Computer Vision 2 – Lecture 3

Tracking by Online Classification (25.04.2016)

Prof. Dr. Bastian Leibe, Dr. Jörg Stückler
leibe@vision.rwth-aachen.de, stueckler@vision.rwth-aachen.de

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

Content of the Lecture

- Single-Object Tracking
  - *Background modeling*
  - Template based tracking
  - *Color based tracking*
  - *Contour based tracking*
  - **Tracking by online classification**
  - Tracking-by-detection
- Bayesian Filtering
- Multi-Object Tracking
- Visual Odometry
- Visual SLAM & 3D Reconstruction

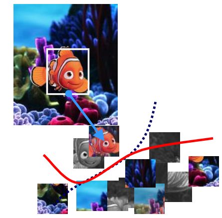
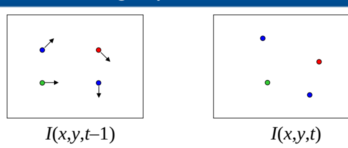


Image source: Helmut Grabner, Disney/Pixar

Recap: Estimating Optical Flow



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

Recap: Lucas-Kanade Optical Flow

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

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

- Minimum least squares solution given by solution of

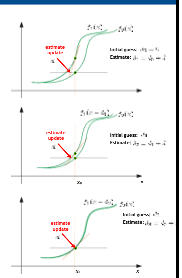
$$\underset{2\times 2}{(A^T A)} \ \underset{2\times 1}{d} = \underset{2\times 1}{A^T b}$$

Recall the Harris detector!

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

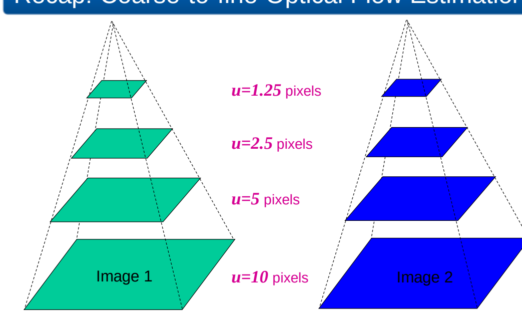
Recap: Iterative Refinement

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

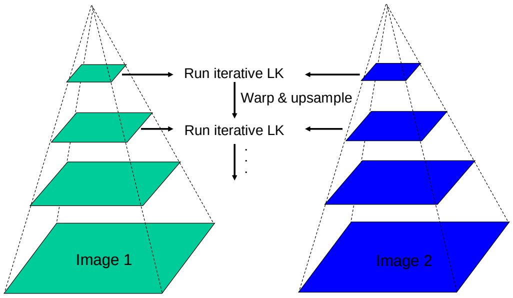


- Iterative procedure
  - Results in subpixel accurate localization.
  - Converges for small displacements.

Recap: Coarse-to-fine Optical Flow Estimation



Recap: Coarse-to-fine Optical Flow Estimation



Gaussian pyramid of image 1 | Gaussian pyramid of image 2

Slide credit: Steve Seitz

Recap: Shi-Tomasi Feature Tracker (→KLT)

- Idea
  - Find good features using eigenvalues of second-moment matrix
  - Key idea: "good" features to track are the ones that can be tracked reliably.
- Frame-to-frame tracking
  - Track with LK and a pure *translation* motion model.
  - More robust for small displacements, can be estimated from smaller neighborhoods (e.g., 5×5 pixels).
- Checking consistency of tracks
  - *Affine* registration to the first observed feature instance.
  - Affine model is more accurate for larger displacements.
  - Comparing to the first frame helps to minimize drift.

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 $\Delta\mathbf{p}$

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

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

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

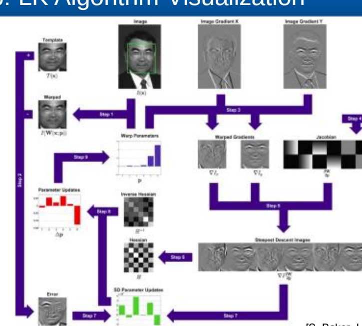
Gradient ∇I | **Jacobian** $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ | Increment parameters to solve for $\Delta\mathbf{p}$

Slide credit: Robert Collins

Recap: Generalized LK Algorithm

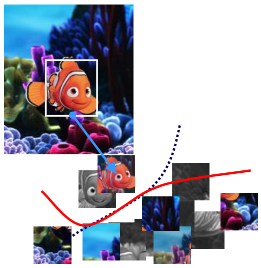
- Iterate
  - Warp I to obtain $I(\mathbf{W}([x,y];\mathbf{p}))$
  - Compute the error image $T([x,y]) - I(\mathbf{W}([x,y];\mathbf{p}))$
  - Warp the gradient ∇I with $\mathbf{W}([x,y];\mathbf{p})$
  - Evaluate $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $([x,y];\mathbf{p})$ (**Jacobian**)
  - 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: LK Algorithm Visualization



[S. Baker, I. Matthews, IJCV'04]

Today: Tracking by Online Classification



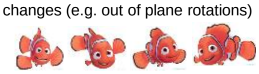
Can Machine Learning solve the problem for us?

Topics of This Lecture

- **Tracking by Online Classification**
  - Motivation
- Recap: Boosting for Detection
  - AdaBoost
  - Viola-Jones Detector
- Extension to Online Classification
  - Online Boosting
  - Online Feature Selection
  - Results
- Extensions
  - Problem: Drift
  - Drift-compensation strategies

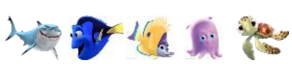
Tracking Requirements

- Adaptivity
  - Appearance changes (e.g. out of plane rotations)



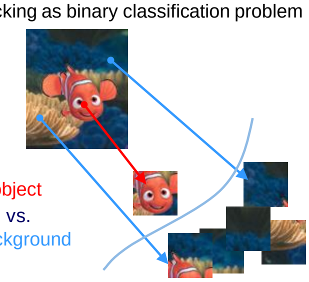
- Robustness
  - Occlusions, cluttered background, illumination conditions

- Generality
  - Any object



Tracking as Classification

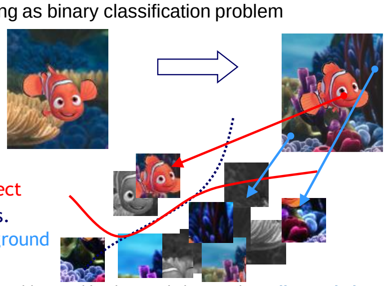
- Tracking as binary classification problem



object
vs.
background

Tracking as Classification

- Tracking as binary classification problem



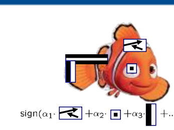
object
vs.
background

- Handle object <u>and</u> background changes by **online updating**

Idea: Use Boosting for Feature Selection

Object Detector

Fixed training set
General object detector



Boosting for Feature Selection

P. Viola, M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. CVPR'01.

Object Tracker

On-line update
Object vs. Background

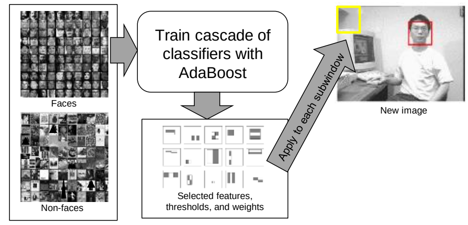
On-Line Boosting for Feature Selection

H. Grabner, H. Bischof. On-line Boosting and Vision. CVPR'06.

Topics of This Lecture

- Tracking by Online Classification
  - Motivation
- **Recap: Boosting for Detection**
  - AdaBoost
  - Viola-Jones Detector
- Extension to Online Classification
  - Online Boosting
  - Online Feature Selection
  - Results
- Extensions
  - Problem: Drift
  - Drift-compensation strategies

Recap: Viola-Jones Face Detector



- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade (6061 features in final layer)
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Slide credit: Kristen Grauman

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:

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

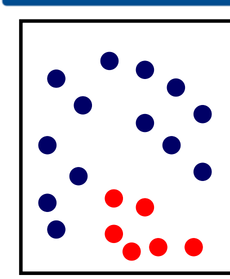
Recap: AdaBoost – Algorithm

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

Topics of This Lecture

- Tracking by Online Classification
  - Motivation
- Recap: Boosting for Detection
  - AdaBoost
  - Viola-Jones Detector
- **Extension to Online Classification**
  - Online Boosting
  - Online Feature Selection
  - Results
- Extensions
  - Problem: Drift
  - Drift-compensation strategies

Offline Boosting

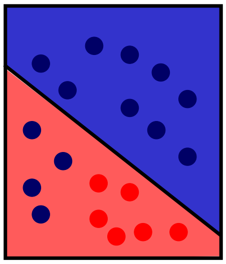


```
Given:
   - set of labeled training samples
   - weight distribution over them

Algorithm:
for n = 1 to N
   - train a weak classifier using
     samples and weight dist.
   - calculate error
   - calculate weight
   - update weight dist.
next
```


Slide credit: Helmut Grabner

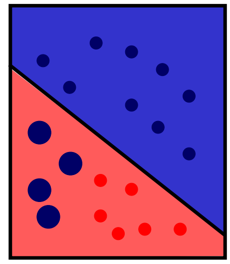
Offline Boosting



```
Given:
   - set of labeled training samples
   - weight distribution over them

Algorithm:
for n = 1 to N
   - train a weak classifier using
     samples and weight dist.
   - calculate error
   - calculate weight
   - update weight dist.
next
```


Offline Boosting

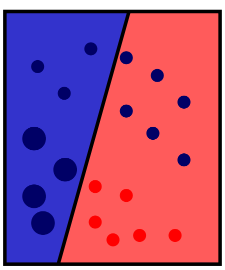


$\alpha_1 \cdot$ 

```
Given:
   - set of labeled training samples
   - weight distribution over them

Algorithm:
for n = 1 to N
   - train a weak classifier using
     samples and weight dist.
   - calculate error
   - calculate weight
   - update weight dist.
next
```

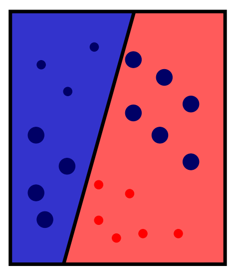

Offline Boosting



```
Given:
   - set of labeled training samples
   - weight distribution over them

Algorithm:
for n = 1 to N
   - train a weak classifier using
     samples and weight dist.
   - calculate error
   - calculate weight
   - update weight dist.
next
```


Offline Boosting

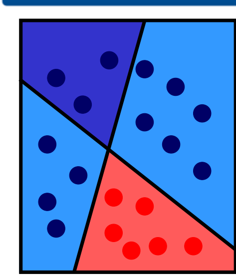


$\alpha_2 \cdot$ 

```
Given:
   - set of labeled training samples
   - weight distribution over them

Algorithm:
for n = 1 to N
   - train a weak classifier using
     samples and weight dist.
   - calculate error
   - calculate weight
   - update weight dist.
next
```


Offline Boosting



$= \alpha_1 \cdot$  $+ \alpha_2 \cdot$ 

```
Given:
   - set of labeled training samples
   - weight distribution over them

Algorithm:
for n = 1 to N
   - train a weak classifier using
     samples and weight dist.
   - calculate error
   - calculate weight
   - update weight dist.
next

Result:
```

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

From Offline to Online Boosting

- Goal
  - Formulate the algorithm such that we can present only 1 training sample at a time (and then forget about it).
  - ⇒ Dual problem: instead of keeping all samples and adding weak classifiers, keep a fixed set of weak classifiers and add samples.
- What changes?
  - Updating the classifiers online can be done easily.
    - Many classification approaches can use online updates.
  - Computing the classifier weights is also straightforward if we know the estimated error (which we can compute).

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.

From Offline to Online Boosting

off-line | on-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

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

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

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

on-line

Given:

for n = 1 to N

next

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

From Offline to Online Boosting

Only one training example to update the classifier

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

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

on-line

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

for n = 1 to N

next

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

From Offline to Online Boosting

Update importance for the current sample

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

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

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 importance weight λ

next

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

From Offline to Online Boosting

Online update the weak classifier

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

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

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 importance weight λ

next

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

From Offline to Online Boosting

off-line | Update errors and weights | **on-line**

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

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

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

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

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

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

on-line

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

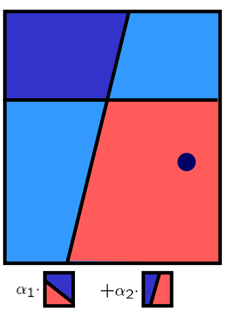
- initial importance $\lambda = 1$

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

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

Online Boosting



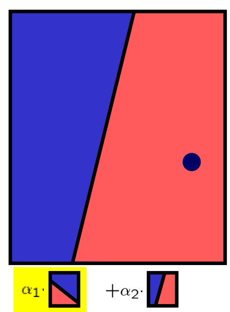
Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance

for n = 1 to N
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight

next

Online Boosting



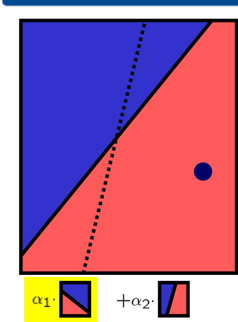
Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance

for n = 1 to N
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight

next

Online Boosting



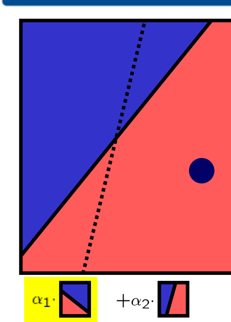
Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance

for n = 1 to N
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight

next

Online Boosting



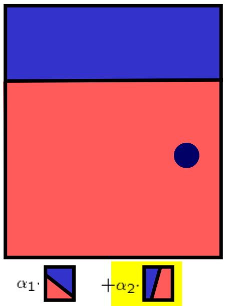
Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance

for n = 1 to N
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight

next

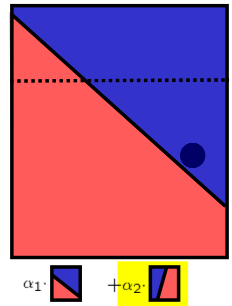
Online Boosting



Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance
for n = 1 to N
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight
next

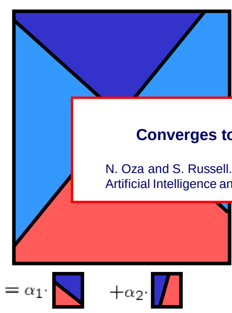
Online Boosting



Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance
for n = 1 to N
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight
next

Online Boosting



Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
...er using
- update importance weight
next

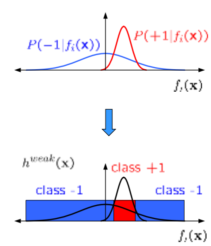
Converges to the off-line results...

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

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

Online Boosting for Feature Selection

- Each feature corresponds to a weak classifier.
- Features
  - Haar-like wavelets
  - Orientation histograms
  - Locally binary patterns (LBP)
- Fast computation using efficient data structures
  - integral images
  - integral histograms



F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. CVPR'05.

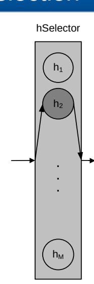
Online Boosting for Feature Selection

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

$\mathcal{H}^{weak} = \{h_1^{weak}, ..., h_M^{weak}\}$
$\mathcal{F} = \{f_1, ..., f_M\}$

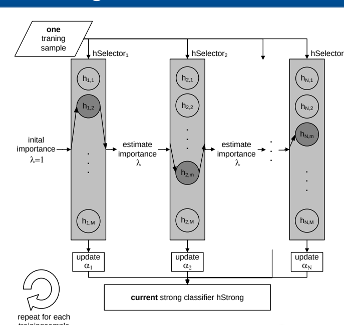
$h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x})$
$m = \arg\min_i e_i$

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

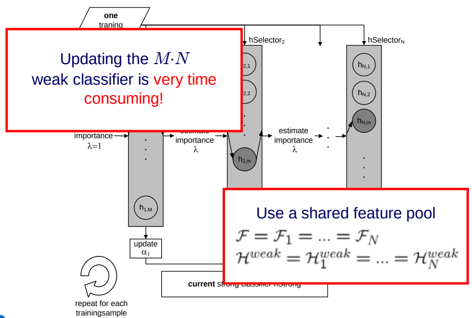


H. Grabner and H. Bischof. On-line boosting and vision. CVPR, 2006.

Online Boosting for Feature Selection



Online Boosting for Feature Selection



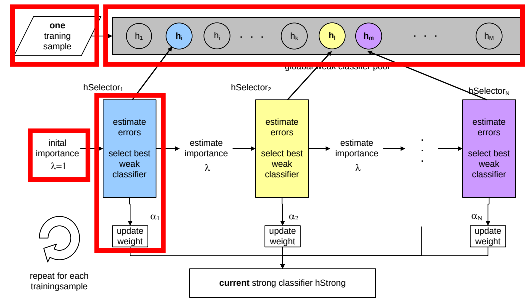
Updating the $M{\cdot}N$ weak classifier is very time consuming!

Use a shared feature pool

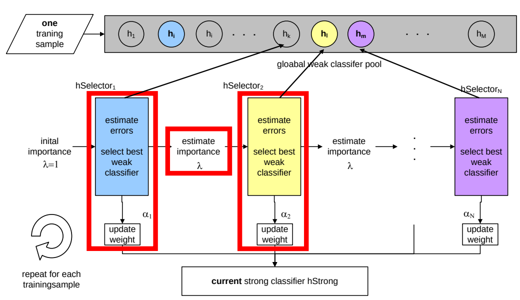
$\mathcal{F} = \mathcal{F}_1 = \dots = \mathcal{F}_N$

$\mathcal{H}^{weak} = \mathcal{H}_1^{weak} = \dots = \mathcal{H}_N^{weak}$

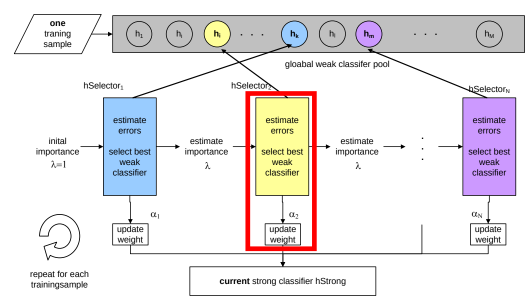
Direct Feature Selection



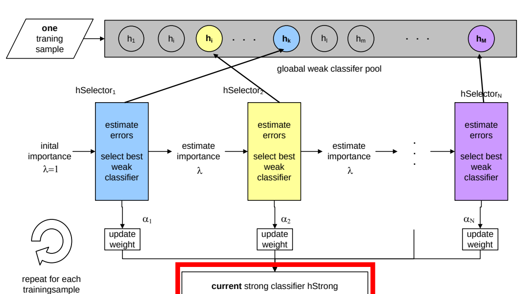
Direct Feature Selection



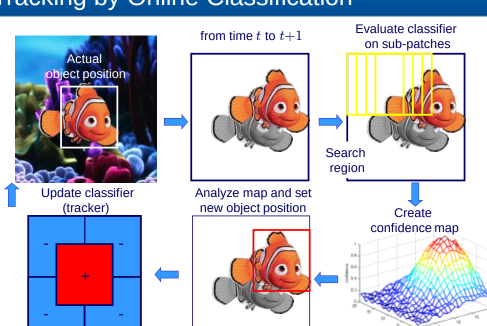
Direct Feature Selection



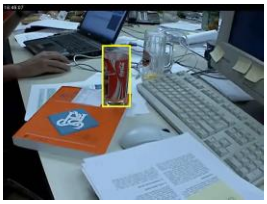
Direct Feature Selection

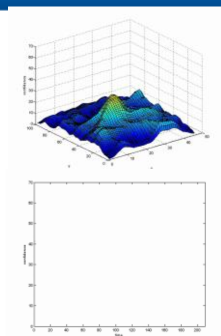


Tracking by Online Classification

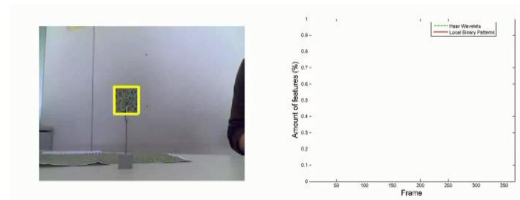


Tracking Results

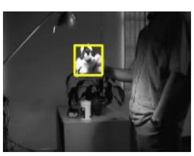




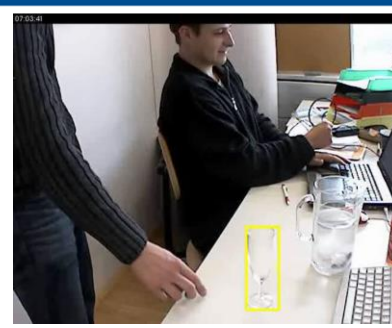
Online Feature Exchange



Additional Tracking Results



"Tracking the Invisible"



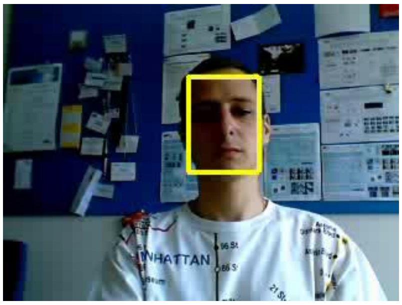
Summary: Tracking by Online Classification

- Interpret tracking as a classification problem
  - Continuously updating a classifier which discriminates the object from the background.
- Online Boosting
  - Adaptation of AdaBoost to process 1 training sample at a time.
  - Process sample by fixed set of classifiers to compute its importance weight.
  - Converges to the same result as Offline Boosting.
- Online Boosting for Feature Selection
  - Perform Boosting on Selectors instead of weak classifiers.
  - Each Selector chooses from a pool of weak classifiers.
  - Selected features and voting weights change over time.
  - Shared feature pool for real-time processing.

Topics of This Lecture

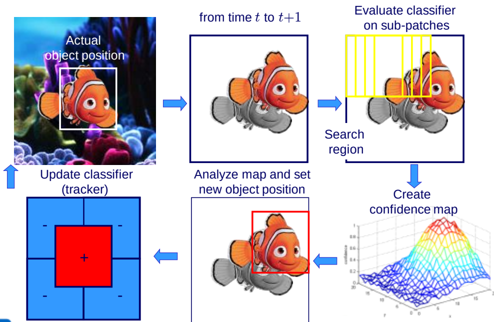
- **Tracking by Online Classification**
  - Motivation
- Recap: Boosting for Detection
  - AdaBoost
  - Viola-Jones Detector
- Extension to Online Classification
  - Online Boosting
  - Online Feature Selection
  - Results
- **Extensions**
  - Problem: Drift
  - Drift-compensation strategies

When Does It Fail...



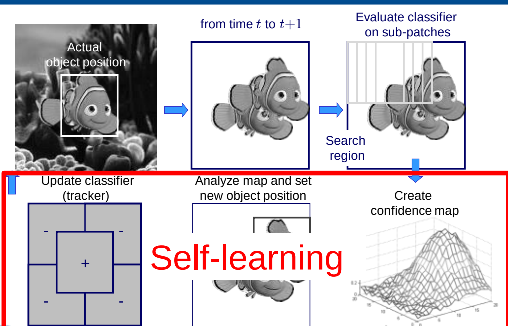
Slide credit: Helmut Grabner   Video source: Grabner et al., ECCV'08

Why Does It Fail?



Slide credit: Helmut Grabner   Image source: Disney/Pixar

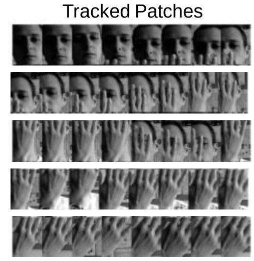
Why Does It Fail?



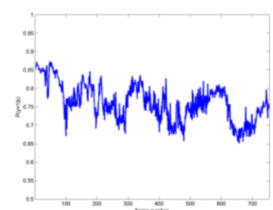
Slide credit: Helmut Grabner   Image source: Disney/Pixar

Drifting Due to Self-Learning Policy

Tracked Patches



Confidence



⇒ Not only does it drift, it also remains confident about it!

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

Self-Learning and Drift

- Drift
  - Major problem in all adaptive or self-learning trackers.
  - Difficulty: distinguish "allowed" appearance change due to lighting or viewpoint variation from "unwanted" appearance change due to drifting.
  - Cannot be decided based on the tracker confidence!
    - Since the confidence is always dependent on the learned model
    - Model may already be affected by drift when the confidence is measured.
  - Several approaches have been proposed to address this.

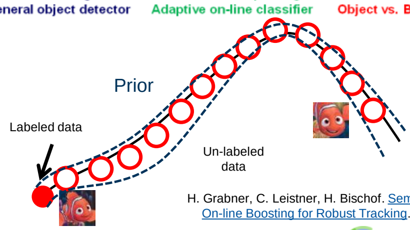
Strategy 1: Match Against Initialization

- Used mostly in low-level trackers (e.g., KLT)
  - Advantage: robustly catches drift
  - Disadvantage: cannot follow appearance changes

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

Strategy 2: Semi-Supervised Learning

Object Detector	Our approach	Object Tracker
Fixed Training set General object detector	Fixed Prior for updating an Adaptive on-line classifier	On-line update Object vs. Background



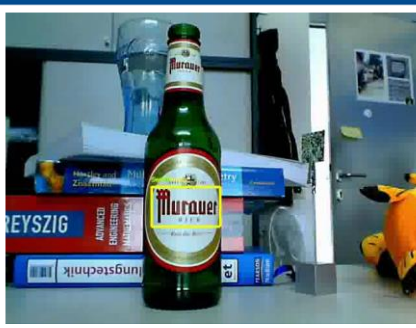
H. Grabner, C. Leistner, H. Bischof. Semi-Supervised On-line Boosting for Robust Tracking. ECCV'08.

Slide credit: Helmut Grabner

Tracking despite Occlusions

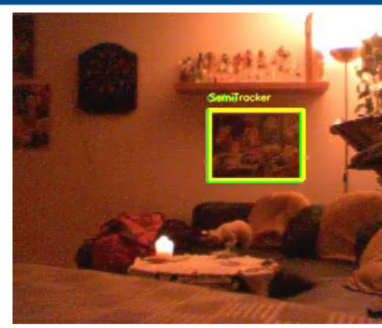
Video source: Grabner et al., ECCV'08

Object Disappearance



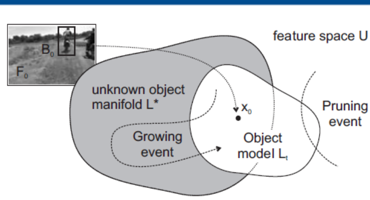
Video source: Grabner et al., ECCV'08

Long-Term Tracking (1h)



Video source: Grabner et al., ECCV'08

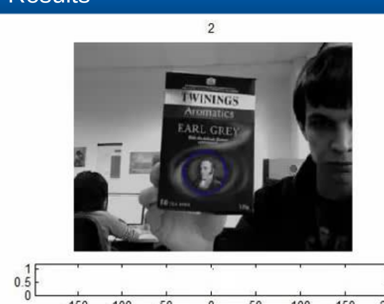
Strategy 3: Using Additional Cues



- Tracking-Learning-Detection
  - Combination of KLT and Tracking-by-Detection
  - Use a KLT tracker as additional cue to generate confident (positive and negative) training examples.
  - Learn an object detector on the fly using Online Random Ferns.

Z. Kalal, K. Mikolajczyk, J. Matas. Tracking-Learning-Detection. PAMI 2011.

TLD Results



Video source: Z. Kalal

Accumulated Training Examples

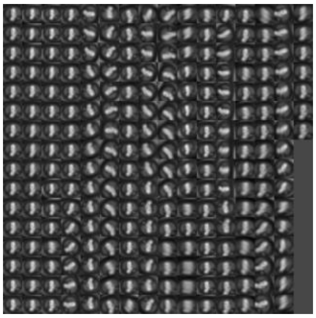


Image source: Z. Kalal

TLD Results

Video source: Z. Kalal

References and Further Reading

- The original Online AdaBoost paper
  - N. Oza and S. Russell. Online Bagging and Boosting. Artificial Intelligence and Statistics, 2001.
- Online Boosting for Tracking
  - H. Grabner, H. Bischof. On-line Boosting and Vision. CVPR'06.
- Semi-Supervised Boosting
  - H. Grabner, C. Leistner, H. Bischof. Semi-Supervised On-line Boosting for Robust Tracking. ECCV'08.
- Tracking-Learning-Detection
  - Z. Kalal, K. Mikolajczyk, J. Matas. Tracking-Learning-Detection. PAMI 2011.