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

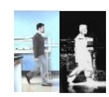
Announcements

- 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.

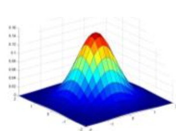
Content of the Lecture



- 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

Image source: Tobias Jaegoli

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:

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

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

$\Rightarrow$ *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

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \quad \text{where} \quad \boldsymbol{\Sigma}_k = \sigma_k^2\mathbf{I}$$

  - Check every new pixel value against the existing K components until a match is found (pixel value within $2.5\,\sigma_k$ of μ_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/σ_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.

$$\pi_k^{(t+1)} = (1-\alpha)\pi_k^{(t)} + \alpha M_{k,t}$$

  - Adapt only the parameters for the matching component

$$\boldsymbol{\mu}_k^{(t+1)} = (1-\rho)\boldsymbol{\mu}_k^{(t)} + \rho x^{(t+1)}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = (1-\rho)\boldsymbol{\Sigma}_k^{(t)} + \rho(x^{(t+1)} - \boldsymbol{\mu}_k^{(t+1)})(x^{(t+1)} - \boldsymbol{\mu}_k^{(t+1)})^T$$

where

$$\rho = \alpha \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

(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 $K(\cdot)$ as

$$p(\mathbf{x}^{(t)}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x}^{(t)} - \mathbf{x}^{(i)})$$

  - Choose K to be a Gaussian $\mathcal{N}(0, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} = \mathrm{diag}\{\sigma_j\}$. Then

$$p(\mathbf{x}^{(t)}) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\frac{(x_j^{(t)} - x_j^{(i)})^2}{\sigma_j^2}}$$

  - A pixel is considered foreground if $p(\mathbf{x}^{(t)}) < \theta$ for a threshold θ.
    - 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

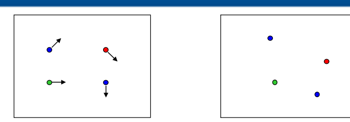
[A. Elgammal, D. Harwood, L. Davis, ECCV'00]

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



$I(x,y,t-1)$ $I(x,y,t)$

- 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.

Slide credit: Svetlana Lazebnik

Recap: Lucas-Kanade Optical Flow

- Use all pixels in a $K{\times}K$ window to get more equations.
- Least squares problem:

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\times 2}{A}\ \underset{2\times 1}{d} = \underset{25\times 1}{b}$$

- Minimum least squares solution given by solution of

$$\underset{2\times 2}{(A^T A)}\ \underset{2\times 1}{d} = \underset{2\times 1}{A^T b}$$

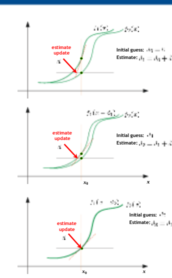
Recall the Harris detector!

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

Slide credit: Svetlana Lazebnik

Recap: Iterative LK Refinement

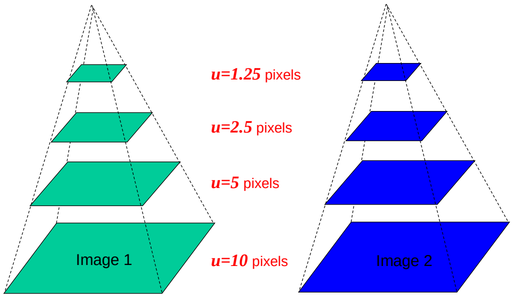
- Estimate velocity at each pixel using one iteration of LK estimation.
- Warp one image toward the other using the estimated flow field.
- Refine estimate by repeating the process.



- Iterative procedure
  - Results in subpixel accurate localization.
  - Converges for small displacements.

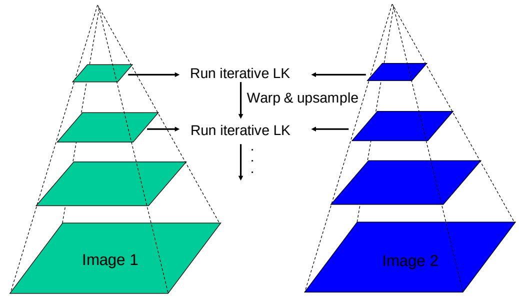
Slide adapted from Steve Seitz

Recap: Coarse-to-fine Optical Flow Estimation



Gaussian pyramid of image 1 Gaussian pyramid of image 2

Recap: Coarse-to-fine Optical Flow Estimation



Gaussian pyramid of image 1 Gaussian pyramid of image 2

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.

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Recap: General LK Image Registration

- Goal
  - Find the warping parameters $\mathbf{p}$ that minimize the sum-of-squares intensity difference between the template image $T(\mathbf{x})$ and the warped input image $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$.
- LK formulation
  - Formulate this as an optimization problem

$$\arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]^2$$

  - We assume that an initial estimate of $\mathbf{p}$ is known and iteratively solve for increments to the parameters $\Delta\mathbf{p}$:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right]^2$$

Recap: Step-by-Step Derivation

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

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) \approx I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + \mathcal{O}(\Delta\mathbf{p}^2)$$

$$= I(\mathbf{W}([x,y];p_1,\ldots,p_n))$$

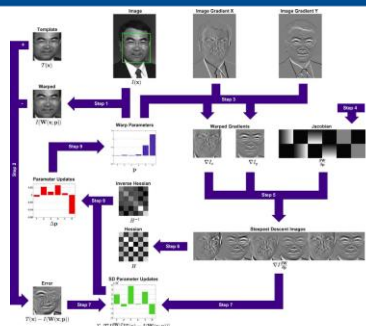
$$+ \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{bmatrix} \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_n \end{bmatrix}$$

Gradient ∇I | Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ | Increment parameters to solve for $\Delta\mathbf{p}$

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 ∇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 $\mathbf{H} = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$
  - Compute $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[T([x,y]) - I(\mathbf{W}([x,y];\mathbf{p}))\right]$
  - Compute $\Delta\mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^T \left[T([x,y]) - I(\mathbf{W}([x,y];\mathbf{p}))\right]$
  - 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

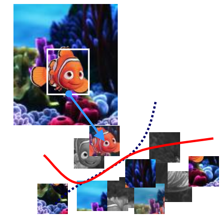
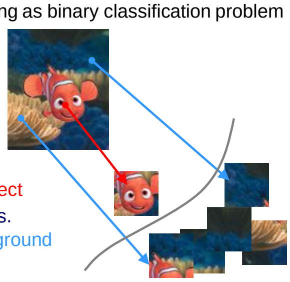


Image source: Robert Collins

Recap: Tracking as Online Classification

- Tracking as binary classification problem

object
vs.
background

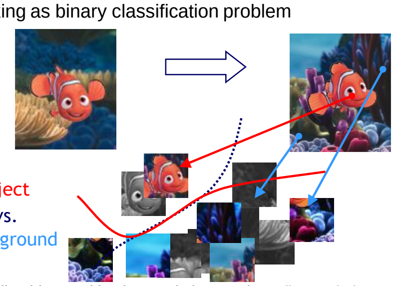


Slide credit: Helmut Grabner
Image source: Disney/Pixar

Recap: Tracking as Online Classification

- Tracking as binary classification problem

object
vs.
background



- Handle object and background changes by online updating

Slide credit: Helmut Grabner
Image source: Disney/Pixar

Recap: AdaBoost – "Adaptive Boosting"

- Main idea [Freund & Schapire, 1996]
  - Iteratively select an ensemble of classifiers
  - Reweight misclassified training examples after each iteration to focus training on difficult cases.
- Components
  - $h_m(\mathbf{x})$: "weak" or base classifier
    - Condition: <50% training error over any distribution
  - $H(\mathbf{x})$: "strong" or final classifier
- AdaBoost:
  - Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

$$H(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$$

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 $h_m(\mathbf{x})$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n) \qquad I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$$

   b) Estimate the weighted error of this classifier on $\mathbf{X}$:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

   c) Calculate a weighting coefficient for $h_m(\mathbf{x})$:

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$

   d) Update the weighting coefficients:

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\right\}$$

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 λ 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.

N. Oza and S. Russell. Online Bagging and Boosting. Artificial Intelligence and Statistics, 2001.

Recap: From Offline to Online Boosting

off-line

Given:
- set of labeled training samples $\mathcal{X} = \{\langle \mathbf{x}_1, y_1 \rangle, ..., \langle \mathbf{x}_L, y_L \rangle \mid y_i \pm 1\}$
- weight distribution over them $D_0 = 1/L$

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

next

$$h^{strong}(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x})\right)$$

on-line

Given:
- ONE labeled training sample $\langle \mathbf{x}, y \rangle \mid y \pm 1$
- strong classifier to update
- initial importance $\lambda = 1$

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

next

$$h^{strong}(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x})\right)$$

Slide credit: Helmut Grabner

Recap: Online Boosting for Feature Selection

- Introducing "Selector"
  - Selects one feature from its local feature pool

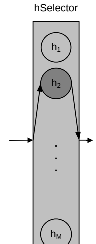
$\mathcal{H}^{weak} = \{h_1^{weak}, ..., h_M^{weak}\}$

$\mathcal{F} = \{f_1, ..., f_M\}$

$h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x})$

$m = \arg\min_i e_i$

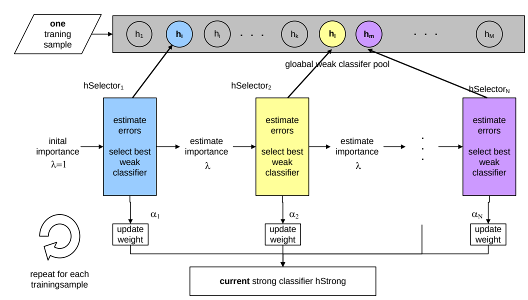
On-line boosting is performed on the Selectors and not on the weak classifiers directly.



H. Grabner and H. Bischof. On-line boosting and vision. CVPR, 2006.

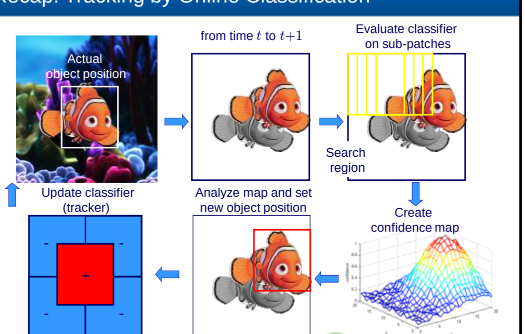
Slide credit: Helmut Grabner

Recap: Direct Feature Selection



Slide credit: Helmut Grabner

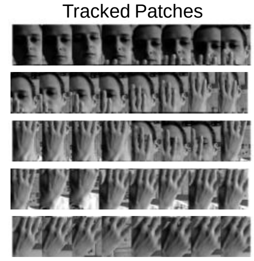
Recap: Tracking by Online Classification



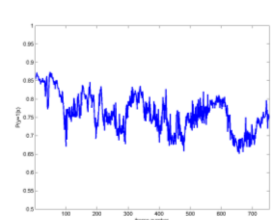
Slide credit: Helmut Grabner
Image source: Disney/Pixar

Recap: Drifting Due to Self-Learning Policy

Tracked Patches



Confidence

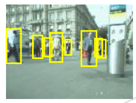


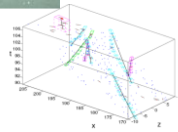
$\Rightarrow$ Not only does it drift, it also remains confident about it!

Slide credit: Helmut Grabner
Image source: Grabner et al., ECCV'08

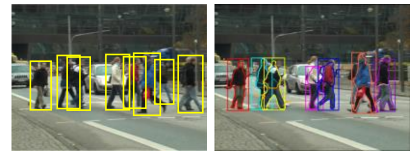
Course Outline

- 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



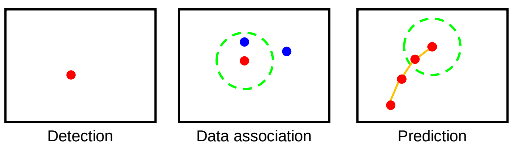


Recap: Tracking-by-Detection



- 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

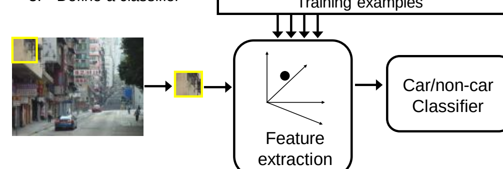
Recap: Elements of Tracking



- 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

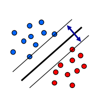
- For sliding-window object detection, we need to:
  1. Obtain training data
  2. Define features
  3. Define a classifier



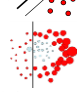
Slide credit: Kristen Grauman

Recap: Object Detector Design

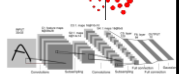
- In practice, the classifier often determines the design.
  - Types of features
  - Speedup strategies
- We looked at 3 state-of-the-art detector designs
  - Based on SVMs



  - Based on Boosting

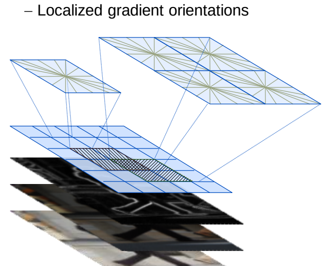


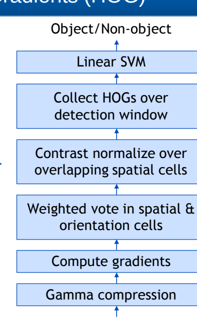
  - Based on CNNs



Recap: Histograms of Oriented Gradients (HOG)

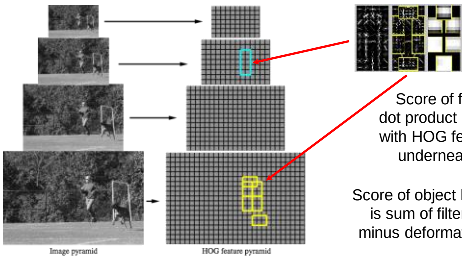
- Holistic object representation
  - Localized gradient orientations





Slide adapted from Navneet Dalal

Recap: Deformable Part-based Model (DPM)



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

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

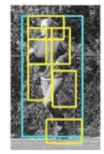
- Multiscale model captures features at two resolutions

[Felzenszwalb, McAllister, Ramanan, CVPR'08]

Recap: DPM Hypothesis Score

$$\text{score}(p_0,\ldots,p_n) = \underbrace{\sum_{i=0}^{n} F_i \cdot \phi(H,p_i)}_{\text{"data term"}} - \underbrace{\sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)}_{\text{"spatial prior"}}$$

filters; displacements; deformation parameters



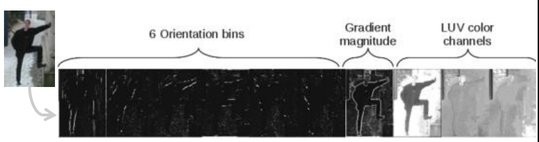
$$\text{score}(z) = \beta \cdot \Psi(H,z)$$

concatenation filters and deformation parameters

concatenation of HOG features and part displacement features

[Felzenszwalb, McAllister, Ramanan, CVPR'08]

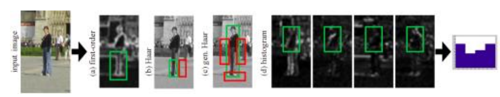
Recap: Integral Channel Features



- 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.

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

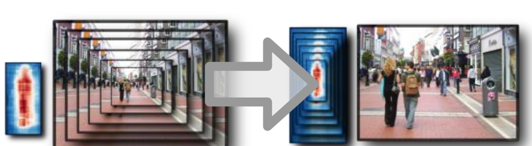
Recap: Integral Channel Features



- 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

R. Benenson, M. Mathias, R. Timofte, L. Van Gool. Pedestrian Detection at 100 Frames per Second, CVPR'12.

Recap: VeryFast Detector

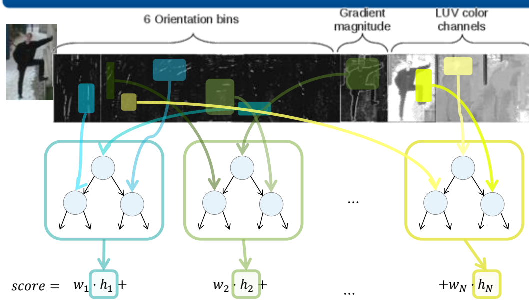
- Idea 2: Reduce training time by feature interpolation

50 models, 1 image scale

5 models, 1 image scale

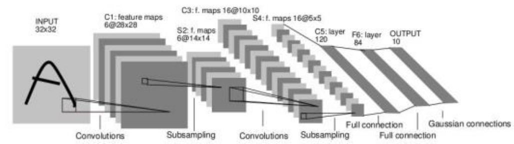
- Shown to be possible for Integral Channel features
  - P. Dollár, S. Belongie, Perona. The Fastest Pedestrian Detector in the West, BMVC 2010.

Recap: VeryFast Classifier Construction



- Ensemble of short trees, learned by AdaBoost

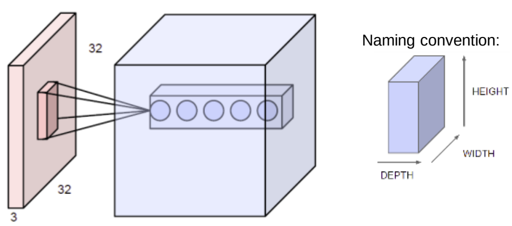
Recap: Convolutional Neural Networks



- 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

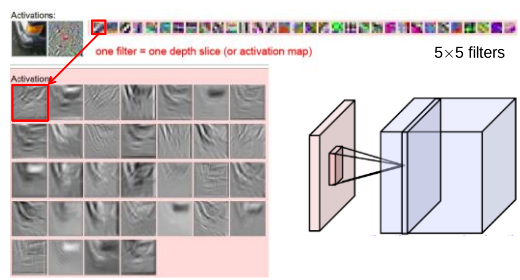
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Recap: Convolution Layers



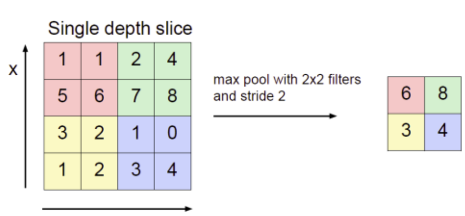
- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single [1×1×depth] depth column in output volume.

Recap: Activation Maps



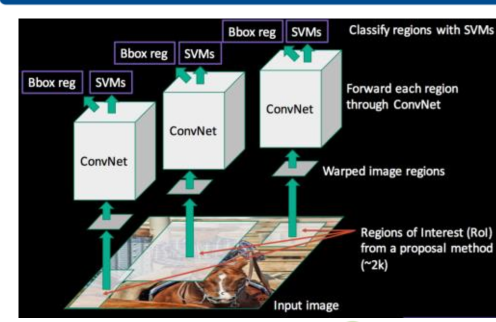
Activation maps

Recap: Pooling Layers



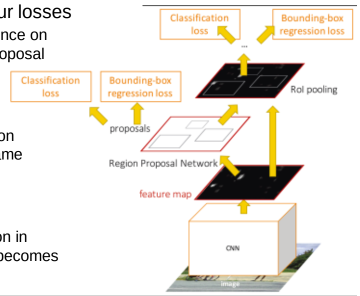
- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

Recap: R-CNN for Object Detection



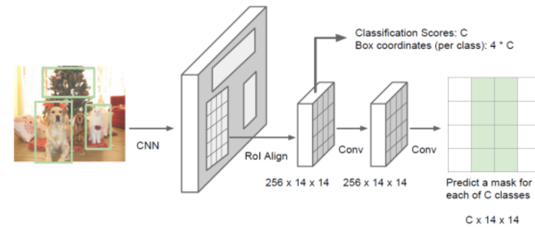
Recap: Faster R-CNN

- 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.



Slide credit: Ross Girshick

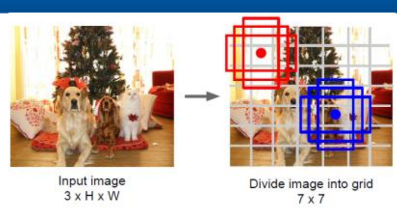
Recap: Mask R-CNN



K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, arXiv 1703.06870.

Slide credit: FeiFei Li

Recap: YOLO / SSD

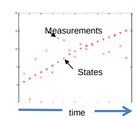


- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)

Slide credit: FeiFei Li

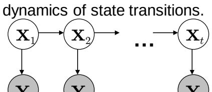
Course Outline

- 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

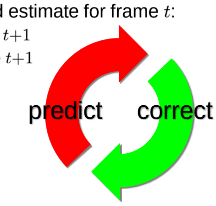
- Inference problem
  - The hidden state consists of the true parameters we care about, denoted $\mathbf{X}$.
  - The measurement is our noisy observation that results from the underlying state, denoted $\mathbf{Y}$.
  - At each time step, state changes (from $\mathbf{X}_{t-1}$ to $\mathbf{X}_t$) and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



Slide credit: Kristen Grauman

Recap: Tracking as Induction

- Base case:
  - Assume we have initial prior that predicts state in absence of any evidence: $P(\mathbf{X}_0)$
  - At the first frame, *correct* this given the value of $\mathbf{Y}_0 = \mathbf{y}_0$
- Given corrected estimate for frame t:
  - Predict for frame $t+1$
  - Correct for frame $t+1$



Slide credit: Svetlana Lazebnik

Recap: Prediction and Correction

- Prediction:

$$P(X_t \mid y_0,\ldots,y_{t-1}) = \int \underbrace{P(X_t \mid X_{t-1})}_{\text{Dynamics model}} \underbrace{P(X_{t-1} \mid y_0,\ldots,y_{t-1})}_{\text{Corrected estimate from previous step}} dX_{t-1}$$

- Correction:

$$P(X_t \mid y_0,\ldots,y_t) = \frac{\overbrace{P(y_t \mid X_t)}^{\text{Observation model}} \overbrace{P(X_t \mid y_0,\ldots,y_{t-1})}^{\text{Predicted estimate}}}{\int P(y_t \mid X_t) P(X_t \mid y_0,\ldots,y_{t-1}) dX_t}$$

Recap: Linear Dynamic Models

- Dynamics model
  - State undergoes linear transformation D_t plus Gaussian noise

$$\boldsymbol{x}_t \sim N\left(\boldsymbol{D}_t \boldsymbol{x}_{t-1}, \Sigma_{d_t}\right)$$

- Observation model
  - Measurement is linearly transformed state plus Gaussian noise

$$\boldsymbol{y}_t \sim N\left(\boldsymbol{M}_t \boldsymbol{x}_t, \Sigma_{m_t}\right)$$

Recap: Constant Velocity (1D Points)

- State vector: position p and velocity v

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \qquad \begin{aligned} p_t &= p_{t-1} + (\Delta t) v_{t-1} + \varepsilon \\ v_t &= v_{t-1} + \xi \end{aligned}$$

(greek letters denote noise terms)

$$x_t = D_t x_{t-1} + noise = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + noise$$

- Measurement is position only

$$y_t = M x_t + noise = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + noise$$

Recap: Constant Acceleration (1D Points)

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

$$x_t = \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} \qquad \begin{aligned} p_t &= p_{t-1} + (\Delta t) v_{t-1} + \tfrac{1}{2}(\Delta t)^2 a_{t-1} + \varepsilon \\ v_t &= v_{t-1} + (\Delta t) a_{t-1} + \xi \\ a_t &= a_{t-1} + \zeta \end{aligned}$$

(greek letters denote noise terms)

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

- Measurement is position only

$$y_t = M x_t + noise = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} + noise$$

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

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

- Then we have

$$x_t = \begin{bmatrix} p_{1,t} \\ p_{2,t} \\ p_{3,t} \end{bmatrix} \qquad \begin{aligned} p_{1,t} &= p_{1,t-1} + (\Delta t) p_{2,t-1} + \tfrac{1}{2}(\Delta t)^2 p_{3,t-1} + \varepsilon \\ p_{2,t} &= p_{2,t-1} + (\Delta t) p_{3,t-1} + \xi \\ p_{3,t} &= -p_{1,t-1} + \zeta \end{aligned} \qquad D_t = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 1 & \Delta t \\ -1 & 0 & 0 \end{bmatrix}$$

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")

$$P(X_t \mid y_0,\ldots,y_{t-1})$$

Mean and std. dev. of predicted state:

$$\mu_t^-, \sigma_t^-$$

Time advances: t++

$$P(X_t \mid y_0,\ldots,y_t)$$

Mean and std. dev. of corrected state:

$$\mu_t^+, \sigma_t^+$$

Recap: General Kalman Filter (>1dim)

PREDICT

$$x_t^- = D_t x_{t-1}^+$$
$$\Sigma_t^- = D_t \Sigma_{t-1}^+ D_t^T + \Sigma_{d_t}$$

CORRECT

$$K_t = \Sigma_t^- M_t^T \left(M_t \Sigma_t^- M_t^T + \Sigma_{m_t}\right)^{-1}$$
$$x_t^+ = x_t^- + K_t\left(y_t - M_t x_t^-\right)$$
$$\Sigma_t^+ = \left(I - K_t M_t\right)\Sigma_t^-$$

"residual"
"Kalman gain"

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

Slide credit: Kristen Grauman

Recap: Kalman Filter – Detailed Algorithm

- Algorithm summary
  - Assumption: linear model
$$\mathbf{x}_t = \mathbf{D}_t \mathbf{x}_{t-1} + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{M}_t \mathbf{x}_t + \delta_t$$
  - Prediction step
$$\mathbf{x}_t^- = \mathbf{D}_t \mathbf{x}_{t-1}^+$$
$$\Sigma_t^- = \mathbf{D}_t \Sigma_{t-1}^+ \mathbf{D}_t^T + \Sigma_{d_t}$$
  - Correction step
$$\mathbf{K}_t = \Sigma_t^- \mathbf{M}_t^T \left(\mathbf{M}_t \Sigma_t^- \mathbf{M}_t^T + \Sigma_{m_t}\right)^{-1}$$
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{M}_t \mathbf{x}_t^-\right)$$
$$\Sigma_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{M}_t\right)\Sigma_t^-$$

Extended Kalman Filter (EKF)

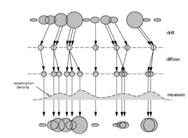
- Algorithm summary
  - Nonlinear model
$$\mathbf{x}_t = \mathbf{g}(\mathbf{x}_{t-1}) + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \delta_t$$
  - Prediction step
$$\mathbf{x}_t^- = \mathbf{g}(\mathbf{x}_{t-1}^+)$$
$$\Sigma_t^- = \mathbf{G}_t \Sigma_{t-1}^+ \mathbf{G}_t^T + \Sigma_{d_t}$$
  - Correction step
$$\mathbf{K}_t = \Sigma_t^- \mathbf{H}_t^T \left(\mathbf{H}_t \Sigma_t^- \mathbf{H}_t^T + \Sigma_{m_t}\right)^{-1}$$
$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{h}(\mathbf{x}_t^-)\right)$$
$$\Sigma_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right)\Sigma_t^-$$

with the Jacobians

$$\mathbf{G}_t = \left.\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_{t-1}^+}$$
$$\mathbf{H}_t = \left.\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_t^-}$$

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

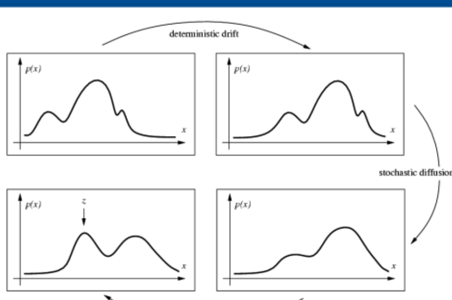
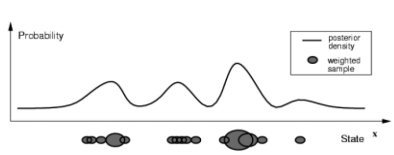


Figure from Isard & Blake

Recap: Factored Sampling



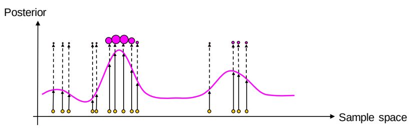
- Idea: Represent state distribution non-parametrically
  - Prediction: Sample points from prior density for the state, $P(X)$
  - Correction: Weight the samples according to $P(Y|X)$

$$P(X_t \mid y_0, \ldots, y_t) = \frac{P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1})}{\int P(y_t \mid X_t) P(X_t \mid y_0, \ldots, y_{t-1}) dX_t}$$

Slide credit: Svetlana Lazebnik

Recap: Particle Filtering

- Many variations, one general concept:
  - *Represent the posterior pdf by a set of randomly chosen weighted samples (particles)*

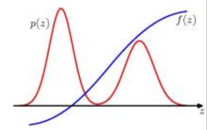


  - Randomly Chosen = Monte Carlo (MC)
  - As the number of samples become very large – the characterization becomes an equivalent representation of the true pdf.

Background: Monte-Carlo Sampling

- Objective:
  - Evaluate expectation of a function $f(\mathbf{z})$ w.r.t. a probability distribution $p(\mathbf{z})$.

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}$$



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

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

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

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

$\Rightarrow$ Unbiased estimate, independent of the dimension of $\mathbf{z}$!

Background: Importance Sampling

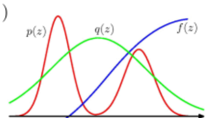
- Idea
  - Use a proposal distribution $q(\mathbf{z})$ from which it is easy to draw samples and which is close in shape to f.
  - Express expectations in the form of a finite sum over samples $\{\mathbf{z}^{(l)}\}$ drawn from $q(\mathbf{z})$.

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L}\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)})$$

  - with importance weights

$$r_l = \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}$$



Recap: Sequential Importance Sampling

function $\left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — Sample from proposal pdf

$w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — Update weights

$\eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$w_t^i = w_t^i/\eta$ — Normalize weights

end

Recap: Sequential Importance Sampling

function $\left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\mathbf{x}_t^i \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)$ — Sample from proposal pdf

$w_t^i = w_{t-1}^i \dfrac{p(\mathbf{y}_t|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t)}$ — Update weights

$\eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$w_t^i = w_t^i/\eta$ — Normalize weights

end

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

Recap: SIS Algorithm with Transitional Prior

function $\left[\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N\right] = SIS\left[\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N, \mathbf{y}_t\right]$

$\eta = 0$ — Initialize

for $i = 1{:}N$

$\mathbf{x}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$ — Sample from proposal pdf

$w_t^i = w_{t-1}^i p(\mathbf{y}_t|\mathbf{x}_t^i)$ — Update weights

$\eta = \eta + w_t^i$ — Update norm. factor

end

for $i = 1{:}N$

$w_t^i = w_t^i/\eta$ — Normalize weights

end

Transitional prior
$q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}^i)$

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 $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.
- Idea: Resampling
  - Eliminate particles with low importance weights and increase the number of particles with high importance weight.

$$\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N \rightarrow \left\{\mathbf{x}_t^{i*}, \frac{1}{N}\right\}_{i=1}^N$$

  - The new set is generated by sampling with replacement from the discrete representation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ such that

$$Pr\left\{\mathbf{x}_t^{i*} = \mathbf{x}_t^j\right\} = w_t^j$$

Recap: Efficient Resampling Approach

- From Arulampalam paper:

```
Algorithm 2: Resampling Algorithm
[{x_k^{j*}, w_k^j, i^j}_{j=1}^{N_s}] = RESAMPLE [{x_k^i, w_k^i}_{i=1}^{N_s}]
• 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 ~ 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^{j*} = x_k^i
  – Assign weight: w_k^j = N_s^{-1}
  – Assign parent: i^j = 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

function $\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = PF\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$

Apply SIS filtering $\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = SIS\left[\left\{\mathbf{x}_{t-1}^i, w_{t-1}^i\right\}_{i=1}^N, \mathbf{y}_t\right]$

Calculate N_{eff}

if $N_{eff} < N_{thr}$

$\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right] = RESAMPLE\left[\left\{\mathbf{x}_t^i, w_t^i\right\}_{i=1}^N\right]$

end

- 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: Sampling-Importance-Resampling (SIR)

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

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ — Initialize

for $i = 1{:}N$

Sample $\mathbf{x}_t^i \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^i)$ — Generate new samples

$w_t^i = p(\mathbf{y}_t | \mathbf{x}_t^i)$ — Update weights

end

for $i = 1{:}N$

Draw i with probability $\propto w_t^i$ — Resample

Add $\mathbf{x}_t^i$ *to* $\mathcal{X}_t$

end

Recap: Sampling-Importance-Resampling (SIR)

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

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $i = 1{:}N$

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 $i = 1{:}N$

Draw i with probability $\propto w_t^i$

Add $\mathbf{x}_t^i$ *to* $\mathcal{X}_t$

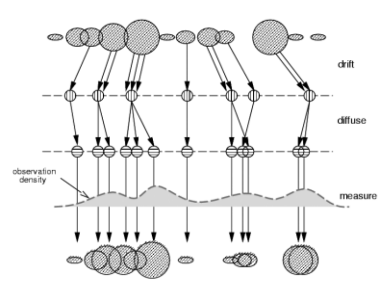
end

Important property:

Particles are distributed according to pdf from previous time step.

Particles are distributed according to posterior from this time step.

Recap: Condensation Algorithm

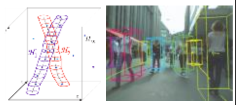


- Start with weighted samples from previous time step
- Sample and shift according to dynamics model
- Spread due to randomness; this is predicted density $P(X_t|Y_{t-1})$
- Weight the samples according to observation density
- Arrive at corrected density estimate $P(X_t|Y_t)$

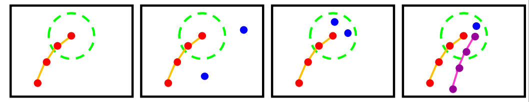
M. Isard and A. Blake, CONDENSATION -- conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998

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



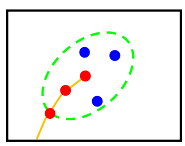
Recap: Motion Correspondence Ambiguities



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?

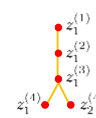
Recap: Mahalanobis Distance

- Gating / Validation volume
  - Our KF state of track $\mathbf{x}_l$ is given by the prediction $\hat{\mathbf{x}}_l^{(k)}$ and covariance $\Sigma_{p,l}^{(k)}$.
  - We define the innovation that measurement $\mathbf{y}_j$ brings to track $\mathbf{x}_l$ at time k as
    $$\mathbf{v}_{j,l}^{(k)} = (\mathbf{y}_j^{(k)} - \mathbf{x}_{p,l}^{(k)})$$
  - With this, we can write the observation likelihood shortly as
    $$p(\mathbf{y}_j^{(k)}|\mathbf{x}^{(k)}) \sim \exp\left\{-\frac{1}{2}\mathbf{v}_{j,l}^{(k)^T}\Sigma_{p,l}^{(k)^{-1}}\mathbf{v}_{j,l}^{(k)}\right\}$$
  - We define the ellipsoidal gating or validation volume as
    $$V^{(k)}(\gamma) = \left\{\mathbf{y} | (\mathbf{y} - \mathbf{x}_{p,l}^{(k)})^T \Sigma_{p,l}^{(k)^{-1}} (\mathbf{y} - \mathbf{x}_{p,l}^{(k)}) \le \gamma\right\}$$



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 χ^2 distribution with kn_z degrees of freedom, with a threshold α (set according to χ^2 distribution tables).
    - Problem for long tracks: modified log-likelihood gets dominated by old terms and responds very slowly to new ones.

    ⇒ 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

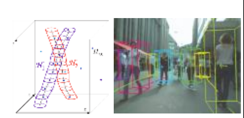
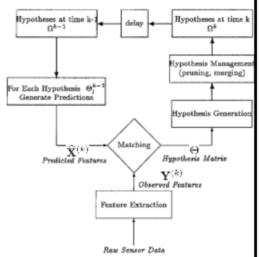


Image sources: Andreas Ess

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.

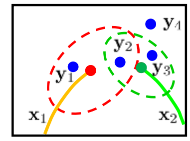


D. Reid, An Algorithm for Tracking Multiple Targets, IEEE Trans. Automatic Control, Vol. 24(6), pp. 843-854, 1979.

Recap: Hypothesis Generation

- Create hypothesis matrix of the feasible associations

$$\Theta = \begin{array}{c} \mathbf{x}_1 \; \mathbf{x}_2 \; \mathbf{x}_{fa} \; \mathbf{x}_{nt} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{array}{c} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \mathbf{y}_4 \end{array}$$



- Interpretation
  - Columns represent tracked objects, rows encode measurements
  - A non-zero element at matrix position (i,j) denotes that measurement $\mathbf{y}_i$ is contained in the validation region of track $\mathbf{x}_j$.
  - Extra column $\mathbf{x}_{fa}$ for association as *false alarm*.
  - Extra column $\mathbf{x}_{nt}$ for association as *new track*.
  - Enumerate all *assignments* that are consistent with this matrix.

Recap: Assignments

Z_j	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_{fa}$	$\mathbf{x}_{nt}$
$\mathbf{y}_1$	0	0	1	0
$\mathbf{y}_2$	1	0	0	0
$\mathbf{y}_3$	0	1	0	0
$\mathbf{y}_4$	0	0	0	1

- 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 $\mathbf{x}_{fa}$, $\mathbf{x}_{nt}$

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:

$$\begin{aligned} p(\Omega_j^{(k)} | \mathbf{Y}^{(k)}) &= p(Z_j^{(k)}, \Omega_{p(j)}^{(k-1)} | \mathbf{Y}^{(k)}) \\ &\overset{Bayes}{=} \eta p(\mathbf{Y}^{(k)} | Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) p(Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) \\ &= \eta p(\mathbf{Y}^{(k)} | Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}) p(Z_j^{(k)} | \Omega_{p(j)}^{(k-1)}) p(\Omega_{p(j)}^{(k-1)}) \end{aligned}$$

Normalization factor — Measurement likelihood — Prob. of assignment set — Prob. of parent

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 $\hat{\mathbf{x}}_j^{(k)}$ with innovation covariance $\widehat{\mathbf{\Sigma}}_j^{(k)}$.
  - 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

$$\begin{aligned} p\left(\mathbf{Y}^{(k)} | Z_j^{(k)}, \Omega_{p(j)}^{(k-1)}\right) &= \prod_{i=1}^{M_k} \mathcal{N}\left(\mathbf{y}_i^{(k)}; \hat{\mathbf{x}}_j, \widehat{\mathbf{\Sigma}}_j^{(k)}\right)^{\delta_i} W^{-(1-\delta_i)} \\ &= W^{-(N_{fal}+N_{new})} \prod_{i=1}^{M_k} \mathcal{N}\left(\mathbf{y}_i^{(k)}; \hat{\mathbf{x}}_j, \widehat{\mathbf{\Sigma}}_j^{(k)}\right)^{\delta_i} \end{aligned}$$

Recap: Probability of an Assignment Set

$$p(Z_j^{(k)} | \Omega_{p(j)}^{(k-1)})$$

- 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

$$p(N_{det} | \Omega_{p(j)}^{(k-1)}) = \binom{N}{N_{det}} p_{det}^{N_{det}} (1 - p_{det})^{(N - N_{det})}$$

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 $\lambda_{new}W$

$$\begin{aligned} p(N_{det}, N_{fal}, N_{new} | \Omega_{p(j)}^{(k-1)}) &= \binom{N}{N_{det}} p_{det}^{N_{det}} (1 - p_{det})^{(N - N_{det})} \\ &\quad \cdot \mu(N_{fal}; \lambda_{fal}W) \cdot \mu(N_{new}; \lambda_{new}W) \end{aligned}$$

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

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

$\Rightarrow$ 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

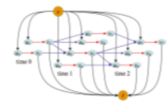
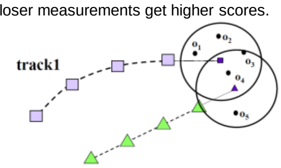


image source: [Zhang, Li, Nevatia, CVPR'08]

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.



	ai1	ai2
1	3.0	
2	5.0	
3	6.0	1.0
4	9.0	8.0
5		3.0

  - Choose at most one match in each row and column to maximize sum of scores

Slide credit: Robert Collins

Recap: Linear Assignment Problem

- Formal definition
  - Maximize $\sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij} z_{ij}$

subject to
$$\sum_{j=1} z_{ij} = 1;\; i = 1,2,\ldots,N$$
$$\sum_{i=1} z_{ij} = 1;\; j = 1,2,\ldots,M$$
$$z_{ij} \in \{0,1\}$$

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. $\arg\min_{z_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij} z_{ij}$

Slide adapted from Robert Collins

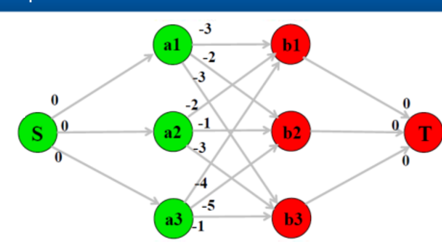
Recap: Optimal Solution

- 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 $\mathcal{O}(N^3)$.

  $\Rightarrow$ If you need LAP, you should use it.

Slide credit: Robert Collins

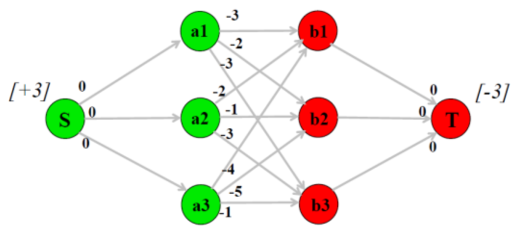
Recap: Min-Cost Flow



- 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.

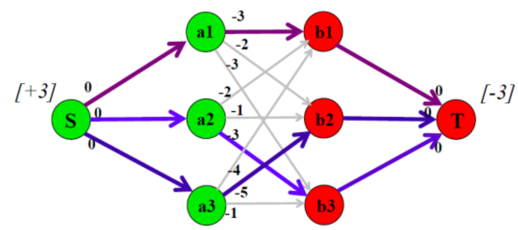
Slide credit: Robert Collins

Recap: Min-Cost Flow



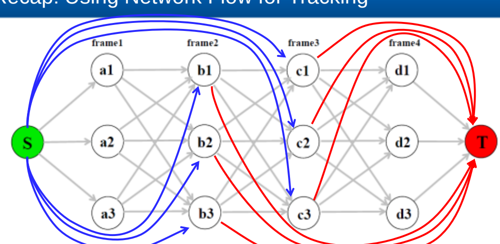
- Conversion into flow graph
  - Pump N units of flow from source to sink.
  - Internal nodes pass on flow ($\sum$ flow in $=$ $\sum$ flow out).
  - $\Rightarrow$ Find the optimal paths along which to ship the flow.

Recap: Min-Cost Flow

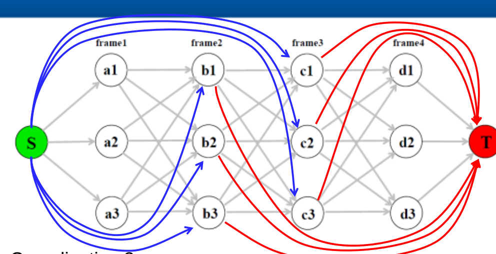


- Conversion into flow graph
  - Pump N units of flow from source to sink.
  - Internal nodes pass on flow ($\sum$ flow in $=$ $\sum$ flow out).
  - $\Rightarrow$ Find the optimal paths along which to ship the flow.

Recap: Using Network Flow for Tracking



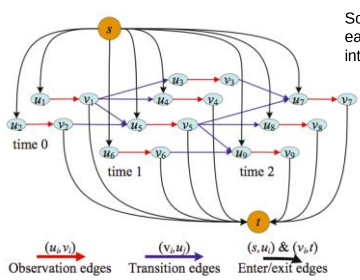
- Complication 1
  - Tracks can start later than frame1 (and end earlier than frame4)
  - $\Rightarrow$ Connect the source and sink nodes to all intermediate nodes.



- 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

$$\mathcal{T}* = \underset{\mathcal{T}}{\textbf{argmin}} \sum_i C_{in,i} f_{in,i} + \sum_i C_{i,out} f_{i,out}$$

$$+ \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_i f_i$$

- subject to
  - Flow conservation at all nodes

$$f_{in,i} + \sum_j f_{j,i} = f_i = f_{out,i} + \sum_j f_{i,j} \;\forall i$$

  - Edge capacities

$$f_i \leq 1$$

Min-Cost Formulation

- Objective Function

IN OUT

$$\mathcal{T}* = \underset{\mathcal{T}}{\mathbf{argmin}} \sum_i C_{in,i} f_{in,i} + \sum_i C_{i,out} f_{i,out}$$

$$+ \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_i f_i$$

TRANSITION

Likelihood of the detection

- Equivalent to Maximum A-Posteriori formulation

$$\mathcal{T}* = \underset{\mathcal{T}}{\mathbf{argmax}} \prod_i P(\mathbf{o}_i|\mathcal{T}) P(\mathcal{T})$$

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

Independence assumption + Markov

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

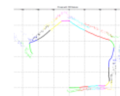
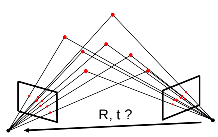


image source: [Clemente et al., RSS 2007]

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)

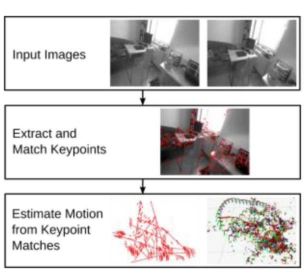
$$p(\mathbf{I}_2 \mid \mathbf{I}_1, \boldsymbol{\xi}) \quad \longrightarrow \quad E(\boldsymbol{\xi}) = \int_{\mathbf{u}\in\Omega} |\mathbf{I}_1(\mathbf{u}) - \mathbf{I}_2(\omega(\mathbf{u}, \boldsymbol{\xi}))| \, d\mathbf{u}$$

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

$$p(\mathbf{Y}_2 \mid \mathbf{Y}_1, \boldsymbol{\xi}) \quad \longrightarrow \quad E(\boldsymbol{\xi}) = \sum_i |\mathbf{y}_{1,i} - \omega(\mathbf{y}_{2,i}, \boldsymbol{\xi})|$$

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)



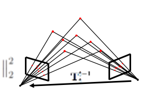
Images from Jakob Engel

Recap: Motion Estimation from Point Correspondences

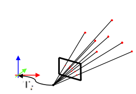
- 2D-to-2D
  - Reproj. error:

$$E\left(\mathbf{T}_t^{t-1}, X\right) = \sum_{i=1}^{N} \|\bar{\mathbf{y}}_{t,i} - \pi\left(\bar{\mathbf{x}}_i\right)\|_2^2 + \|\bar{\mathbf{y}}_{t-1,i} - \pi\left(\mathbf{T}_t^{t-1}\bar{\mathbf{x}}_i\right)\|_2^2$$

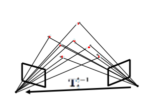
  - Introduced linear algorithm: 8-point



- 2D-to-3D
  - Reprojection error: $E(\mathbf{T}_t) = \sum_{i=1}^{N} \|\mathbf{y}_{t,i} - \pi(\mathbf{T}_t \bar{\mathbf{x}}_i)\|_2^2$
  - Introduced linear algorithm: DLT PnP



- 3D-to-3D
  - Reprojection error: $E\left(\mathbf{T}_t^{t-1}\right) = \sum_{i=1}^{N} \|\overline{\mathbf{x}}_{t-1,i} - \mathbf{T}_t^{t-1}\overline{\mathbf{x}}_{t,i}\|_2^2$
  - 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
     $\tilde{\mathbf{y}}_i \mathbf{E} \tilde{\mathbf{y}}'_i = \mathbf{a}_i \mathbf{E}_s = 0 \longrightarrow \mathbf{A}\mathbf{E}_s = 0 \qquad \mathbf{A} = (\mathbf{a}_1^\top, \ldots, \mathbf{a}_N^\top)^\top$
     using Kronecker product $\mathbf{a}_i = \tilde{\mathbf{y}}_i \otimes \tilde{\mathbf{y}}'_i$ and $\mathbf{E}_s = (e_{11}, e_{12}, e_{13}, \ldots, e_{33})^\top$
  2. Apply singular value decomposition (SVD) on $\mathbf{A} = \mathbf{U_A S_A V_A^\top}$ and unstack the 9th column of $\mathbf{V_A}$ into $\tilde{\mathbf{E}}$.
  3. Project the approximate $\tilde{\mathbf{E}}$ into the (normalized) essential space: Determine the SVD of $\tilde{\mathbf{E}} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \sigma_2, \sigma_3)\mathbf{V}^\top$ with $\mathbf{U}, \mathbf{V} \in \mathbf{SO}(3)$ and replace the singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3$ with 1,1,0 to find

$$\mathbf{E} = \mathbf{U}\,\mathrm{diag}(1,1,0)\,\mathbf{V}^\top$$

Recap: Eight-Point Algorithm cont.

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

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

- 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: $\|\mathbf{E}\| = \left\|\widehat{\mathbf{t}}\right\| = 1$
- 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: $\left\|\widehat{\mathbf{t}}\right\| = 0 \Rightarrow \|\mathbf{E}\| = 0$
- 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

$$r_{i,j} = \frac{\|\mathbf{x}_{t-2,t-1,i} - \mathbf{x}_{t-2,t-1,j}\|_2}{\|\mathbf{x}_{t-1,t,i} - \mathbf{x}_{t-1,t,j}\|_2}$$

  - 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 $\mathbf{T}_{0:t}$

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

Recap: Triangulation

- Goal: Reconstruct 3D point $\tilde{\mathbf{x}} = (x, y, z, w)^\top \in \mathbb{P}^3$ from 2D image observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ for known camera poses $\{\mathbf{T}_1, \ldots, \mathbf{T}_N\}$
- Linear solution: Find 3D point such that reprojections equal its projections

$$\mathbf{y}'_i = \pi(\mathbf{T}_i\tilde{\mathbf{x}}) = \begin{pmatrix} \frac{r_{11}x + r_{12}y + r_{13}z + t_x w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \\ \frac{r_{21}x + r_{22}y + r_{23}z + t_y w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \end{pmatrix}$$

  - Each image provides one constraint $\mathbf{y}_i - \mathbf{y}'_i = 0$
  - Leads to system of linear equations $\mathbf{A}\tilde{\mathbf{x}} = 0$, two approaches:
    - Set $w = 1$ and solve nonhomogeneous system
    - Find nullspace of $\mathbf{A}$ using SVD (this is what we did in CV I)
- Non-linear solution: Minimize least squares reprojection error (more accurate)

$$\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{y}'_i\|_2^2 \right\}$$

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
  3D: $\tilde{\mathbf{x}}_i = (x_i, y_i, z_i, w_i)^\top \in \mathbb{P}^3$ 2D: $\tilde{\mathbf{y}}_i = (x'_i, y'_i, w'_i)^\top \in \mathbb{P}^2$
  gives two constraints
  $$\begin{pmatrix} \mathbf{0} & -w'_i\tilde{\mathbf{x}}_i^\top & y'_i\tilde{\mathbf{x}}_i^\top \\ w'_i\tilde{\mathbf{x}}_i^\top & \mathbf{0} & -x'_i\tilde{\mathbf{x}}_i^\top \end{pmatrix} \begin{pmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{pmatrix} = \mathbf{0}$$
  through $\tilde{\mathbf{y}}_i \times (\mathbf{P}\tilde{\mathbf{x}}_i) = 0$
- Form linear system of equation $\mathbf{A}\mathbf{p} = \mathbf{0}$ 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 $\mathbf{T}_{0:t}$

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 $\mathbf{T}_k$ 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, $N \geq 3$
  $$X_{t-1} = \{\mathbf{x}_{t-1,1}, \ldots, \mathbf{x}_{t-1,N}\} \qquad X_t = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,N}\}$$
- Determine means of 3D point sets
  $$\boldsymbol{\mu}_{t-1} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{t-1,i} \qquad \boldsymbol{\mu}_t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{t,i}$$
- Determine rotation from
  $$\mathbf{A} = \sum_{i=1}^{N} (\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_t - \boldsymbol{\mu}_t)^\top \qquad \mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \qquad \mathbf{R}_{t-1}^t = \mathbf{V}\mathbf{U}^\top$$
- Determine translation as $\mathbf{t}_{t-1}^t = \boldsymbol{\mu}_t - \mathbf{R}_{t-1}^t\boldsymbol{\mu}_{t-1}$

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

Input: stereo image sequence $I_{0:t}^l, I_{0:t}^r$
Output: aggregated camera poses $\mathbf{T}_{0:t}$

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

Recap: Keypoint Detectors

- Corners
  - Image locations with locally prominent intensity variation
  - Intersections of edges
- Examples: Harris, FAST
- Scale-selection: Harris-Laplace

Harris Corners

- 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

DoG (SIFT) Blobs

Image source: Svetlana Lazebnik

Recap: RANSAC

- RANdom SAmple Consensus algorithm for robust estimation
- Algorithm:
  Input: data D, s required data points for fitting, success probability p, outlier ratio ϵ
  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

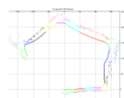


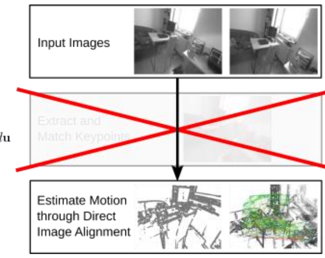
image source: [Clemente et al., RSS 2007]

Recap: Direct Visual Odometry Pipeline

- Avoid manually designed keypoint detection and matching
- Instead: direct image alignment

$$E(\xi) = \int_{\mathbf{u}\in\Omega} |\mathbf{I}_1(\mathbf{u}) - \mathbf{I}_2(\omega(\mathbf{u},\xi))|\, d\mathbf{u}$$

- Warping requires depth
  - RGB-D
  - Fixed-baseline stereo
  - Temporal stereo, tracking and (local) mapping

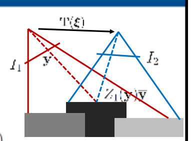


Slide credit: Jörg Stückler

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:

$$I_1(\mathbf{y}) = I_2(\pi(\mathbf{T}(\xi)Z_1(\mathbf{y})\overline{\mathbf{y}}))$$



  - Estimate the warp by minimizing the residuals (similar to LK alignment)

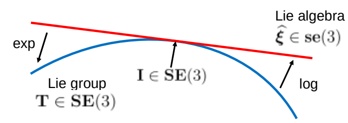
$$E(\xi) = \sum_{\mathbf{y}\in\Omega} \frac{r(\mathbf{y},\xi)^2}{\sigma_I^2} \qquad r(\mathbf{y},\xi) = I_1(\mathbf{y}) - I_2(\pi(\mathbf{T}(\xi)Z_1(\mathbf{y})\overline{\mathbf{y}}))$$

⇒ Non-linear least-squares problem (use second-order tools)

  - Important issue in practice: *How to parametrize the poses?*

Adapted from Jörg Stückler

Recap: Representing Motion using Lie Algebra se(3)



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

$\omega \in \mathbb{R}^3$

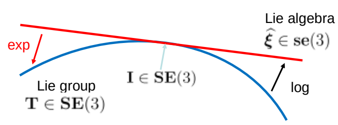
$\mathbf{v} \in \mathbb{R}^3$

$$\widehat{\xi} := \begin{pmatrix} \widehat{\omega} & \mathbf{v} \\ \mathbf{0} & 0 \end{pmatrix} \in \mathbb{R}^{4\times4}$$

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

Slide credit: Jörg Stückler

Recap: Exponential Map of SE(3)



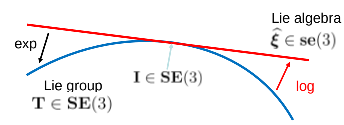
- The exponential map finds the transformation matrix for a twist:

$$\exp\left(\widehat{\xi}\right) = \begin{pmatrix} \exp(\widehat{\omega}) & \mathbf{A}\mathbf{v} \\ \mathbf{0} & 1 \end{pmatrix}$$

$$\exp(\widehat{\omega}) = \mathbf{I} + \frac{\sin|\omega|}{|\omega|}\widehat{\omega} + \frac{1-\cos|\omega|}{|\omega|^2}\widehat{\omega}^2 \qquad \mathbf{A} = \mathbf{I} + \frac{1-\cos|\omega|}{|\omega|^2}\widehat{\omega} + \frac{|\omega| - \sin|\omega|}{|\omega|^3}\widehat{\omega}^2$$

Slide credit: Jörg Stückler

Recap: Logarithm Map of SE(3)



- The logarithm maps twists to transformation matrices:

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

$$\log(\mathbf{R}) = \frac{|\omega|}{2\sin|\omega|}\left(\mathbf{R} - \mathbf{R}^T\right) \qquad |\omega| = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right)$$

Slide credit: Jörg Stückler

Recap: Working with Twist Coordinates

- Let's define the following notation:
  - Inversion of hat operator: $\begin{pmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}^{\vee} = (\omega_1\ \omega_2\ \omega_3\ v_1\ v_2\ v_3)^{\top}$
  - Conversion: $\boldsymbol{\xi}(\mathbf{T}) = (\log(\mathbf{T}))^{\vee}, \quad \mathbf{T}(\boldsymbol{\xi}) = \exp(\widehat{\boldsymbol{\xi}})$
  - Pose inversion: $\boldsymbol{\xi}^{-1} = \log(\mathbf{T}(\boldsymbol{\xi})^{-1}) = -\boldsymbol{\xi}$
  - Pose concatenation: $\boldsymbol{\xi}_1 \oplus \boldsymbol{\xi}_2 = (\log(\mathbf{T}(\boldsymbol{\xi}_2)\,\mathbf{T}(\boldsymbol{\xi}_1)))^{\vee}$
  - Pose difference: $\boldsymbol{\xi}_1 \ominus \boldsymbol{\xi}_2 = \left(\log\left(\mathbf{T}(\boldsymbol{\xi}_2)^{-1}\,\mathbf{T}(\boldsymbol{\xi}_1)\right)\right)^{\vee}$

Recap: Optimization with Twist Coordinates

- Twists provide a minimal local representation without singularities
- Since $\mathbf{SE}(3)$ is a smooth manifold, we can decompose transformations in each optimization step into the transformation itself and an infinitesimal increment

$$\mathbf{T}(\boldsymbol{\xi}) = \mathbf{T}(\boldsymbol{\xi}) \exp\left(\widehat{\delta\boldsymbol{\xi}}\right) = \mathbf{T}(\delta\boldsymbol{\xi} \oplus \boldsymbol{\xi})$$

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

$$\delta\boldsymbol{\xi}^* = \mathbf{0} - \eta \nabla_{\delta\boldsymbol{\xi}} E(\boldsymbol{\xi}_i, \delta\boldsymbol{\xi})$$

$$\mathbf{T}(\boldsymbol{\xi}_{i+1}) = \mathbf{T}(\boldsymbol{\xi}_i) \exp\left(\widehat{\delta\boldsymbol{\xi}^*}\right)$$

Algorithm: Direct RGB-D Visual Odometry

Input: RGB-D image sequence $I_{0:t}, Z_{0:t}$
Output: aggregated camera poses $\mathbf{T}_{0:t}$

Algorithm:
For each current RGB-D image I_k, Z_k :

1. Estimate relative camera motion $\mathbf{T}_k^{k-1}$ 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 $\mathbf{T}_k = \mathbf{T}_{k-1}\mathbf{T}_k^{k-1}$

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

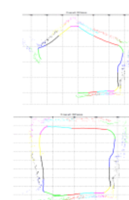
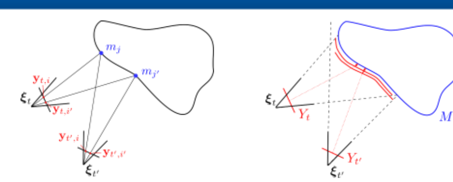


image source: [Clemente et al., RSS 2007]

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 $I^l_{0:t} = \{I^l_0, \ldots, I^l_t\}$, $I^r_{0:t} = \{I^r_0, \ldots, I^r_t\}$
  - 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 $\mathbf{T}_t \in \mathbf{SE}(3)$ in world reference frame. For convenience, we also write $\boldsymbol{\xi}_t = \boldsymbol{\xi}(\mathbf{T}_t)$
  - Environment map M

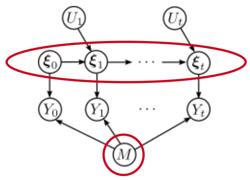
Recap: Map Observations in Visual SLAM



With Y_t we denote observations of the environment map in image I_t, e.g.,
- Indirect point-based method: $Y_t = \{\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,N}\}$ (2D or 3D image points)
- Direct RGB-D method: $Y_t = \{I_t, Z_t\}$ (all image pixels)
- ...

- Involves data association to map elements $M = \{m_1, \ldots, m_S\}$
  - We denote correspondences by $c_{t,i} = j,\ 1 \le i \le N,\ 1 \le j \le S$

Recap: Probabilistic Formulation of Visual SLAM

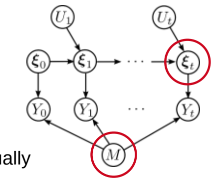


- SLAM posterior probability: $p(\boldsymbol{\xi}_{0:t}, M \mid Y_{0:t}, U_{1:t})$
- Observation likelihood: $p(Y_t \mid \boldsymbol{\xi}_t, M)$
- State-transition probability: $p(\boldsymbol{\xi}_t \mid \boldsymbol{\xi}_{t-1}, U_t)$

Recap: Online SLAM Methods

- Marginalize out previous poses

$$p(\boldsymbol{\xi}_t, M \mid Y_{0:t}, U_{1:t}) =$$
$$\int \cdots \int p(\boldsymbol{\xi}_{0:t}, M \mid Y_{0:t}, U_{1:t})\, d\boldsymbol{\xi}_{t-1} \ldots d\boldsymbol{\xi}_0$$



- 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

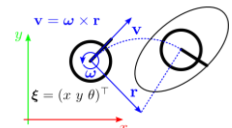
$$\mathbf{x}_t = \begin{pmatrix} \boldsymbol{\xi}_t \\ \mathbf{m}_{t,1} \\ \vdots \\ \mathbf{m}_{t,S} \end{pmatrix} \qquad \Sigma_t = \begin{pmatrix} \Sigma_{t,\xi\xi} & \Sigma_{t,\xi\mathbf{m}_1} & \cdots & \Sigma_{t,\xi\mathbf{m}_S} \\ \Sigma_{t,\mathbf{m}_1\xi} & \Sigma_{t,\mathbf{m}_1\mathbf{m}_1} & \cdots & \Sigma_{t,\mathbf{m}_1\mathbf{m}_S} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{t,\mathbf{m}_S\xi} & \Sigma_{t,\mathbf{m}_S\mathbf{m}_1} & \cdots & \Sigma_{t,\mathbf{m}_S\mathbf{m}_S} \end{pmatrix}$$

$$= \begin{pmatrix} \Sigma_{t,\xi\xi} & \Sigma_{t,\xi\mathbf{m}} \\ \Sigma_{t,\mathbf{m}\xi} & \Sigma_{t,\mathbf{mm}} \end{pmatrix}$$

Recap: 2D EKF-SLAM State-Transition Model

- State/control variables

$$\boldsymbol{\xi}_t = (x_t\ y_t\ \theta_t)^\top \qquad \mathbf{m}_{t,j} = (m_{t,j,x}\ m_{t,j,y})^\top$$
$$\mathbf{u}_t = (v_t\ \omega_t)^\top = (\|\mathbf{v}\|_2\ \ \|\boldsymbol{\omega}\|_2)^\top$$



- State-transition model
  - Pose:

$$\boldsymbol{\xi}_t = g_\xi(\boldsymbol{\xi}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_{\xi,t} \qquad \boldsymbol{\epsilon}_{\xi,t} \sim \mathcal{N}(0, \Sigma_{d_t,\xi})$$

$$g_\xi(\boldsymbol{\xi}_{t-1}, \mathbf{u}_t) = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta_{t-1} + \frac{v_t}{\omega_t}\sin(\theta_t + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta_{t-1} - \frac{v_t}{\omega_t}\cos(\theta_t + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

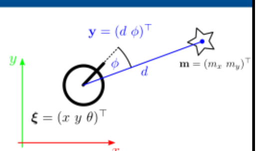
  - Landmarks: $\mathbf{m}_t = g_{\mathbf{m}}(\mathbf{m}_{t-1}) = \mathbf{m}_{t-1}$
  - Combined:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{d_t}) \qquad g(\mathbf{x}_{t-1}, \mathbf{u}_t) = \begin{pmatrix} g_\xi(\boldsymbol{\xi}_{t-1}, \mathbf{u}_t) \\ g_{\mathbf{m}}(\mathbf{m}_{t-1}) \end{pmatrix} \qquad \Sigma_{d_t} = \begin{pmatrix} \Sigma_{d_t,\xi} & 0 \\ 0 & 0 \end{pmatrix}$$

Recap: 2D EKF-SLAM Observation Model

- State/measurement variables

$$\mathbf{y}_t = (d_t\ \phi_t)^\top \qquad \mathbf{m}_{t,j} = (m_{t,j,x}\ m_{t,j,y})^\top$$



- Observation model:

$$\mathbf{y}_t = h(\boldsymbol{\xi}_t, \mathbf{m}_{t,c_t}) + \boldsymbol{\delta}_t \qquad \boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{m_t})$$

$$h(\boldsymbol{\xi}_t, \mathbf{m}_{t,c_t}) = \begin{pmatrix} \left\|\mathbf{m}^{\text{rel}}_{t,c_t}\right\|_2 \\ \operatorname{atan2}\left(\mathbf{m}^{\text{rel}}_{t,c_t,y}, \mathbf{m}^{\text{rel}}_{t,c_t,x}\right) \end{pmatrix}$$

$$\mathbf{m}^{\text{rel}}_{t,c_t} := \mathbf{R}(-\theta_t)\left(\mathbf{m}_{t,c_t} - (x_t\ y_t)^\top\right)$$

Recap: State Initialization

- First frame:
  - Anchor reference frame at initial pose $\quad \mathbf{x}_0 = \mathbf{0}$
  - Set pose covariance to zero $\quad \Sigma^-_{0,\xi\xi} = 0$
- New landmark:
  - Initial position unknown $\quad \Sigma^-_{0,\xi\mathbf{m}} = \Sigma^-_{0,\mathbf{m}\xi} = 0$
  - Initialize mean at zero
  - Initialize covariance to infinity (large value) $\quad \Sigma^-_{0,\mathbf{mm}} = \infty\mathbf{I}$

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

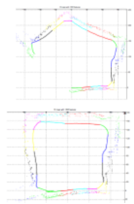
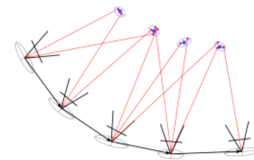


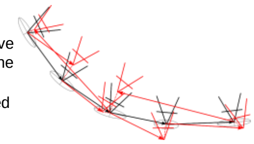
image source: [Clemente et al., RSS 2007]

Recap: Full SLAM Approaches

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



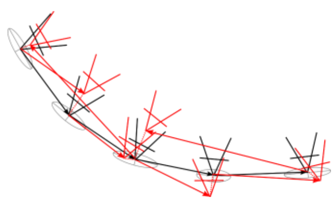
- Pose graph optimization:
  - Optimization of poses from relative pose constraints deduced from the image observations
  - Map recovered from the optimized poses



Slide credit: Jörg Stückler

Pose Graph Optimization

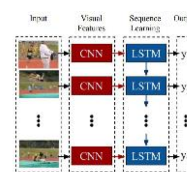
- Optimization of poses
  - From relative pose constraints deduced from the image observations
  - Map recovered from the optimized poses
- Deduce relative constraints between poses from image observations, e.g.,
  - 8-point algorithm
  - Direct image alignment



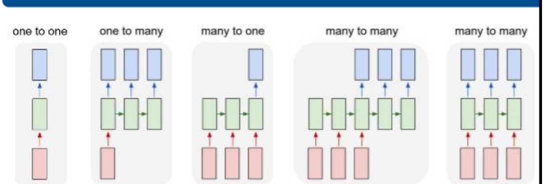
Slide credit: Jörg Stückler

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



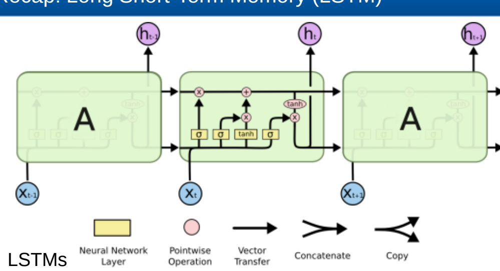
Recap: Recurrent Networks



- Feed-forward networks
  - Simple neural network structure: 1-to-1 mapping of inputs to outputs
- Recurrent Neural Networks
  - Generalize this to arbitrary mappings

Image source: Andrej Karpathy

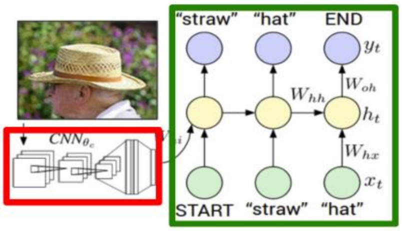
Recap: Long Short-Term Memory (LSTM)



- 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/

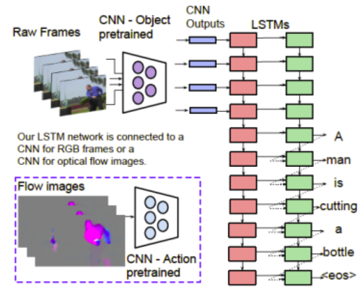
Recap: Image Tagging



- Simple combination of CNN and RNN
  - Use CNN to define initial state $\mathbf{h}_0$ of an RNN.
  - Use RNN to produce text description of the image.

Slide adapted from Andrei Karpathy

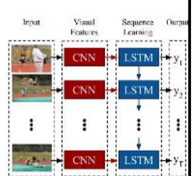
Recap: Video to Text Description



Source: Subhashini Venugopalan, ICCV'15

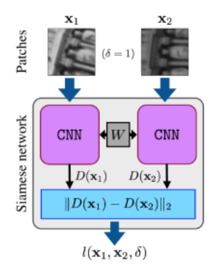
Course Outline

- 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
  - …

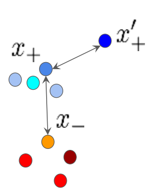


Recap: Metric Learning – Contrastive Loss

- Mapping an image to a metric embedding space
  - Metric space: distance relationship = class membership

$$\|f(x) - f(x_+)\| \to 0$$

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

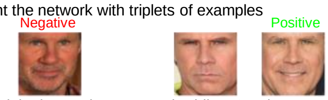


Yi et al., LIFT: Learned Invariant Feature Transform, ECCV 16

Slide credit: Christopher Choy

Recap: Metric Learning – Triplet Loss

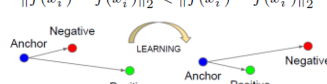
- Learning a discriminative embedding
  - Present the network with triplets of examples



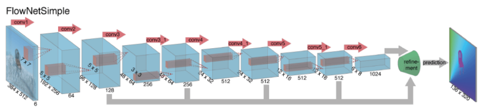
  - Apply triplet loss to learn an embedding $f(\cdot)$ that groups the positive example closer to the anchor than the negative one.

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$

⇒ Used ntification

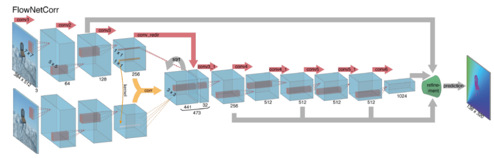


Recap: FlowNet – FlowNetSimple Design



- 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…

Recap: FlowNet – FlowNetCorr Design

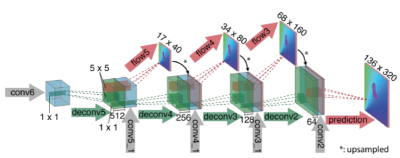


- Correlation network
  - Central idea: compute a correlation score between two feature maps

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

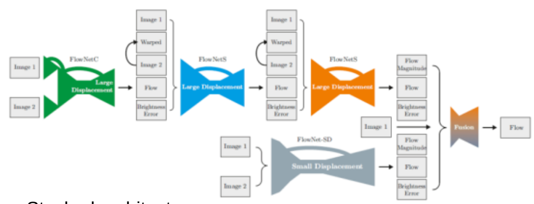
  - Then refine the correlation scores and turn them into flow predictions

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

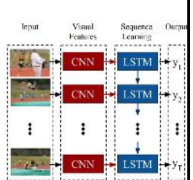
Recap: FlowNet 2.0 Improved Design



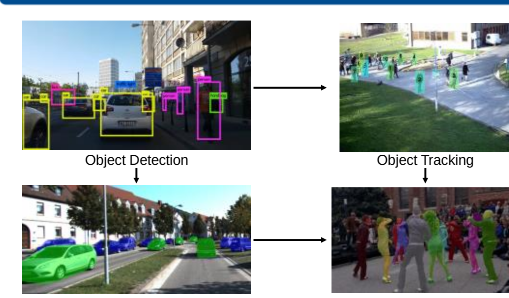
- Stacked architecture
  - Several instances of FlowNetC and FlowNetS stacked together to estimate large-displacement flow
  - Sub-network specialized on small motions
  - Fusion layer

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



Any More Questions?

Good luck for the exam!