k - \mathbf{R}_{k-1} \mu_{k-1}
  \]

Recap: 2D-to-3D Visual Odometry

- Input: sequence \( I_{\alpha}, I_{\beta} \)
- Output: aggregated camera poses \( T_{0\tau} \)
- Algorithm:
  1. Extract and match keypoints between \( I_{\alpha} \) and \( I_{\beta} \)
  2. Determine camera pose (Essential matrix) and triangulate 3D keypoints \( \mathbf{X}_k \)
  3. For each current image \( I_k \):
     1. Extract and match keypoints between \( I_k \) and \( I_{\beta} \)
     2. Compute camera pose \( T_k \) using PnP from 2D-to-3D matches
     3. Triangulate all new keypoint matches between \( I_{k-1} \) and \( I_k \) and add them to the local map \( \mathbf{X}_k \)

Recap: 3D-to-3D Stereo Visual Odometry

- Input: stereo image sequence \( I_{\alpha}, I_{\beta} \)
- Output: aggregated camera poses \( T_{0\tau} \)
- Algorithm:
  For each current stereo image \( I_{\alpha}, I_{\beta} \):
  1. Extract and match keypoints between \( I_{\alpha} \) and \( I_{\beta} \)
  2. Triangulate 3D points \( \mathbf{X}_k \) between \( I_{\alpha} \) and \( I_{\beta} \)
  3. Compute camera pose \( T_{k-1} \) from 3D-to-3D point matches \( \mathbf{X}_k \) to \( \mathbf{X}_{k-1} \)
  4. Aggregate camera poses by \( T_k = T_{k-1} T_{k-1}^{-1} \)

Recap: Keypoint Detectors

- Corners
  - Image locations with locally prominent intensity variation
  - Intersections of edges
- Examples: Harris, FAST
- Scale-selection: Harris-Laplace
- Blobs
  - Image regions that stick out from their surrounding in intensity/texture
  - Circular high-contrast regions
- E.g.: LoG, DoG (SIFT), SURF
- Scale-space extrema in LoG/DoG

Recap: RANSAC

- \textbf{RANdom SAmple Consensus} algorithm for robust estimation
- Algorithm:
  Input: data \( \mathcal{D} \), required data points for fitting, success probability \( p \), outlier ratio \( \zeta \)
  Output: inlier set
  1. Compute required number of iterations \( N = \frac{\log(1 - \zeta)}{\log(1 - (1 - p)^3)} \)
  2. For \( N \) iterations do:
     1. Randomly select a subset of \( k \) data points
     2. Fit model on the subset
     3. Count inliers and keep model/subset with largest number of inliers
  3. Refit model using found inlier set
Course Outline

- Single-Object Tracking
- Bayesian Filtering
- Multi-Object Tracking
  - Introduction
  - MHT, (JPDAF)
  - Network Flow Optimization
- Visual Odometry
  - Sparse interest-point based methods
  - Dense direct methods
- Visual SLAM & 3D Reconstruction
- Deep Learning for Video Analysis

Recap: Direct Visual Odometry Pipeline

- Avoid manually designed keypoint detection and matching
- Instead: direct image alignment
- Warping requires depth
  - RGB-D
  - Fixed-baseline stereo
  - Temporal stereo, tracking and (local) mapping

Recap: Direct Image Alignment Principle

- Idea
  - If we know the pixel depth, we can “simulate” an image from a different viewpoint
  - Ideally, the warped image is the same as the image taken from that pose:
    \[ I_1(y) = I_2(\pi(T(x)Z(y,y))) \]
  - Estimate the warp by minimizing the residuals (similar to LK alignment)

Recap: Representing Motion using Lie Algebra se(3)

- \( \text{se}(3) \) is a smooth manifold, i.e. a Lie group
- Its Lie algebra \( \text{se}(3) \) provides an elegant way to parametrize poses for optimization
- Its elements \( \xi \in \text{se}(3) \) form the tangent space of \( \text{SE}(3) \) at identity
- The \( \text{se}(3) \) elements can be interpreted as rotational and translational velocities (twists)

Recap: Exponential Map of \( \text{SE}(3) \)

- The exponential map finds the transformation matrix for a twist:
  \[ \exp(\xi) = \begin{pmatrix} \exp(\omega) & Av \ 0 & 1 \end{pmatrix} \]

Recap: Logarithm Map of \( \text{SE}(3) \)

- The logarithm maps twists to transformation matrices:
  \[ \log(T) = \begin{pmatrix} \log(R) & A^{-t} \ 0 & 0 \end{pmatrix} \]
  \[ \omega = \cos^{-1}\left(\frac{\text{tr}(R) - 1}{2}\right) \]
Recap: Working with Twist Coordinates

- Let's define the following notation:
  - Inversion of hat operator: \(\hat{T} = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}\)
  - Conversion: \(\xi(\hat{T}) = \log(\hat{T})\)
  - Pose inversion: \(\xi^{-1} = \log(T(\xi^{-1}))\)
  - Pose concatenation: \(\xi \otimes \xi = \log(T(\xi)T(\xi))\)
  - Pose difference: \(\xi = \log(T(\xi)T^{-1}(\xi))\)

Recap: Optimization with Twist Coordinates

- Twists provide a minimal local representation without singularities
- Since \(\mathbb{SE}(3)\) is a smooth manifold, we can decompose transformations in each optimization step into the transformation itself and an infinitesimal increment
  \(T(\xi) = T(\xi)exp(\delta\xi)\)
- We can then optimize an energy function \(E(\xi, \delta\xi)\) in order to estimate the pose increment \(\delta\xi\), e.g., using Gradient descent

Algorithm: Direct RGB-D Visual Odometry

Input: RGB-D image sequence \(I_{t_0}, Z_{t_0}\)
Output: aggregated camera poses \(T_{t_1}\)

Algorithm:
For each current RGB-D image \(I_t, Z_t\):
1. Estimate relative camera motion \(T_{t-1}^{-1}\) towards the previous RGB-D frame using direct image alignment
2. Concatenate estimated camera motion with previous frame camera pose to obtain current camera pose estimate \(T_t = T_{t-1} T_{t-1}^{-1}\)

Recap: Definition of Visual SLAM

- Visual SLAM:
  - The process of simultaneously estimating the egomotion of an object and the environment map using only inputs from visual sensors on the object
- Inputs: images at discrete time steps \(t\),
  - Monocular case: Set of images \(I_t = \{I_1, \ldots, I_t\}\)
  - Stereo case: Left/right images \(I_t = \{I_t^L, I_t^R\}, \hat{K}_t = \{K_t^L, K_t^R\}\)
  - RGB-D case: Color/depth images \(I_t = \{I_t, Z_t\}, \hat{K}_t = \{K_t, Z_t\}\)
  - Robotics: control inputs \(U_t, Z_t\)
- Output:
  - Camera pose estimates \(T_t \in \mathbb{SE}(3)\) in world reference frame.
  - For convenience, we also write \(\xi_t = \xi(T_t)\)
  - Environment map \(M\)

Recap: Map Observations in Visual SLAM

With \(Y_t\) we denote observations of the environment map in image \(I_t\), e.g.,
- Indirect point based method: \(Y_t = \{y_{1t}, \ldots, y_{Nt}\}\) (2D or 3D image points)
- Direct RGB-D method: \(Y_t = \{y_{1t}, \hat{z}_{1t}\}\) (all image pixels)
- ...
- Involves data association to map elements \(M = \{m_1, \ldots, m_S\}\)
  - We denote correspondences by \(\hat{y}_{ij} = j, 1 \leq i \leq N, 1 \leq j \leq S\)

Course Outline

- Single-Object Tracking
- Bayesian Filtering
- Multi-Object Tracking
- Visual Odometry
  - Sparse interest-point based methods
  - Dense direct methods
- Visual SLAM & 3D Reconstruction
  - Online SLAM methods
  - Full SLAM methods
- Deep Learning for Video Analysis
Recap: Probabilistic Formulation of Visual SLAM

- SLAM posterior probability: \( p(\xi_{a}, M | Y_{a}, U_{a}) \)
- Observation likelihood: \( p(Y_{i} | \xi_{i}, M) \)
- State-transition probability: \( p(\xi_{i} | \xi_{i-1}, U_{i}) \)

Recap: Online SLAM Methods

- Marginalize out previous poses:
  \[
  p(\xi_{i}, M | Y_{0:i}, U_{0:i}) = \int \ldots \int p(\xi_{i}, M | Y_{0:i}, U_{0:i}) d\xi_{0} \ldots d\xi_{i-1}
  \]
- Poses can be marginalized individually in a recursive way
- Variants:
  - Tracking-and-Mapping: Alternating pose and map estimation
  - Probabilistic filters, e.g., EKF-SLAM

Recap: EKF SLAM

- Detected keypoint \( y_{i} \) in an image observes ‘landmark’ position \( m_{j} \) in the map \( M = \{m_{1}, \ldots, m_{S}\} \).
- Idea: Include landmarks into state variable

Recap: 2D EKF-SLAM State-Transition Model

- State/control variables
  \[
  \xi_{i} = \begin{pmatrix} x_{i} \ y_{i} \ \theta_{i} \end{pmatrix}^{T}, \quad m_{i} = \begin{pmatrix} m_{i,1} \ m_{i,2} \ m_{i,3} \end{pmatrix}^{T}
  \]
- State-transition model
  - Pose:
    \[
    p(\xi_{i} | \xi_{i-1}, u_{i}) = \mathcal{N}(0, \Sigma_{\xi_{i}})
    \]
  - Landmarks:
    \[
    p(m_{i} | \xi_{i-1}, u_{i}) = \mathcal{N}(0, \Sigma_{m_{i}})
    \]
  - Combined:
    \[
    x_{i} = \mathcal{N}(\mu_{x,i}, \Sigma_{x_{i}}), \quad \Sigma_{x_{i}} = \begin{pmatrix} \Sigma_{m_{i}} & 0 \\ 0 & a \end{pmatrix}
    \]

Recap: 2D EKF-SLAM Observation Model

- State/measurement variables
  \[
  y_{i} = \begin{pmatrix} x_{i} \ y_{i} \ \theta_{i} \end{pmatrix}^{T}, \quad m_{i} = \begin{pmatrix} m_{i,1} \ m_{i,2} \ m_{i,3} \end{pmatrix}^{T}
  \]
- Observation model:
  \[
  y_{i} = (x_{i}, y_{i})^{T}, \quad \theta_{i}, \quad m_{i} = \begin{pmatrix} m_{i,1} \ m_{i,2} \ m_{i,3} \end{pmatrix}^{T}
  \]
  \[
  m_{i,3}^{c} = \mathcal{N}(0, \Sigma_{m_{i}})
  \]

Recap: State Initialization

- First frame:
  - Anchor reference frame at initial pose
  - Set pose covariance to zero
  \[
  x_{0} = 0, \quad \Sigma_{x_{0}} = 0
  \]
- New landmark:
  - Initial position unknown
  - Initialize mean at zero
  - Initialize covariance to infinity (large value)
  \[
  \Sigma_{m_{i}} = \infty I
  \]
Recap: Full SLAM Approaches

- SLAM graph optimization:
  - Joint optimization for poses and map elements from image observations of map elements and control inputs

- Pose graph optimization:
  - Optimization of poses from relative pose constraints deduced from the image observations
  - Map recovered from the optimized poses

Course Outline

- Single-Object Tracking
- Bayesian Filtering
- Multi-Object Tracking
- Visual Odometry
  - Sparse interest-point based methods
  - Dense direct methods
- Visual SLAM & 3D Reconstruction
  - Online SLAM methods
  - Full SLAM methods
- Deep Learning for Video Analysis
  - CNNs for video analysis
  - CNNs for motion estimation
  - Video object segmentation

Recap: Long Short-Term Memory (LSTM)

- LSTMs
  - Inspired by the design of memory cells
  - Each module has 4 layers, interacting in a special way
  - Effect: LSTMs can learn longer dependencies (~100 steps) than RNNs
Recap: Image Tagging

- Simple combination of CNN and RNN
  - Use CNN to define initial state $h_0$ of an RNN.
  - Use RNN to produce text description of the image.

Recap: Video to Text Description

Course Outline

- Single-Object Tracking
- Bayesian Filtering
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction
  - Online SLAM methods
  - Full SLAM methods
- Deep Learning for Video Analysis
  - CNNs for video analysis
  - CNNs for motion estimation
  - Video object segmentation

Recap: Learning Similarity Functions

- Siamese Network
  - Present the two stimuli to two identical copies of a network (with shared parameters)
  - Train them to output similar values if the inputs are (semantically) similar.
- Used for many matching tasks
  - Face identification
  - Stereo estimation
  - Optical flow
  - ...

Recap: Metric Learning – Contrastive Loss

- Mapping an image to a metric embedding space
  - Metric space: distance relationship = class membership
  
  $\|f(x) - f(x_+)\| \rightarrow 0$

  $\|f(x) - f(x_-)\| \geq m$

  Yi et al., LIFT: Learned Invariant Feature Transform, ECCV 16

Recap: Metric Learning – Triplet Loss

- Learning a discriminative embedding
  - Present the network with triplets of examples

  - Apply triplet loss to learn an embedding $f(\cdot)$ that groups the positive example closer to the anchor than the negative one.

  $\|f(x^a_i) - f(x^p_i)\|^2 < \|f(x^a_i) - f(x^n_i)\|^2$

  ⇒ Used
Recap: FlowNet – FlowNetSimple Design

• Simple initial design
  – Simply stack two sequential images together and feed them through the network
  – In order to compute flow, the network has to compare image patches
  – But it has to figure out on its own how to do that…


• Correlation network
  – Central idea: compute a correlation score between two feature maps
    \[ c(x_1, x_2) = \sum_{\alpha} f_1(x_1 + \alpha) \cdot f_2(x_2 + \alpha) \]
  – Then refine the correlation scores and turn them into flow predictions

Recap: FlowNet – Flow Refinement

• Flow refinement stage (both network designs)
  – After series of conv and pooling layers, the resolution has been reduced
  – Refine the coarse pooled representation by upconvolution layers (unpooling + upconvolution)
  – Skip connections to preserve high-res information from early layers

Recap: FlowNet 2.0 Improved Design

• Stacked architecture
  – Several instances of FlowNetC and FlowNetS stacked together to estimate large-displacement flow
  – Sub-network specialized on small motions
  – Fusion layer

Course Outline

• Single-Object Tracking
• Bayesian Filtering
• Multi-Object Tracking
• Visual Odometry
• Visual SLAM & 3D Reconstruction
  – Online SLAM methods
  – Full SLAM methods
• Deep Learning for Video Analysis
  – CNNs for video analysis
  – CNNs for motion estimation
  – Video object segmentation

Recap: Video Object Segmentation

• Object Detection
• Object Tracking
• Object Segmentation
• Video Object Segmentation
Any More Questions?

Good luck for the exam!