Computer Vision 2
WS 2018/19

Part 20 – Repetition
23.01.2019

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:

  \[ \mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

  – Test if a newly observed pixel value has a high likelihood under this Gaussian model.

  \[ \Rightarrow \text{Automatic estimation of a sensitivity threshold for each pixel.} \]
Recap: Stauffer-Grimson Background Model

• Idea
  – Model the distribution of each pixel by a mixture of $K$ Gaussians
    \[
    p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)
    \]
    where \( \Sigma_k = \sigma_k^2 I \)
  – 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
    \[
    B = \arg \min_b \left( \sum_{k=1}^{b} \frac{w_k}{\sigma_k} > T \right)
    \]

[C. Stauffer, W.E.L. Grimson, CVPR'99]
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 $M_{k,t} = 1$ iff component $k$ is the model that matched, else 0.
    $$\pi_k^{(t+1)} = (1 - \alpha)\pi_k^{(t)} + \alpha M_{k,t}$$
    $$\mu_k^{(t+1)} = (1 - \rho)\mu_k^{(t)} + \rho x^{(t+1)}$$
    $$\Sigma_k^{(t+1)} = (1 - \rho)\Sigma_k^{(t)} + \rho(x^{(t+1)} - \mu_k^{(t+1)})(x^{(t+1)} - \mu_k^{(t+1)})^T$$
  – Adapt only the parameters for the matching component
    where
    $$\rho = \alpha \mathcal{N}(x_n | \mu_k, \Sigma_k)$$
    (i.e., the update is weighted by the component likelihood)
Recap: Kernel Background Modeling

- Nonparametric density estimation
  - Estimate a pixel’s background distribution using the kernel density estimator $K(\cdot)$ as
    \[ p(x^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} K(x^{(t)} - x^{(i)}) \]
  - Choose $K$ to be a Gaussian $\mathcal{N}(0, \Sigma)$ with $\Sigma = \text{diag}\{\sigma_j\}$. Then
    \[ p(x^{(t)}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi \sigma_j^2}} e^{-\frac{1}{2} \frac{(x_j^{(t)} - x_j^{(i)})^2}{\sigma_j^2}} \]
  - 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

Image source: Robert Collins
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:

$$
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} =
- \begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
$$

$$A \quad d = b$$

25x2 2x1 25x1

• Minimum least squares solution given by solution of

$$
(A^T A) \quad d = A^T b
$$

$$
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} =
- \begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\begin{bmatrix}AT A \\
AT b
\end{bmatrix}$$

Recall the Harris detector!
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.
Recap: Coarse-to-fine Optical Flow Estimation

Gaussian pyramid of image 1

\[ u = 10 \text{ pixels} \]
\[ u = 5 \text{ pixels} \]
\[ u = 2.5 \text{ pixels} \]
\[ u = 1.25 \text{ pixels} \]

Gaussian pyramid of image 2

\[ u = 10 \text{ pixels} \]
Recap: Coarse-to-fine Optical Flow Estimation

Gaussian pyramid of image 1

Run iterative LK

Warp & upsample

Run iterative LK

Gaussian pyramid of image 2

Image 1

Image 2

Slide credit: Steve Seitz
Recap: Shi-Tomasi Feature Tracker (→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., 5×5 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: General LK Image Registration

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

• **LK formulation**
  - Formulate this as an optimization problem
    \[
    \arg\min_p \sum_x [I(W(x;p)) - T(x)]^2
    \]
  - We assume that an initial estimate of $p$ is known and iteratively solve for increments to the parameters $\Delta p$:
    \[
    \arg\min_{\Delta p} \sum_x [I(W(x;p + \Delta p)) - T(x)]^2
    \]
Recap: Step-by-Step Derivation

- Key to the derivation
  - Taylor expansion around $\Delta p$

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

\[
= I(W([x, y]; p_1, \ldots, p_n))
\]

\[
+ \left[ \begin{array}{cc}
\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}
\end{array} \right] \left[ \begin{array}{cccc}
\frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\
\frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n}
\end{array} \right] \left[ \begin{array}{c}
\Delta p_1 \\
\Delta p_2 \\
\vdots \\
\Delta p_n
\end{array} \right]
\]

Gradient Jacobian Increment parameters to solve for $\Delta p$
Recap: Inverse Compositional LK Algorithm

• Iterate
  – Warp \(I\) to obtain \(I(\mathbf{W}([x, y]; \mathbf{p}))\)
  – Compute the error image \(T([x, y]) = I(\mathbf{W}([x, y]; \mathbf{p}))\)
  – Warp the gradient \(\nabla I\) with \(\mathbf{W}([x, y]; \mathbf{p})\)
  – Evaluate \(\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\) at \(([x, y]; \mathbf{p})\) (Jacobian)
  – Compute steepest descent images \(\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\)
  – Compute Hessian matrix \(\mathbf{H} = \sum_x \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]\)
  – Compute \(\sum_x \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[T([x, y]) - I(\mathbf{W}([x, y]; \mathbf{p}))\right]\)
  – Compute \(\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_x \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[T([x, y]) - I(\mathbf{W}([x, y]; \mathbf{p}))\right]\)
  – Update the parameters \(\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}\)

• Until \(\Delta \mathbf{p}\) magnitude is negligible
Recap: Inverse Compositional LK Algorithm

[S. Baker, I. Matthews, IJCV'04]
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: Tracking as Online Classification

• Tracking as binary classification problem

object vs. background

Image source: Disney/Pixar

Slide credit: Helmut Grabner
Recap: Tracking as Online Classification

- Tracking as binary classification problem

  - Handle object and background changes by online updating

• Main idea
  – Iteratively select an ensemble of classifiers
  – Reweight misclassified training examples after each iteration to focus training on difficult cases.

• Components
  – $h_m(x)$: “weak” or base classifier
    - Condition: <50% training error over any distribution
  – $H(x)$: “strong” or final classifier

• AdaBoost:
  – Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:
    $$H(x) = sign \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)$$
Recap: AdaBoost – Algorithm

1. Initialization: Set \( w_n^{(1)} = \frac{1}{N} \) 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
      \[
      J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)
      \]
   b) Estimate the weighted error of this classifier on \( X \):
      \[
      \epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}
      \]
   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:
      \[
      w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(h_m(x_n) \neq t_n) \}
      \]
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 \( N \rightarrow \infty \) iterations.

Recap: From Offline to Online Boosting

**off-line**

*Given:*
- set of labeled training samples
  \[ \mathcal{X} = \{ \langle x_1, y_1 \rangle, \ldots, \langle x_L, y_L \rangle \mid y_i \pm 1 \} \]
- weight distribution over them
  \[ D_0 = 1/L \]

for \( n = 1 \) to \( N \)
- train a weak classifier using samples and weight dist.
  \[ h_{n}^{weak}(x) = \mathcal{L}(\mathcal{X}, D_{n-1}) \]
- calculate error \( e_n \)
- calculate weight \( \alpha_n = f(e_n) \)
- update weight dist. \( D_n \)

next

\[ h^{strong}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_{n}^{weak}(x)) \]

**on-line**

*Given:*
- ONE labeled training sample
  \[ \langle x, y \rangle \mid y \pm 1 \]
- strong classifier to update
- initial importance \( \lambda = 1 \)

for \( n = 1 \) to \( N \)
- update the weak classifier using samples and importance
  \[ h_{n}^{weak}(x) = \mathcal{L}(h_{n}^{weak}, \langle x, y \rangle, \lambda) \]
- update error estimation \( \tilde{e}_n \)
- update weight \( \alpha_n = f(\tilde{e}_n) \)
- update importance weight \( \lambda \)

next

\[ h^{strong}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_{n}^{weak}(x)) \]
Recap: Online Boosting for Feature Selection

• Introducing “Selector”
  – Selects one feature from its local feature pool

\[
\mathcal{H}^{weak} = \{h_1^{weak}, \ldots, h_M^{weak}\} \\
\mathcal{F} = \{f_1, \ldots, f_M\} \\
h_{sel}(x) = h_m^{weak}(x) \\
m = \text{arg min}_i e_i
\]

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

Recap: Direct Feature Selection

- **one training sample**
  - Initial importance: \( \lambda = 1 \)
  - Estimate errors
  - Select best weak classifier
  - Update weight \( \alpha_1 \)
  - Repeat for each training sample

- Global weak classifier pool

- **current strong classifier** \( h_{\text{Strong}} \)
Recap: Tracking by Online Classification

- **Search region**
- **Evaluate classifier on sub-patches**
- **Create confidence map**
- **Analyze map and set new object position**
- **from time $t$ to $t+1$**
- **Update classifier (tracker)**
- **Actual object position**

Slide credit: Helmut Grabner

Image source: Disney/Pixar
Recap: Drifting Due to Self-Learning Policy

⇒ Not only does it drift, it also remains confident about it!
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: 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: 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: Sliding-Window Object Detection

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

Slide credit: Kristen Grauman
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: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations

Object/Non-object

Linear SVM

Collect HOGs over detection window

Contrast normalize over overlapping spatial cells

Weighted vote in spatial & orientation cells

Compute gradients

Gamma compression

Image Window
Recap: Deformable Part-based Model (DPM)

- Multiscale model captures features at two resolutions

Score of filter: dot product of filter with HOG features underneath it

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

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Recap: DPM Hypothesis Score

\[
score(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)
\]

“data term”
“spatial prior”
displacements

deformation parameters

score(\(z\)) = \beta \cdot \Psi(H, z)

concatenation filters and
defformation parameters

concatenation of HOG
features and part
displacement features

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
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.

Recap: Integral Channel Features

- Generalize also block computation
  - 1\textsuperscript{st} order features:
    - Sum of pixels in rectangular region.
  - 2\textsuperscript{nd}-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

Recap: VeryFast Detector

- **Idea 2**: Reduce training time by feature interpolation

  - Shown to be possible for Integral Channel features

50 models, 1 image scale

≈

5 models, 1 image scale
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.

Naming convention:

Slide credit: FeiFei Li, Andrej Karpathy
Recap: Activation Maps

Activation maps

5 × 5 filters

Slide adapted from FeiFei Li, Andrej Karpathy
Recap: Pooling Layers

- **Effect:**
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

![Diagram of a single depth slice with an example of max pooling with 2x2 filters and stride 2](image)

Slide adapted from FeiFei Li, Andrej Karpathy
Recap: R-CNN for Object Detection

- Bbox reg
- SVMs
- ConvNet
- Classify regions with SVMs
- Forward each region through ConvNet
- Warped image regions
- Regions of Interest (RoI) from a proposal method (~2k)

Slide credit: Ross Girshick
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
    \[ \Rightarrow \text{Object detection in a single pass becomes possible.} \]

Slide credit: Ross Girshick
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)
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: 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: $P(X_0)$
  – 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 $t+1$
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:

\[
P(X_t | y_0, \ldots, y_t) = \frac{P(y_t | X_t)P(X_t | y_0, \ldots, y_{t-1})}{\int P(y_t | X_t)P(X_t | y_0, \ldots, y_{t-1})dX_t}
\]

  * Observation model
  * Predicted estimate
Recap: Linear Dynamic Models

• Dynamics model
  – State undergoes linear transformation $D_t$ plus Gaussian noise

  $$x_t \sim N \left( D_t x_{t-1}, \Sigma_d_t \right)$$

• Observation model
  – Measurement is linearly transformed state plus Gaussian noise

  $$y_t \sim N \left( M_t x_t, \Sigma_m_t \right)$$
Recap: Constant Velocity (1D Points)

- State vector: position $p$ and velocity $v$
  \[
  x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix}
  \]
  \[
  p_t = p_{t-1} + (\Delta t)v_{t-1} + \varepsilon
  \]
  \[
  v_t = v_{t-1} + \xi
  \]

- Measurement is position only
  \[
  y_t = Mx_t + \text{noise} = \begin{bmatrix} 1 & 0 \end{bmatrix} x_t + \text{noise}
  \]

Greek letters denote noise terms.
Recap: Constant Acceleration (1D Points)

- State vector: position \( p \), velocity \( v \), and acceleration \( a \).

\[
x_t = \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix}
\]

\[
p_t = p_{t-1} + (\Delta t)v_{t-1} + \frac{1}{2}(\Delta t)^2 a_{t-1} + \varepsilon \\
v_t = v_{t-1} + (\Delta t)a_{t-1} + \xi \\
a_t = a_{t-1} + \zeta
\]

\[
x_t = D_t x_{t-1} + \text{noise} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ a_{t-1} \end{bmatrix} + \text{noise}
\]

- Measurement is position only

\[
y_t = Mx_t + \text{noise} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} + \text{noise}
\]

(greek letters denote noise terms)
Recap: General Motion Models

• Assuming we have differential equations for the motion
  – E.g. for (undampened) periodic motion of a linear spring
    \[
    \frac{d^2 p}{dt^2} = -p
    \]

• Substitute variables to transform this into linear system
  \[
  p_1 = p \
p_2 = \frac{dp}{dt} \
p_3 = \frac{d^2 p}{dt^2}
  \]

• Then we have

\[
x_t = \begin{bmatrix}
p_{1,t} 
p_{2,t} 
p_{3,t}
\end{bmatrix}
\]

\[
p_{1,t} = p_{1,t-1} + (\Delta t) p_{2,t-1} + \frac{1}{2} (\Delta t)^2 p_{3,t-1} + \xi
\]

\[
p_{2,t} = p_{2,t-1} + (\Delta t) p_{3,t-1} + \xi
\]

\[
p_{3,t} = -p_{1,t-1} + \xi
\]
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")

\[ P(X_t | y_0, \ldots, y_{t-1}) \]

Mean and std. dev. of predicted state:

\[ \mu_t^-, \sigma_t^- \]

Measurement update ("Correct")

\[ P(X_t | y_0, \ldots, y_t) \]

Mean and std. dev. of corrected state:

\[ \mu_t^+, \sigma_t^+ \]
Recap: General Kalman Filter (>1dim)

**PREDICT**

\[
x_t^- = D_t x_{t-1}^+
\]

\[
\Sigma_t^- = D_t \Sigma_{t-1}^+ D_t^T + \Sigma_d
\]

**CORRECT**

\[
K_t = \Sigma_t^- M_t^T \left( M_t \Sigma_t^- M_t^T + \Sigma_m \right)^{-1}
\]

\[
x_t^+ = x_t^- + K_t \left( y_t - M_t x_t^- \right)
\]

\[
\Sigma_t^+ = \left( I - K_t M_t \right) \Sigma_t^-
\]

More weight on residual when measurement error covariance approaches 0.

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

for derivations, see F&P Chapter 17.3
Recap: Kalman Filter – Detailed Algorithm

• Algorithm summary
  – Assumption: linear model
    \[ x_t = D_t x_{t-1} + \varepsilon_t \]
    \[ y_t = M_t x_t + \delta_t \]
  – Prediction step
    \[ x_t^- = D_t x_{t-1}^+ \]
    \[ \Sigma_t^- = D_t \Sigma_{t-1}^+ D_t^T + \Sigma_d \]
  – Correction step
    \[ K_t = \Sigma_t^- M_t^T \left( M_t \Sigma_t^- M_t^T + \Sigma_{m_t} \right)^{-1} \]
    \[ x_t^+ = x_t^- + K_t \left( y_t - M_t x_t^- \right) \]
    \[ \Sigma_t^+ = (I - K_t M_t) \Sigma_t^- \]
Extended Kalman Filter (EKF)

- Algorithm summary
  - Nonlinear model
    \[ x_t = g(x_{t-1}) + \varepsilon_t \]
    \[ y_t = h(x_t) + \delta_t \]
  - Prediction step
    \[ x_t^- = g(x_{t-1}^+) \]
    \[ \Sigma_t^- = G_t \Sigma_{t-1}^+ G_t^T + \Sigma_d_t \]
  - Correction step
    \[ K_t = \Sigma_t^- H_t^T (H_t \Sigma_t^- H_t^T + \Sigma_{m_t})^{-1} \]
    \[ x_t^+ = x_t^- + K_t (y_t - h(x_t^-)) \]
    \[ \Sigma_t^+ = (I - K_t H_t) \Sigma_t^- \]

with the Jacobians

\[ G_t = \frac{\partial g(x)}{\partial x} \bigg|_{x=x_{t-1}^+} \]
\[ H_t = \frac{\partial h(x)}{\partial x} \bigg|_{x=x_t^-} \]
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: Propagation of General Densities

Figure from Isard & Blake
Recap: Factored Sampling

• Idea: Represent state distribution non-parametrically
  – Prediction: Sample points from prior density for the state, $P(X)$
  – Correction: Weight the samples according to $P(Y|X)$

$$P(X_t | y_0, \ldots, y_t) = \frac{P(y_t | X_t)P(X_t | y_0, \ldots, y_{t-1})}{\int P(y_t | X_t)P(X_t | y_0, \ldots, y_{t-1})dX_t}$$
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: Monte-Carlo Sampling

- **Objective:**
  - Evaluate expectation of a function $f(z)$ w.r.t. a probability distribution $p(z)$.
  
  $\mathbb{E}[f] = \int f(z)p(z) \, dz$

- **Monte Carlo Sampling idea**
  - Draw $L$ independent samples $z^{(l)}$ with $l = 1, \ldots, L$ from $p(z)$.
  - This allows the expectation to be approximated by a finite sum

  $\hat{f} = \frac{1}{L} \sum_{l=1}^{L} f(z^{(l)})$

  - As long as the samples $z^{(l)}$ are drawn independently from $p(z)$, then

  $\mathbb{E}[\hat{f}] = \mathbb{E}[f]$

  $\Rightarrow$ **Unbiased estimate, independent of the dimension of $z$!**

Image source: C.M. Bishop, 2006
Background: Importance Sampling

• Idea
  – Use a proposal distribution $q(z)$ 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 $\{z^{(l)}\}$ drawn from $q(z)$.

$$
\mathbb{E}[f] = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \\
\approx \frac{1}{L} \sum_{l=1}^{L} \frac{p(z^{(l)})}{q(z^{(l)})}f(z^{(l)})
$$

– with importance weights

$$r_l = \frac{p(z^{(l)})}{q(z^{(l)})}$$
Recap: Sequential Importance Sampling

\[
\text{function } \left\{ \left\{ x_t^i, w_t^i \right\} \right\}^N_{i=1} = SIS \left\{ \left\{ x_{t-1}^i, w_{t-1}^i \right\} \right\}^N_{i=1}, y_t \\
\eta = 0 \\
\text{for } i = 1:N \\
\quad x_t^i \sim q(x_t^i | x_{t-1}^i, y_t) \\
\quad w_t^i = w_{t-1}^i \frac{p(y_t | x_t^i)p(x_t^i | x_t^i)}{q(x_t^i | x_{t-1}^i, y_t)} \\
\quad \eta = \eta + w_t^i \\
\text{end} \\
\text{for } i = 1:N \\
\quad w_t^i = w_t^i / \eta \\
\text{end}
\]

Initialize
Sample from proposal pdf
Update weights
Update norm. factor
Normalize weights

Slide adapted from Michael Rubinstein
Recap: Sequential Importance Sampling

\[ \text{function } \left[ \left\{ x_t^i, w_t^i \right\}_{i=1}^N \right] = SIS \left[ \left\{ x_{t-1}^i, w_{t-1}^i \right\}_{i=1}^N, y_t \right] \]
\[ \eta = 0 \]
\[ \text{for } i = 1:N \]
\[ x_t^i \sim q(x_t^i|x_{t-1}^i, y_t) \]
\[ w_t^i = w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, y_t)} \]
\[ \eta = \eta + w_t^i \]
\[ \text{end} \]
\[ \text{for } i = 1:N \]
\[ w_t^i = \frac{w_t^i}{\eta} \]
\[ \text{end} \]

Initialize
Sample from proposal pdf
Update weights
Update norm. factor
Normalize weights

For a concrete algorithm, we need to define the importance density \( q(\cdot|\cdot) \)!
Recap: SIS Algorithm with Transitional Prior

\[
\text{function } \left[ \left\{ x_t^i, w_t^i \right\}_{i=1}^N \right] = SIS \left[ \left\{ x_{t-1}^i, w_{t-1}^i \right\}_{i=1}^N, y_t \right]
\]

\[\eta = 0\]

\textbf{Initialize}

\textbf{Sample from proposal pdf}

\textbf{Update weights}

\textbf{Update norm. factor}

\[x_t^i \sim p(x_t|x_t^{i-1})\]

\[w_t^i = w_t^{i-1}p(y_t|x_t^i)\]

\[\eta = \eta + w_t^i\]

\[\text{Transitional prior}\]

\[q(x_t|x_t^{i-1}, y_t) = p(x_t|x_t^{i-1})\]

\[w_t^i = w_t^i / \eta\]

\textbf{Normalize weights}

\text{Slide adapted from Michael Rubinstein}
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 \( p(x_t \mid y_{1:t}) \).

• Idea: Resampling
  – Eliminate particles with low importance weights and increase the number of particles with high importance weight.

\[
\left\{ x_t^i, w_t^i \right\}_{i=1}^{N} \rightarrow \left\{ x_t^{i*}, \frac{1}{N} \right\}_{i=1}^{N}
\]

  – The new set is generated by sampling with replacement from the discrete representation of \( p(x_t \mid y_{1:t}) \) such that

\[
Pr \left\{ x_t^{i*} = x_t^j \right\} = w_t^j
\]
Recap: Efficient Resampling Approach

• From Arulampalam paper:

Algorithm 2: Resampling Algorithm
\[
\left\{ \{x^*_k, w^*_k, i^*_k\}_{j=1}^{N_s} \right\} = \text{RESAMPLE} \left\{ \{x_k^i, w_k^i\}_{i=1}^{N_s} \right\}
\]

- Initialize the CDF: \( c_1 = 0 \)
- FOR \( i = 2: N_s \)
  - Construct CDF: \( c_i = c_{i-1} + w_k^i \)
- END FOR
- Start at the bottom of the CDF: \( i = 1 \)
- Draw a starting point: \( u_1 \sim \mathcal{U}[0, N_s^{-1}] \)
- FOR \( j = 1: N_s \)
  - Move along the CDF: \( u_j = u_1 + N_s^{-1}(j - 1) \)
  - WHILE \( u_j > c_i \)
    * \( i = i + 1 \)
  - END WHILE
  - Assign sample: \( x^*_k = x_k^{i^*_k} \)
  - Assign weight: \( w^*_k = N_s^{-1} \)
  - Assign parent: \( i^*_k = i \)
- END FOR

Basic idea: choose one initial small random number; deterministically sample the rest by “crawling” up the cdf. This is \( \mathcal{O}(N) \)!
Recap: Generic Particle Filter

\[
\text{function } \left[ \{x_t^i, w_t^i\}_{i=1}^N \right] = PF \left[ \{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N, y_t \right]
\]

Apply SIS filtering \[
\left[ \{x_t^i, w_t^i\}_{i=1}^N \right] = SIS \left[ \{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N, y_t \right]
\]

Calculate \( N_{\text{eff}} \)

\[
\text{if } N_{\text{eff}} < N_{\text{thr}}
\]

\[
\left[ \{x_t^i, w_t^i\}_{i=1}^N \right] = \text{RESAMPLE} \left[ \{x_t^i, w_t^i\}_{i=1}^N \right]
\]

end

• We can also apply resampling selectively
  – Only resample when it is needed, i.e., \( N_{\text{eff}} \) is too low.
  ⇒ Avoids drift when the tracked state is stationary.
Recap: Sampling-Importance-Resampling (SIR)

function \([\mathcal{X}_t] = SIR[\mathcal{X}_{t-1}, \mathbf{y}_t]\)

\[\mathcal{X}_t = \mathcal{X}_{t-1} = \emptyset\]
for \(i = 1:N\)

\[\text{Sample } \mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)\]

\[w_t^i = p(\mathbf{y}_t|\mathbf{x}_t^i)\]
end
for \(i = 1:N\)

\(\text{Draw } i \text{ with probability } \propto w_t^i\)

\(\text{Add } \mathbf{x}_t^i \text{ to } \mathcal{X}_t\)
end

Initialize

Generate new samples

Update weights

Resample

Slide adapted from Michael Rubinstein
Recap: Sampling-Importance-Resampling (SIR)

\[
\text{function } \ [X_t] = SIR [X_{t-1}, y_t] \\
\tilde{X}_t = X_t = \emptyset \\
\text{for } i = 1: N \\
\quad \text{Sample } x_t^i \sim p(x_t | x_{t-1}^i) \\
\quad w_t^i = p(y_t | x_t^i) \\
\text{end} \\
\text{for } i = 1: N \\
\quad \text{Draw } i \text{ with probability } \propto w_t^i \\
\quad \text{Add } x_t^i \text{ to } X_t \\
\text{end}
\]

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: 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 $P(X_t | Y_t)$

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: 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: Mahalanobis Distance

- **Gating / Validation volume**
  - Our KF state of track $x_l$ is given by the prediction $\hat{x}_l^{(k)}$ and covariance $\Sigma_{p,l}^{(k)}$.
  - We define the innovation that measurement $y_j$ brings to track $x_l$ at time $k$ as $v_{j,l}^{(k)} = (y_j^{(k)} - x_{p,l}^{(k)})$
  - With this, we can write the observation likelihood shortly as $p(y_j^{(k)}|x_l^{(k)}) \sim \exp\left\{-\frac{1}{2}v_{j,l}^{(k)T}\Sigma_{p,l}^{(k)-1}v_{j,l}^{(k)}\right\}$
  - We define the ellipsoidal gating or validation volume as $V^{(k)}(\gamma) = \left\{ y | (y - x_{p,l}^{(k)})^T\Sigma_{p,l}^{(k)-1}(y - x_{p,l}^{(k)}) \leq \gamma \right\}$
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 \( kn_z \) degrees of freedom, with a threshold \( \alpha \) (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.
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: 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.
  - Integrate track generation into the assignment process.
  - After hypothesis generation, merge and prune the current hypothesis set.

Recap: Hypothesis Generation

• Create hypothesis matrix of the **feasible associations**

\[
\Theta = \begin{bmatrix}
1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

\( \Theta = \begin{bmatrix}
1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1
\end{bmatrix} \)

• 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**.
  – Enumerate all **assignments** that are consistent with this matrix.
Recap: Assignments

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 $x_{fa}$, $x_{nt}$

<table>
<thead>
<tr>
<th>$Z_j$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_{fa}$</th>
<th>$x_{nt}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$y_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$y_3$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$y_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
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:

\[
p(\Omega^{(k)}_j | Y^{(k)}) = p(Z_j^{(k)}, \Omega^{(k-1)}_{p(j)} | Y^{(k)})
\]

\[
\overset{Bayes}{=} \eta p(Y^{(k)} | Z_j^{(k)}, \Omega^{(k-1)}_{p(j)}) p(Z_j^{(k)}, \Omega^{(k-1)}_{p(j)})
\]

\[
= \eta p(Y^{(k)} | Z_j^{(k)}, \Omega^{(k-1)}_{p(j)}) p(Z_j^{(k)} | \Omega^{(k-1)}_{p(j)}) p(\Omega^{(k-1)}_{p(j)})
\]

- Normalization factor
- Measurement likelihood
- Prob. of assignment set
- Prob. of parent
Recap: Measurement Likelihood

• Use KF prediction
  – Assume that a measurement $y_i^{(k)}$ associated to a track $x_j$ has a Gaussian pdf centered around the measurement prediction $\hat{x}_j^{(k)}$ with innovation covariance $\hat{\Sigma}_j^{(k)}$.
  – 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

$$p \left( y^{(k)} | Z^{(k)}_j, \Omega^{(k-1)}_{p(j)} \right) = \prod_{i=1}^{M_k} \mathcal{N} \left( y_i^{(k)} ; \hat{x}_j, \hat{\Sigma}_j^{(k)} \right)^{\delta_i} W^{-1 - \delta_i}$$

$$= W^{-(N_{fal} + N_{new})} \prod_{i=1}^{M_k} \mathcal{N} \left( y_i^{(k)} ; \hat{x}_j, \hat{\Sigma}_j^{(k)} \right)^{\delta_i}$$
Recap: Probability of an Assignment Set

\[ p(Z_j^{(k)} | \Omega_p^{(k-1)}) \]

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

2. Probability of a specific assignment of measurements
   - Such that $M_k = N_{det} + N_{fal} + N_{new}$ holds.
   - This is determined as $1$ over the number of combinations

\[
\binom{M_k}{N_{det}} \binom{M_k - N_{det}}{N_{fal}} \binom{M_k - N_{det} - N_{fal}}{N_{new}}
\]

3. Probability of a specific assignment of tracks
   - Given that a track can be either detected or not detected.
   - This is determined as $1$ over the number of assignments

\[
\frac{N!}{(N - N_{det})!} \binom{N - N_{det}}{N_{det}}
\]

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

image source: [Zhang, Li, Nevatia, CVPR’08]
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.

Choose at most one match in each row and column to maximize sum of scores.
Recap: Linear Assignment Problem

• Formal definition

  – Maximize

  \[
  \sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} z_{ij}
  \]

  subject to

  \[
  \sum_{j=1}^{M} z_{ij} = 1; \quad i = 1, 2, \ldots, N
  \]

  \[
  \sum_{i=1}^{N} z_{ij} = 1; \quad j = 1, 2, \ldots, M
  \]

  \[
  z_{ij} \in \{0, 1\}
  \]

  Those constraints ensure that \(Z\) is a permutation matrix.

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

  – Note: Alternatively, we can minimize cost rather than maximizing weights.

  \[
  \arg \min_{z_{ij}} \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} z_{ij}
  \]
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 $c_{ij} = \alpha - w_{ij}$
  - Add source/sink nodes with 0 cost.
  - Directed edges with a capacity of 1.
Recap: Min-Cost Flow

- Conversion into flow graph
  - Pump $N$ units of flow from source to sink.
  - Internal nodes pass on flow ($\sum\text{flow in} = \sum\text{flow out}$).
  - Find the optimal paths along which to ship the flow.
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)
  ⇒ Connect the source and sink nodes to all intermediate nodes.
• Complication 2
  – Trivial solution: zero cost flow!
Recap: Network Flow Approach

Solution: Divide each detection into 2 nodes

Zhang, Li, Nevatia, **Global Data Association for Multi-Object Tracking using Network Flows**, CVPR’08.
Recap: Min-Cost Formulation

• Objective Function

\[ \mathcal{T}^* = \arg\min_{\mathcal{T}} \sum_i C_{in,i} f_{in,i} + \sum_i C_{i,\text{out}} f_{i,\text{out}} \]
\[ + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_i f_i \]

• subject to
  – Flow conservation at all nodes

\[ f_{in,i} + \sum_j f_{j,i} = f_i = f_{out,i} + \sum_j f_{i,j} \quad \forall i \]
  – Edge capacities

\[ f_i \leq 1 \]
Min-Cost Formulation

- **Objective Function**

\[ \mathcal{T}^* = \operatorname{argmin}_\mathcal{T} \sum_i C_{\text{in},i} f_{\text{in},i} + \sum_i C_{\text{out},i} f_{\text{out},i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_i f_i \]

**Likelihood of the detection**

- **Equivalent to Maximum A-Posteriori formulation**

\[ \mathcal{T}^* = \operatorname{argmax}_\mathcal{T} \prod_i P(o_i | \mathcal{T}) P(\mathcal{T}) \]

\[ P(\mathcal{T}) = \prod_{T_k \in \mathcal{T}} P(T_k) \]

**Independence assumption**

+ Markov

Slide credit: Laura Leal-Taixe
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

image source: [Clemente et al., RSS 2007]
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)

\[
p(I_2 | I_1, \xi) \quad \Rightarrow \quad E(\xi) = \int_{u \in \Omega} |I_1(u) - I_2(\omega(u, \xi))| \, du
\]

• **Indirect methods**
  – formulate alignment objective in terms of **reprojection error of geometric primitives** (e.g., points, lines)

\[
p(Y_2 | Y_1, \xi) \quad \Rightarrow \quad E(\xi) = \sum_i |y_{1,i} - \omega(y_{2,i}, \xi)|
\]
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:
    \[ E(T_{t-1}^t, X) = \sum_{i=1}^{N} \| \bar{y}_{t,i} - \pi(\bar{x}_i) \|_2^2 + \| \bar{y}_{t-1,i} - \pi(T_{t-1}^t \bar{x}_i) \|_2^2 \]
  - Introduced linear algorithm: 8-point

- **2D-to-3D**
  - Reprojection error:
    \[ E(T_t) = \sum_{i=1}^{N} \| y_{t,i} - \pi(T_t \bar{x}_i) \|_2^2 \]
  - Introduced linear algorithm: DLT PnP

- **3D-to-3D**
  - Reprojection error:
    \[ E(T_{t-1}^t) = \sum_{i=1}^{N} \| \bar{x}_{t-1,i} - T_t^{t-1} \bar{x}_{t,i} \|_2^2 \]
  - 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
     \[ \tilde{y}_i E \tilde{y}_i' = a_i E_s = 0 \quad \Rightarrow \quad A E_s = 0 \]
    using Kronecker product \( a_i = \tilde{y}_i \otimes \tilde{y}_i' \) and \( E_s = (e_{11}, e_{12}, e_{13}, ..., e_{33})^T \)
  2. Apply singular value decomposition (SVD) on \( A = U_A S_A V_A^T \) and unstack the 9th column of \( V_A \) into \( \hat{E} \).
  3. Project the approximate \( \hat{E} \) into the (normalized) essential space:
     Determine the SVD of \( \hat{E} = U \text{ diag}(\sigma_1, \sigma_2, \sigma_3) V^T \) with \( U, V \in SO(3) \)
     and replace the singular values \( \sigma_1 \geq \sigma_2 \geq \sigma_3 \) with 1,1,0 to find
\[
E = U \text{ diag}(1,1,0) V^T
\]
Recap: Eight-Point Algorithm cont.

• Algorithm (cont.):
  – Determine one of the following 2 possible solutions that intersects the points in front of both cameras:

\[
\mathbf{R} = \mathbf{U} \mathbf{R}_Z^\top \left( \pm \frac{\pi}{2} \right) \mathbf{V}^\top \\
\hat{\mathbf{t}} = \mathbf{U} \mathbf{R}_Z \left( \pm \frac{\pi}{2} \right) \text{diag}(1, 1, 0) \mathbf{U}^\top
\]

• A derivation can be found in the MASKS textbook, Ch. 5

• Remarks
  – Algebraic solution does not minimize geometric error
  – Refine using non-linear least-squares of reprojection error
  – Alternative: formulate epipolar constraints as „distance from epipolar line“ and minimize this non-linear least-squares problem
Recap: Eight-Point Algorithm cont.

- Normalized essential matrix: \( \| E \| = \| \hat{t} \| = 1 \)

- 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: \( \| \hat{t} \| = 0 \Rightarrow \| E \| = 0 \)

- 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 $Y_{t-2}, Y_{t-1}, Y_t$ for current and last frame pair
    \[
    \Rightarrow X_{t-2,t-1}, X_{t-1,t}
    \]
  - Rescale translation of current relative pose estimate to match the reconstruction scale with the distance ratio between corresponding point pairs
    \[
    r_{i,j} = \frac{\|x_{t-2,t-1,i} - x_{t-2,t-1,j}\|_2}{\|x_{t-1,t,i} - x_{t-1,t,j}\|_2}
    \]
  - Use mean or robust median over available pair ratios
Algorithm: 2D-to-2D Visual Odometry

Input: image sequence $I_{0:t}$
Output: aggregated camera poses $T_{0:t}$

Algorithm:
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 $T_k = T_{k-1}T_{k}^{k-1}$

Slide credit: Jörg Stückler
Recap: Triangulation

• Goal: Reconstruct 3D point $\tilde{x} = (x, y, z, w)^T \in \mathbb{P}^3$ 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
  $y'_i = \pi(T_i \tilde{x}) = \left( \begin{array}{c} \frac{r_{11}x + r_{12}y + r_{13}z + t_x w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \\ \frac{r_{21}x + r_{22}y + r_{23}z + t_y w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \end{array} \right)$

  – Each image provides one constraint $y_i - y'_i = 0$
  – Leads to system of linear equations $A\tilde{x} = 0$, two approaches:
    • Set $w = 1$ 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)
  $$\min_x \left\{ \sum_{i=1}^N \|y_i - y'_i\|^2_2 \right\}$$
Recap: Direct Linear Transform for PnP

• Goal: determine projection matrix \( \mathbf{P} = (\mathbf{R} \, \mathbf{t}) \in \mathbb{R}^{3 \times 4} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} \)

• Each 2D-to-3D point correspondence
  
  \begin{align*}
  \mathbf{x}_i &= (x_i, y_i, z_i, w_i)^\top \in \mathbb{P}^3 \\
  \mathbf{y}_i &= (x'_i, y'_i, w'_i)^\top \in \mathbb{P}^2
  \end{align*}

gives two constraints

\[
\begin{pmatrix}
    0 & -w'_i \mathbf{x}_i^\top \\
    w'_i \mathbf{x}_i^\top & 0 & y'_i \mathbf{x}_i^\top
  \end{pmatrix}
\begin{pmatrix}
    \mathbf{P}_1^\top \\
    \mathbf{P}_2^\top \\
    \mathbf{P}_3^\top
  \end{pmatrix} = 0
\]

through \( \mathbf{y}_i \times (\mathbf{P} \mathbf{x}_i) = 0 \)

• Form linear system of equation \( \mathbf{A} \mathbf{p} = 0 \) with \( \mathbf{p} := \begin{pmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{pmatrix} \in \mathbb{R}^9 \)

from \( N \geq 6 \) correspondences

• Solve for \( \mathbf{p} \): determine unit singular vector of \( \mathbf{A} \) corresponding to its smallest eigenvalue
Algorithm: 2D-to-3D Visual Odometry

Input: image sequence $I_{0:t}$
Output: aggregated camera poses $T_{0:t}$

Algorithm:
Initialize:
1. Extract and match keypoints between $I_0$ and $I_1$
2. Determine camera pose (Essential matrix) and triangulate 3D keypoints $X_1$

For each current image $I_k$:
1. Extract and match keypoints between $I_{k-1}$ and $I_k$
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 $X_k$
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, \( N \geq 3 \)

\[
X_{t-1} = \{x_{t-1,1}, \ldots, x_{t-1,N}\} \quad X_t = \{x_{t,1}, \ldots, x_{t,N}\}
\]

- Determine means of 3D point sets

\[
\mu_{t-1} = \frac{1}{N} \sum_{i=1}^{N} x_{t-1,i} \quad \mu_t = \frac{1}{N} \sum_{i=1}^{N} x_{t,i}
\]

- Determine rotation from

\[
A = \sum_{i=1}^{N} (x_{t-1} - \mu_{t-1}) (x_t - \mu_t)^\top \quad A = USV^\top \quad R_{t-1}^t = VU^\top
\]

- Determine translation as

\[
t_{t-1} = \mu_t - R_{t-1}^t \mu_{t-1}
\]
Algorithm: 3D-to-3D Stereo Visual Odometry

Input: stereo image sequence $I^{l}_{0:t}, I^{r}_{0:t}$
Output: aggregated camera poses $T_{0:t}$

Algorithm:
For each current stereo image $I^{l}_{k}, I^{r}_{k}$:
1. Extract and match keypoints between $I^{l}_{k}$ and $I^{l}_{k-1}$
2. Triangulate 3D points $X_{k}$ between $I^{l}_{k}$ and $I^{r}_{k}$
3. Compute camera pose $T^{k-1}_{k}$ from 3D-to-3D point matches $X_{k}$ to $X_{k-1}$
4. Aggregate camera poses by $T_{k} = T_{k-1}T^{k-1}_{k}$

Slide credit: Jörg Stückler
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

Harris Corners

DoG (SIFT) Blobs

Slide credit: Jörg Stückler

Image source: Svetlana Lazebnik
Recap: RANSAC

- **RAN**dom **SA**mple **C**onsensus algorithm for robust estimation

**Algorithm:**

Input: data $D$, $s$ required data points for fitting, success probability $p$, outlier ratio $\epsilon$

Output: inlier set

1. Compute required number of iterations $N = \frac{\log (1 - p)}{\log (1 - (1 - \epsilon)^s)}$

2. For $N$ iterations do:
   1. Randomly select a subset of $s$ 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

image source: [Clemente et al., RSS 2007]
Recap: Direct Visual Odometry Pipeline

- Avoid manually designed keypoint detection and matching

- Instead: direct image alignment

\[ E(\xi) = \int_{u \in \Omega} |I_1(u) - I_2(\omega(u, \xi))| \, du \]

- 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:
    \[ I_1(y) = I_2(\pi (T(\xi)Z_1(y)\bar{y})) \]
  – Estimate the warp by minimizing the residuals (similar to LK alignment)
    \[ E(\xi) = \sum_{y \in \Omega} \frac{r(y, \xi)^2}{\sigma_I^2} \quad r(y, \xi) = I_1(y) - I_2(\pi (T(\xi)Z_1(y)\bar{y})) \]

  ⇒ Non-linear least-squares problem (use second-order tools)
  – Important issue in practice: How to parametrize the poses?
Recap: Representing Motion using Lie Algebra se(3)

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

\[
\xi := \begin{pmatrix} \omega \\ v \end{pmatrix} \in \mathbb{R}^6
\]

\[
\omega \in \mathbb{R}^3
\]

\[
v \in \mathbb{R}^3
\]

\[
\hat{\xi} := \begin{pmatrix} \hat{\omega} & v \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{4 \times 4}
\]
• The exponential map finds the transformation matrix for a twist:

\[ \exp(\hat{\xi}) = \begin{pmatrix} \exp(\hat{\omega}) & A \mathbf{v} \\ 0 & 1 \end{pmatrix} \]

\[ \exp(\hat{\omega}) = \mathbf{I} + \frac{\sin|\omega|}{|\omega|} \hat{\omega} + \frac{1 - \cos|\omega|}{|\omega|^2} \hat{\omega}^2 \]

\[ A = \mathbf{I} + \frac{1 - \cos|\omega|}{|\omega|^2} \hat{\omega} + \frac{|\omega| - \sin|\omega|}{|\omega|^3} \hat{\omega}^2 \]
Recap: Logarithm Map of SE(3)

- The logarithm maps twists to transformation matrices:

\[
\log (T) = \begin{pmatrix}
\log(R) & A^{-1}t \\
0 & 0
\end{pmatrix}
\]

\[
\log(R) = \frac{|\omega|}{2 \sin |\omega|} (R - R^T) \quad |\omega| = \cos^{-1} \left( \frac{\text{tr}(R) - 1}{2} \right)
\]
Recap: Working with Twist Coordinates

- Let’s define the following notation:

  \begin{align*}
  &\text{Inversion of hat operator:} \\
  &\begin{pmatrix}
  0 & -\omega_3 & \omega_2 & v_1 \\
  \omega_3 & 0 & -\omega_1 & v_2 \\
  -\omega_2 & \omega_1 & 0 & v_3 \\
  0 & 0 & 0 & 0
  \end{pmatrix}^\vee = (\omega_1 \omega_2 \omega_3 v_1 v_2 v_3)^T
  \\
  &\text{Conversion:} \\
  &\xi(T) = (\log(T))^\vee, \quad T(\xi) = \exp(\hat{\xi})
  \\
  &\text{Pose inversion:} \\
  &\xi^{-1} = \log(T(\xi)^{-1}) = -\xi
  \\
  &\text{Pose concatenation:} \\
  &\xi_1 \oplus \xi_2 = (\log(T(\xi_2)T(\xi_1)))^\vee
  \\
  &\text{Pose difference:} \\
  &\xi_1 \ominus \xi_2 = (\log(T(\xi_2)^{-1}T(\xi_1)))^\vee
  \end{align*}
Recap: Optimization with Twist Coordinates

- Twists provide a minimal local representation without singularities.

- Since \( \text{SE}(3) \) is a smooth manifold, we can decompose transformations in each optimization step into the transformation itself and an infinitesimal increment.

\[
T(\xi) = T(\xi) \exp(\delta\xi) = T(\delta\xi \oplus \xi)
\]

- We can then optimize an energy function \( E(\xi_i, \delta\xi) \) in order to estimate the pose increment \( \delta\xi \), e.g., using Gradient descent.

\[
\delta\xi^* = 0 - \eta \nabla_{\delta\xi} E(\xi_i, \delta\xi)
\]

\[
T(\xi_{i+1}) = T(\xi_i) \exp(\delta\xi^*)
\]
Algorithm: Direct RGB-D Visual Odometry

**Input:** RGB-D image sequence $I_{0:t}, Z_{0:t}$

**Output:** aggregated camera poses $T_{0:t}$

**Algorithm:**
For each current RGB-D image $I_k, Z_k$:
1. Estimate relative camera motion $T_{k-1}^k$ 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 $T_k = T_{k-1} T_{k-1}^k$
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

image source: [Clemente et al., RSS 2007]
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 $I_{0:t} = \{I_0, \ldots, I_t\}$
  – Stereo case: Left/right images $I_{0:t}^l = \{I_0^l, \ldots, I_t^l\}, I_{0:t}^r = \{I_0^r, \ldots, I_t^r\}$
  – RGB-D case: Color/depth images $I_{0:t} = \{I_0, \ldots, I_t\}, Z_{0:t} = \{Z_0, \ldots, Z_t\}$
  – Robotics: **control inputs** $U_{1:t}$

• **Output:**
  – **Camera pose** estimates $T_t \in SE(3)$ 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: $Y_t = \{y_{t,1}, ..., y_{t,N}\}$ (2D or 3D image points)
- Direct RGB-D method: $Y_t = \{I_t, Z_t\}$ (all image pixels)
- ...

- Involves data association to map elements $M = \{m_1, ..., m_S\}$
- We denote correspondences by $c_{t,i} = j$, $1 \leq i \leq N$, $1 \leq j \leq S$
Recap: Probabilistic Formulation of Visual SLAM

- SLAM posterior probability: 
  \[ p(\xi_{0:t}, M | Y_{0:t}, U_{1:t}) \]

- Observation likelihood: 
  \[ p(Y_t | \xi_t, M) \]

- State-transition probability: 
  \[ p(\xi_t | \xi_{t-1}, U_t) \]
Recap: Online SLAM Methods

- **Marginalize** out previous poses

\[
p(\xi_t, M | Y_{0:t}, U_{1:t}) = \int \cdots \int p(\xi_{0:t}, M | Y_{0:t}, U_{1:t}) \, d\xi_{t-1} \cdots d\xi_0
\]

- 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 $x_t$:

$$x_t = \begin{pmatrix} \xi_t \\ m_{t,1} \\ \vdots \\ m_{t,S} \end{pmatrix}$$

$$\Sigma_t = \begin{pmatrix} \Sigma_{t,\xi\xi} & \Sigma_{t,\xi m_1} & \cdots & \Sigma_{t,\xi m_S} \\ \Sigma_{t,m_1\xi} & \Sigma_{t,m_1m_1} & \cdots & \Sigma_{t,m_1m_S} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{t,m_S\xi} & \Sigma_{t,m_Sm_1} & \cdots & \Sigma_{t,m_Sm_S} \end{pmatrix}$$

$$= \begin{pmatrix} \Sigma_{t,\xi\xi} & \Sigma_{t,\xi m} \\ \Sigma_{t,m\xi} & \Sigma_{t,mm} \end{pmatrix}$$
Recap: 2D EKF-SLAM State-Transition Model

- **State/control variables**
  \[
  \xi_t = (x_t \ y_t \ \theta_t)^T \quad m_{t,j} = (m_{t,j,x} \ m_{t,j,y})^T \\
  u_t = (v_t \ \omega_t)^T = (\|v\|_2 \ \|\omega\|_2)^T
  \]

- **State-transition model**
  
  - **Pose**: 
    \[
    \xi_t = g_\xi(\xi_{t-1}, u_t) + \epsilon_{\xi,t} \quad \epsilon_{\xi,t} \sim \mathcal{N}(0, \Sigma_{d_t,\xi})
    \]
    \[
    g_\xi(\xi_{t-1}, u_t) = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \theta_{t-1} + \frac{v_t}{\omega_t} \sin(\theta_t + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \theta_{t-1} - \frac{v_t}{\omega_t} \cos(\theta_t + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}
    \]

  - **Landmarks**: 
    \[
    m_t = g_m(m_{t-1}) = m_{t-1}
    \]

  - **Combined**: 
    \[
    x_t = g(x_{t-1}, u_t) + \epsilon_t \quad \epsilon_t \sim \mathcal{N}(0, \Sigma_{d_t}) \quad g(x_{t-1}, u_t) = \begin{pmatrix} g_\xi(\xi_{t-1}, u_t) \\ g_m(m_{t-1}) \end{pmatrix} \quad \Sigma_{d_t} = \begin{pmatrix} \Sigma_{d_t,\xi} & 0 \\ 0 & 0 \end{pmatrix}
    \]
Recap: 2D EKF-SLAM Observation Model

- State/measurement variables
  \[ y_t = (d_t \phi_t)^\top \quad m_{t,j} = (m_{t,j,x} m_{t,j,y})^\top \]

- Observation model:
  \[ y_t = h(\xi_t, m_{t,ct}) + \delta_t \quad \delta_t \sim \mathcal{N}(0, \Sigma_{\delta_t}) \]
  \[ h(\xi_t, m_{t,ct}) = \begin{pmatrix} \|m_{t,ct,\text{rel}}\|_2 \\ \text{atan2}(m_{t,ct,y,\text{rel}}, m_{t,ct,x,\text{rel}}) \end{pmatrix} \]
  \[ m_{t,ct,\text{rel}} := R(-\theta_t) \left( m_{t,ct} - (x_t y_t)^\top \right) \]
Recap: State Initialization

• First frame:
  – Anchor reference frame at initial pose
  – Set pose covariance to zero

\[
\begin{align*}
  x_0^- &= 0 \\
  \Sigma_{0,\xi\xi}^- &= 0
\end{align*}
\]

• New landmark:
  – Initial position unknown
  – Initialize mean at zero
  – Initialize covariance to infinity (large value)

\[
\begin{align*}
  \Sigma_{0,\xi m}^- &= \Sigma_{0,m\xi}^- = 0 \\
  \Sigma_{0,mm}^- &= \infty I
\end{align*}
\]
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: 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
Pose Graph Optimization

- Optimization of poses
  - From relative pose constraints deduced from the image observations
  - Map recovered from the optimized poses

- Deduce relative constraints between poses from image observations, e.g.,
  - 8-point algorithm
  - Direct image alignment
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: Recurrent Networks

- **Feed-forward networks**
  - Simple neural network structure: 1-to-1 mapping of inputs to outputs

- **Recurrent Neural Networks**
  - Generalize this to arbitrary mappings
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

Image source: Christopher Olah, [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
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.

Slide adapted from Andrej Karpathy
Recap: Video to Text Description

Our LSTM network is connected to a CNN for RGB frames or a CNN for optical flow images.

Source: Subhashini Venugopalan, ICCV'15
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.

\[ \| f(x_i^a) - f(x_i^p) \|^2 < \| f(x_i^a) - f(x_i^n) \|^2 \]

⇒ 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…

Image source: Fischer et al., ICCV'15
• **Correlation network**
  – Central idea: compute a correlation score between two feature maps
    \[
    c(x_1, x_2) = \sum_{o \in [-k,k] \times [-k,k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle
    \]
  – 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

Image source: Fischer et al., ICCV'15
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!