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
Recap: Estimating Optical Flow

- **Optical Flow**: Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ between them.

- **Key assumptions**
  - **Brightness constancy**: projection of the same point looks the same in every frame.
  - **Small motion**: points do not move very far.
  - **Spatial coherence**: points move like their neighbors.
Recap: Lucas-Kanade Optical Flow

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

- Least squares problem:

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

- Minimum least squares solution given by solution of

$$\begin{bmatrix} \sum I_xI_x & \sum I_xI_y \\ \sum I_xI_y & \sum I_yI_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_xI_t \\ \sum I_yI_t \end{bmatrix}$$

Recall the Harris detector!
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.

Slide adapted from Steve Seitz
Recap: Coarse-to-fine Optical Flow Estimation

Gaussian pyramid of image 1

$u=10$ pixels

$u=5$ pixels

$u=2.5$ pixels

$u=1.25$ pixels

Gaussian pyramid of image 2

Image 2

Image 1

Slide credit: Steve Seitz
Recap: Coarse-to-fine Optical Flow Estimation

Run iterative LK
Warp & upsample
Run iterative LK

Gaussian pyramid of image 1

Image 1

Image 2

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., $5 \times 5$ pixels).

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

Recap: 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
    \[
    \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]^2
    \]
  – We assume that an initial estimate of $\mathbf{p}$ is known and iteratively solve for increments to the parameters $\Delta \mathbf{p}$:
    \[
    \arg\min_{\Delta \mathbf{p}} \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta \mathbf{p})) - T(\mathbf{x})]^2
    \]
Recap: Step-by-Step Derivation

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

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

  \[
  = I(W([x, y]; p_1, \ldots, p_n)) + \nabla I \begin{bmatrix}
  \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\
  \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n}
  \end{bmatrix} \begin{bmatrix}
  \Delta p_1 \\
  \Delta p_2 \\
  \vdots \\
  \Delta p_n
  \end{bmatrix}
  \]

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

  Increment parameters to solve for $\Delta p$
Recap: Generalized LK Algorithm

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

- **Until $\Delta p$ magnitude is negligible**
Recap: LK Algorithm Visualization

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

[S. Baker, I. Matthews, IJCV’04]
Today: Tracking by Online Classification

Can Machine Learning solve the problem for us?

Image source: Helmut Grabner, Disney/Pixar
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

Image source: Disney/Pixar
Tracking as Classification

- Tracking as binary classification problem

- Handle object and background changes by **online updating**
Idea: Use Boosting for Feature Selection

Object Detector

Fixed training set
General object detector

On-line update
Object vs. Background

Boosting for Feature Selection

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

On-Line Boosting for Feature Selection

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

Image source: Disney/Pixar

Slide credit: Helmut Grabner
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/]

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler
Slide credit: Kristen Grauman

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

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

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

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \ldots, N$.

2. For $m = 1, \ldots, M$ iterations
   a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function
      \[ J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n) \]
   
   b) Estimate the weighted error of this classifier on $X$:
      \[ \epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \]
   
   c) Calculate a weighting coefficient for $h_m(x)$:
      \[ \alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\} \]
   
   d) Update the weighting coefficients:
      \[ w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(h_m(x_n) \neq t_n) \} \]
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
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

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

Slide credit: Helmut Grabner
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
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:
\[ h^{\text{strong}}(x) = \text{sign} \left( \sum_{n=1}^{N} \alpha_n \cdot h_n^{\text{weak}}(x) \right) \]
• 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 $\lambda$ for the $n+1$ classifier.
  – Proven [Oza]: Given the same training set, Online Boosting converges to the same weak classifiers as Offline Boosting in the limit of $N \to \infty$ iterations.

From Offline to Online Boosting

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

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

next

\[ h_{\text{strong}}(x) = \text{sign} \left( \sum_{n=1}^{N} \alpha_n \cdot h_{n, weak}(x) \right) \]
From Offline to Online Boosting

**off-line**

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

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

next

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

**on-line**

**Given:**

for \( n = 1 \) to \( N \)

next

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

- set of labeled training samples
  \[ \mathcal{X} = \{ \langle x_1, y_1 \rangle, \ldots, \langle x_L, y_L \rangle \mid y_i \pm 1 \} \]

- weight distribution over them
  \[ D_0 = 1/L \]

for \( n = 1 \) to \( N \)

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

next

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

Only one training example to update the classifier

on-line

Given:

- ONE labeled training sample
  \[ \langle x, y \rangle \mid y \pm 1 \]
- strong classifier to update

for \( n = 1 \) to \( N \)

next

\[ h^{strong}(x) = \text{sign}( \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x) ) \]
From Offline to Online Boosting

**off-line**

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

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

\( h_0^{\text{strong}}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{\text{weak}}(x)) \)

**on-line**

**Given:**
- ONE labeled training sample \( \langle x, y \rangle \mid y \pm 1 \)
- strong classifier to update

for \( n = 1 \) to \( N \)
- update importance weight \( \lambda \)

\( h_0^{\text{strong}}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{\text{weak}}(x)) \)
From Offline to Online Boosting

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

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

next

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

Online update the weak classifier

Given:
- ONE labeled training sample \( \langle x, y \rangle \mid y \pm 1 \)
- strong classifier to update

- initial importance \( \lambda = 1 \)

for \( n = 1 \) to \( N \)
- update the weak classifier using samples and importance \( h_n^{weak}(x) = \mathcal{L}(h_n^{weak}, \langle x, y \rangle, \lambda) \)
- update importance weight \( \lambda \)

next

\( h^{strong}(x) = \text{sign}( \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x) ) \)
From Offline to Online Boosting

**off-line**

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

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

next

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

**Update errors and weights**

**on-line**

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

- initial importance $\lambda = 1$

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

next

$h_{\text{strong}}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_n^{\text{weak}}(x))$
From Offline to Online Boosting

**off-line**

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

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

next

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

**on-line**

Given:
- ONE labeled training sample \( \langle x, y \rangle \mid y \pm 1 \)
- strong classifier to update

- initial importance \( \lambda = 1 \)

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

next

\[ h^{strong}(x) = \text{sign} \left( \sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(x) \right) \]
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
Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- initial importance

\[
\begin{align*}
\text{for } n = 1 \text{ to } N \\
& \quad \text{update the weak classifier using sample and importance} \\
& \quad \text{update error estimation} \\
& \quad \text{update weight} \\
& \quad \text{update importance weight} \\
\text{next}
\end{align*}
\]
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
Given:
- ONE labeled training sample
- strong classifier to update

Algorithm:
- update the weak classifier using sample and importance
- update error estimation
- update weight
- update importance weight
  next

Converges to the off-line results...


\[ h^{\text{strong}}(x) = \text{sign} \left( \sum_{n=1}^{N} \alpha_n \cdot h_n^{\text{weak}}(x) \right) \]
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.
Introducing “Selector”

- Selects one feature from its local feature pool

\[ \mathcal{H}_{\text{weak}} = \{ h_1^{\text{weak}}, \ldots, h_M^{\text{weak}} \} \]
\[ \mathcal{F} = \{ f_1, \ldots, f_M \} \]

\[ h_{\text{sel}}(x) = h_m^{\text{weak}}(x) \]
\[ m = \arg \min_i e_i \]

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

Online Boosting for Feature Selection

- **one training sample**
  - Initial importance $\lambda = 1$
  - Update $\alpha_1$

- **estimate importance $\lambda$**
  - Update $\alpha_2$

- Repeat for each training sample

- **current strong classifier $h_{\text{Strong}}$**

- **update $\alpha_N$**

Slide credit: Helmut Grabner
Updating the $M \cdot N$ weak classifier is very time consuming!

Use a shared feature pool

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

$$\mathcal{H}_{weak} = \mathcal{H}_{weak}^1 = \ldots = \mathcal{H}_{N}^{weak}$$
Direct Feature Selection

one training sample

hSelector_1

estimate errors
select best weak classifier

update weight

α_1

estimate importance

λ

repeat for each training sample

h_i

h_k

h_m

... 

global weak classifier pool

hSelector_2

estimate errors
select best weak classifier

update weight

α_2

estimate importance

λ

h_Selector_N

estimate errors
select best weak classifier

update weight

α_N

current strong classifier hStrong

slide credit: Helmut Grabner
one training sample

\[ h_1, h_i, \ldots, h_k \]

estimate errors
select best weak classifier

\[ \alpha_1 \]

update weight

initial importance \( \lambda = 1 \)

\[ h_1, h_i, \ldots, h_M \]

global weak classifier pool

estimate errors
select best weak classifier

\[ \alpha_2 \]

estimate importance \( \lambda \)

update weight

repeat for each training sample

estimate errors
select best weak classifier

\[ \alpha_N \]

update weight

\[ h_{\text{Selector}_1}, h_{\text{Selector}_2}, \ldots, h_{\text{Selector}_N} \]

current strong classifier \( h_{\text{Strong}} \)

Slide credit: Helmut Grabner
Direct Feature Selection

one training sample

repeat for each training sample

estimate errors select best weak classifier

update weight \( \alpha_1 \)

hSelector_1

estimate importance \( \lambda \)

initial importance \( \lambda = 1 \)

update weight

\( h_1 \) \( h_i \) \( h_k \) \( h_m \) \( h_M \)

global weak classifier pool

estimate errors select best weak classifier

update weight \( \alpha_2 \)

\( hSelector_2 \)

estimate importance \( \lambda \)

estimate errors select best weak classifier

update weight \( \alpha_3 \)

\( hSelector_N \)

\( \alpha_N \)

\( h_i \)

\( h_m \)

\( h_M \)

current strong classifier \( h_{\text{Strong}} \)
Direct Feature Selection

one training sample

\( h_1 \rightarrow \ldots \rightarrow h_i \rightarrow \ldots \rightarrow h_k \rightarrow \ldots \rightarrow h_M \)

global weak classifier pool

\( \text{hSelector}_1 \)

\( \text{estimate errors} \)

\( \text{select best weak classifier} \)

\( \lambda = 1 \)

\( \alpha_1 \)

\( \text{update weight} \)

\( \text{estimate importance} \)

\( \alpha_2 \)

\( \text{update weight} \)

\( \text{estimate errors} \)

\( \text{select best weak classifier} \)

\( \lambda \)

\( \alpha \)

\( \text{update weight} \)

\( \text{estimate errors} \)

\( \text{select best weak classifier} \)

\( \alpha_N \)

\( \text{update weight} \)

\( \text{current strong classifier } h_{\text{Strong}} \)
Tracking by Online Classification

Update classifier (tracker)

from time $t$ to $t+1$

Analyze map and set new object position

Evaluate classifier on sub-patches

Create confidence map

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide credit: Helmut Grabner
Image source: Disney/Pixar
Tracking Results

Lecture: Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

Slide credit: Helmut Grabner
Video source: Grabner et al., BMVC’06
Online Feature Exchange

Slide credit: Helmut Grabner

Video source: Grabner et al., BMVC’06
Additional Tracking Results

Slide credit: Helmut Grabner

Video source: Grabner et al., BMVC'06
“Tracking the Invisible”

Slide credit: Helmut Grabner

Video source: Grabner et al., BMVC’06
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...
Why Does It Fail?

**Lecture:** Computer Vision 2 (SS 2016) – Template-based Tracking
Prof. Dr. Bastian Leibe, Dr. Jörg Stückler

1. **Update classifier (tracker)**
2. **Search region**
3. **Create confidence map**
4. **Analyze map and set new object position**
5. **Evaluate classifier on sub-patches**

**Slide credit:** Helmut Grabner
**Image source:** Disney/Pixar
Why Does It Fail?

- Actual object position
- Update classifier (tracker)
- Analyze map and set new object position
- Create confidence map
- Evaluate classifier on sub-patches
- Search region

Self-learning
Drifting Due to Self-Learning Policy

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

Strategy 2: Semi-Supervised Learning

Object Detector \hspace{1cm} Our approach \hspace{1cm} Object Tracker

Fixed Training set \hspace{1cm} Fixed Prior for updating an Adaptive on-line classifier \hspace{1cm} On-line update Object vs. Background

Prior

Labeled data \rightarrow Un-labeled data


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.

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

- Online Boosting for Tracking

- Semi-Supervised Boosting

- Tracking-Learning-Detection