Visual Tracking course @ UCU
Day 3: State-of-art methods
Dmytro Mishkin

Center for Machine Perception
Czech Technical University in Prague

Lviv, Ukraine

2017

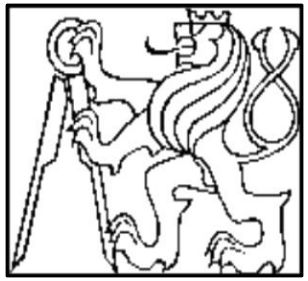
Heavily based on tutorial:

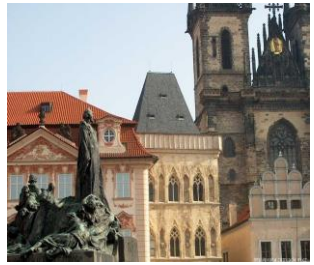
Visual Tracking

by Jiri Matas

„… Although tracking itself is by and large solved problem...“,

-- Jianbo Shi & Carlo Tomasi
CVPR1994 --





Course plan

Day 1: Basics

- Visual tracking: not one, but many problems.
- The KLT tracker and Optical Flow
- HW: KLT and simple Tracker by Detection

Day 2: CNN (mostly not about tracking)

- General CNN finetuning
- CNN design choices
- CNN-based trackers
- HW: Finetuning VGGNet for MDNet tracker

Day 3: State-of-art

- Discriminative Correlation Filters
- Long-term trackers
- How to evaluate tracker
- HW: KCF implementation

Outline of the Day 3

1. Correlation filters trackers family.
2. Online discriminative tracking.
3. Long-term trackers.
4. How to evaluate trackers.

Tracking with Correlation Filters

Acknowledgement to João F. Henriques from
Institute of Systems and Robotics
University of Coimbra
for providing materials for this presentation

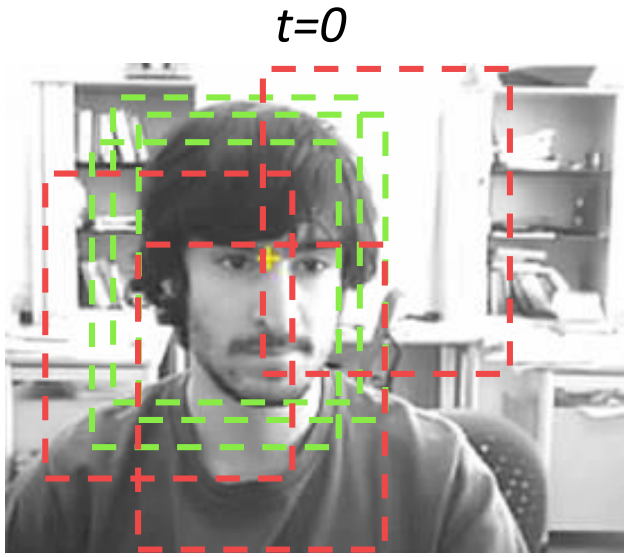
Overview

- Discriminative tracking
- Connection of correlation and the discriminative tracking

- Brief history of correlation filters
- Breakthrough by MOSSE tracker

- Kernelized Correlation Filters
- Discriminative Correlation Filters

Discriminative Tracking (T. by Detection)



samples

+1 +1 +1 −1 −1 −1

labels

Classifier

Classify subwindows to find target

t>0

Discriminative Tracking

- How to get training samples for the classifier?
- **Standard approach**:
  - bboxes with high overlap with the GT → Pos. samples
  - bboxes far from the GT → Neg. samples

t=0

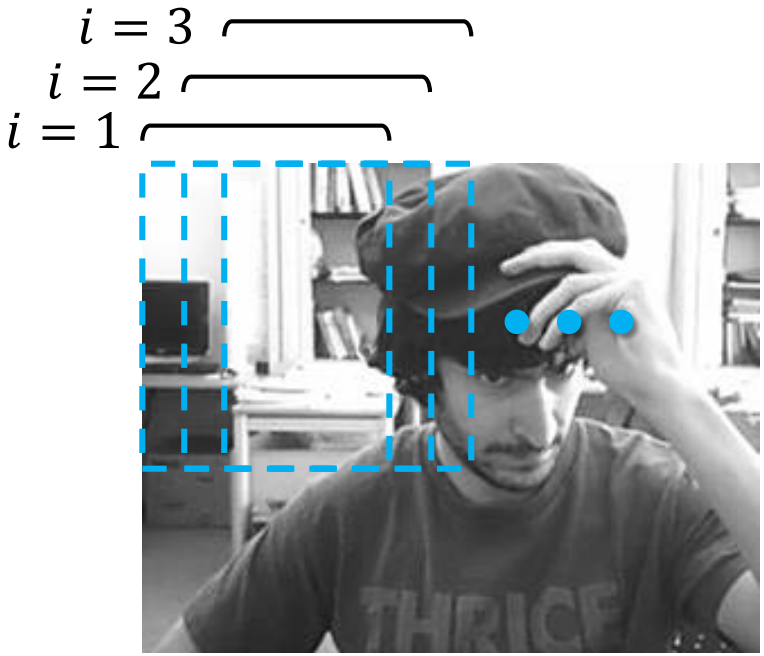
- Neg. samples
- Pos. samples
- Unspecified

- What with the samples in the unspecified area?

Connection to Correlation

- Let's have a linear classifier with weights $\mathbf{w}$

$$y = \mathbf{w}^T \mathbf{x}$$



- During tracking we want to evaluate the classifier at subwindows $\mathbf{x}_i$:

$$y_i = \mathbf{w}^T \mathbf{x}_i$$

- Then we can concatenate y_i into a vector $\mathbf{y}$ (i.e. response map)

- This is equivalent to **cross-correlation** formulation which can be computed efficiently in Fourier domain

$$\mathbf{y} = \mathbf{x} \circledast \mathbf{w}$$

- Note: Convolution is related; it is the same as cross-correlation, but with the flipped image of $\mathbf{w}$ ($\rightarrow$).

Connection to Correlation

The Convolution Theorem

"Cross-correlation is **equivalent** to an **element-wise product** in Fourier domain"

$$\mathbf{y} = \mathbf{x} \circledast \mathbf{w} \quad \Longleftrightarrow \quad \hat{\mathbf{y}} = \hat{\mathbf{x}}^* \times \hat{\mathbf{w}}$$

- where:
  - $\hat{\mathbf{y}} = \mathcal{F}(\mathbf{y})$ is the Discrete Fourier Transform (DFT) of $\mathbf{y}$. (likewise for $\hat{\mathbf{x}}$ and $\hat{\mathbf{w}}$)
  - $\times$ is element-wise product
  - $.^*$ is complex-conjugate (i.e. negate imaginary part).

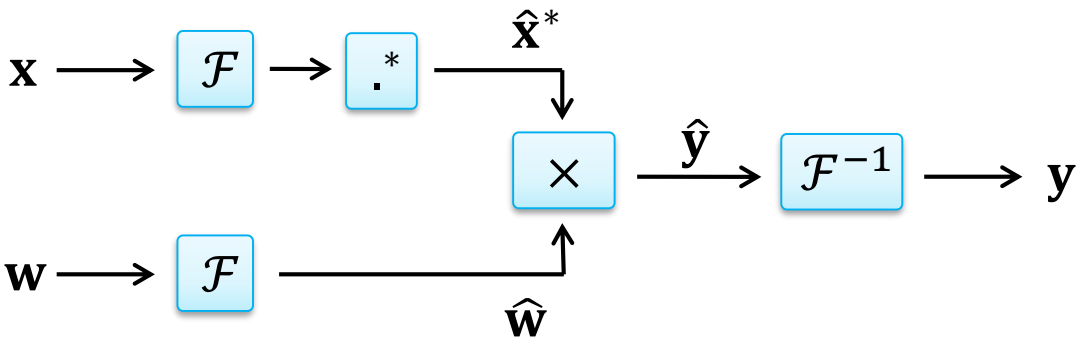
- Note that cross-correlation, and the DFT, are **cyclic** (the window wraps at the image edges).

The Convolution Theorem

"Cross-correlation is **equivalent** to an **element-wise product** in Fourier domain"

$$\mathbf{y} = \mathbf{x} \circledast \mathbf{w} \quad \Longleftrightarrow \quad \hat{\mathbf{y}} = \hat{\mathbf{x}}^* \times \hat{\mathbf{w}}$$

- In practice:



- Can be orders of magnitude faster:
  - For $n \times n$ images, cross-correlation is $\mathcal{O}(n^4)$.
  - Fast Fourier Transform (and its inverse) are $\mathcal{O}(n^2 \log n)$.

Connection to Correlation

The Convolution Theorem

"Cross-correlation is **equivalent** to an **element-wise product** in Fourier domain"

$$\mathbf{y} = \mathbf{x} \circledast \mathbf{w} \quad \Longleftrightarrow \quad \hat{\mathbf{y}} = \hat{\mathbf{x}}^* \times \hat{\mathbf{w}}$$

- Conclusion:

The evaluation of any linear classifier can be accelerated with the Convolution Theorem.

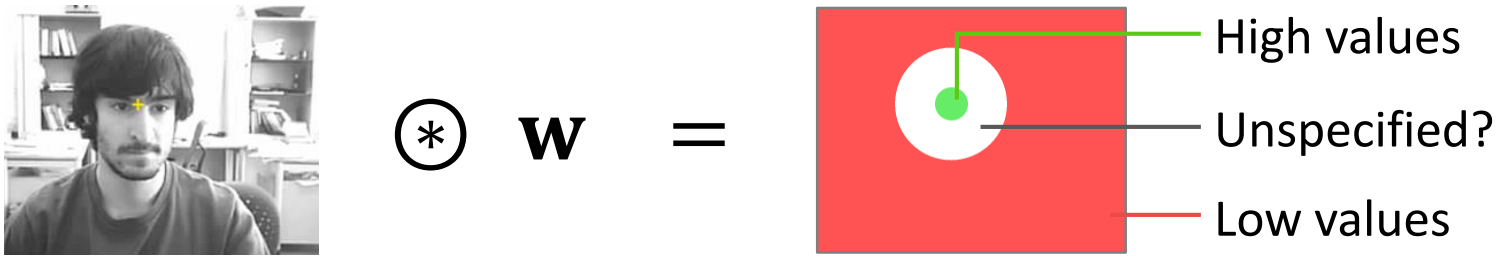
- "linear" can become non-linear using kernel trick in some specific cases(will be discussed later)

- Q: How the $\mathbf{w}$ for correlation should look like? What about **training**?

Connection to Correlation

- Q: How the $\mathbf{w}$ for correlation should look like? What about **training**?

Objective



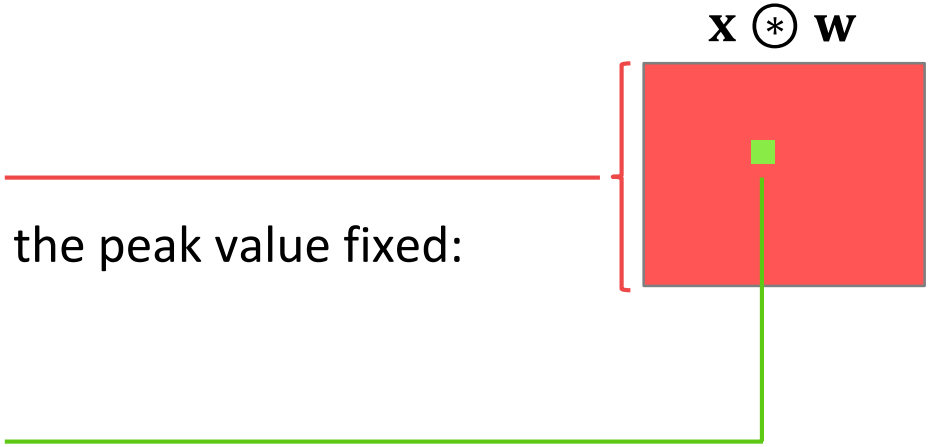
- Intuition of requirements of cross-correlation of classifier(filter) $\mathbf{w}$ and a training image $\mathbf{x}$
  - A **high peak** near the true location of the target
  - **Low values** elsewhere (to minimize false positive)

Brief History of Correlation Filters

Minimum Average Correlation Energy (MACE) filters, 1980's

- Bring average correlation output towards 0:

$$\min_{\mathbf{w}} \ \|\mathbf{x} \circledast \mathbf{w}\|^2$$



except for target location, keep the peak value fixed:

$$\text{subject to: } \mathbf{w}^T\mathbf{x} = 1$$

- This produces a **sharp peak** at target location with closed form solution:

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}}{\hat{\mathbf{x}}^* \times \hat{\mathbf{x}}}$$

- $\hat{\mathbf{x}}^* \times \hat{\mathbf{x}}$ is called the **spectrum** and is real-valued.
- division and product ($\times$) are element-wise.

- **Sharp peak = good localization**! Are we done?

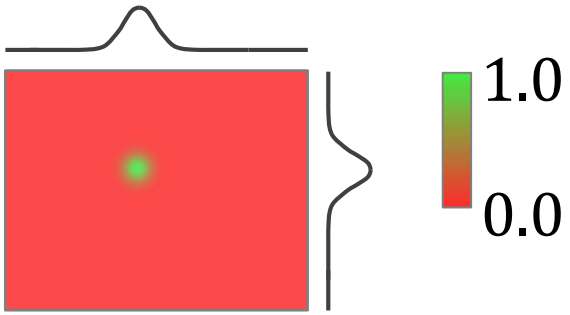
Brief History of Correlation Filters

The MACE filter suffers from 2 main issues:

1. Hard constraints easily lead to overfitting.
   - **UMACE** ("Unconstrained MACE") addresses this by removing the hard constraints and require to produce a high average correlation response on positive samples. However, it still suffer from the 2nd problem.
2. Enforcing a sharp peak is too strong condition; lead to overfitting
   - **Gaussian-MACE / MSE-MACE** – peak to follow a 2D Gaussian shape

$$\min_{\mathbf{w}} \|\mathbf{x} \circledast \mathbf{w} - \mathbf{g}\|^2, \qquad \mathbf{g} =$$

$$\text{subject to:} \quad \mathbf{w}^T\mathbf{x} = 1$$

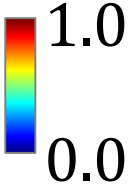


   - In the original method (1990's), the minimization was *still* subject to the MACE hard constraint.
     (*It later turned out to be unnecessary*!)

Brief History of Correlation Filters

Sharp vs. Gaussian peaks

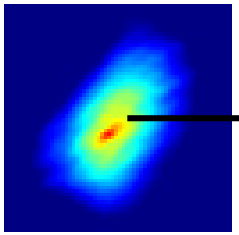
Training image: $\mathbf{x} =$ 



Naïve filter
($\mathbf{w} = \mathbf{x}$)

Classifier
($\mathbf{w}$)

Output
($\mathbf{w} * \mathbf{x}$)

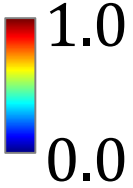


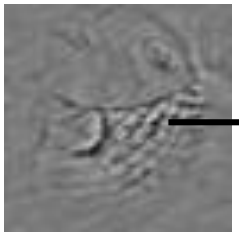
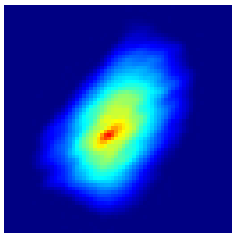
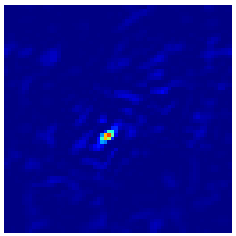
- Very broad peak is hard to localize (especially with clutter).
- State-of-the-art classifiers (e.g. SVM) show **same** behavior!

Brief History of Correlation Filters

Sharp vs. Gaussian peaks

Training image: $\mathbf{x} =$ 



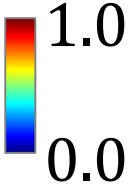
	Naïve filter ($\mathbf{w} = \mathbf{x}$)	Sharp peak (UMACE)
Classifier ($\mathbf{w}$)		
Output ($\mathbf{w} * \mathbf{x}$)		

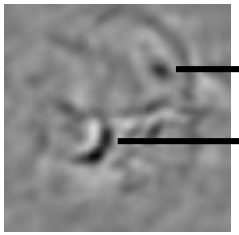
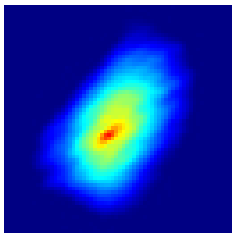
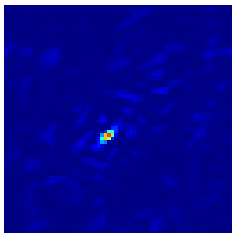
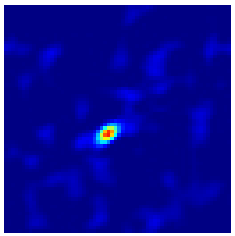
- A very sharp peak is obtained by emphasizing **small image details** (like the fish's scales here).
- **generalizes poorly**; fine scale details that are usually not robust

Brief History of Correlation Filters

Sharp vs. Gaussian peaks

Training image: $\mathbf{x} =$ 



	Naïve filter ($\mathbf{w} = \mathbf{x}$)	Sharp peak (UMACE)	Gaussian peak (GMACE)
Classifier ($\mathbf{w}$)			
Output ($\mathbf{w} * \mathbf{x}$)			

- A good compromise.
- Tiny details are ignored.
- focuses on **larger, more robust structures**.

Breakthrough by MOSSE tracker

Min. Output Sum of Sq. Errors (MOSSE)

- Presented by David Bolme and colleagues at CVPR 2010

- Tracker run at speed over a **600 frames per second**

- **very simple** to implement
  - no complex features only raw pixel values
  - only FFT and element-wise operation

- performance similar to the most sophisticated tracker (at that time)

Breakthrough by MOSSE tracker

How does it work?

- Use only the "Gaussian peak" objective (no hard constraints)

$$\min_{\mathbf{w}} \|\mathbf{x} \circledast \mathbf{w} - \mathbf{g}\|^2, \qquad \mathbf{g} =$$

- Found the following solution using the Convolution Theorem:

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{g}}^* \times \hat{\mathbf{x}}}{\hat{\mathbf{x}}^* \times \hat{\mathbf{x}} + \lambda}$$

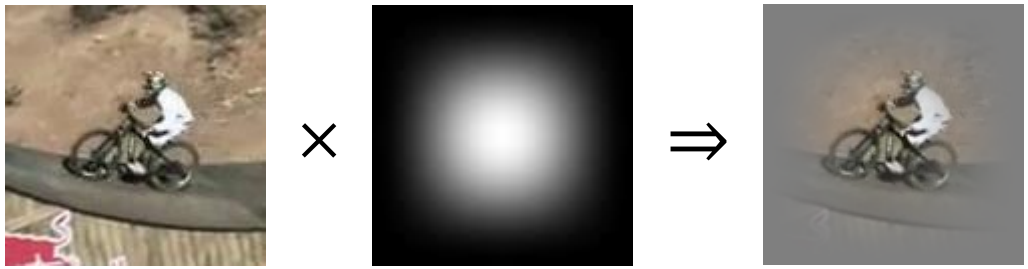
($\lambda = 10^{-4}$ is artificially added to prevent divisions by 0)

- **No expensive matrix operations!** $\Rightarrow$ only FFT and element-wise op.

Breakthrough by MOSSE tracker

Implementation aspects

- Cosine (or sine) window preprocessing



  - image edges smooth to zero
    $\Rightarrow$ the filter sees an image as a “cyclic” (important for the FFT)
  - gives more importance to the target center.

- Simple update

$$\hat{\mathbf{w}}_{\text{new}} = \frac{\hat{\mathbf{g}}^* \times \hat{\mathbf{x}}}{\hat{\mathbf{x}}^* \times \hat{\mathbf{x}} + \lambda}$$

Train a MOSSE filter $\hat{\mathbf{w}}_{\text{new}}$ using the new image $\hat{\mathbf{x}}$.

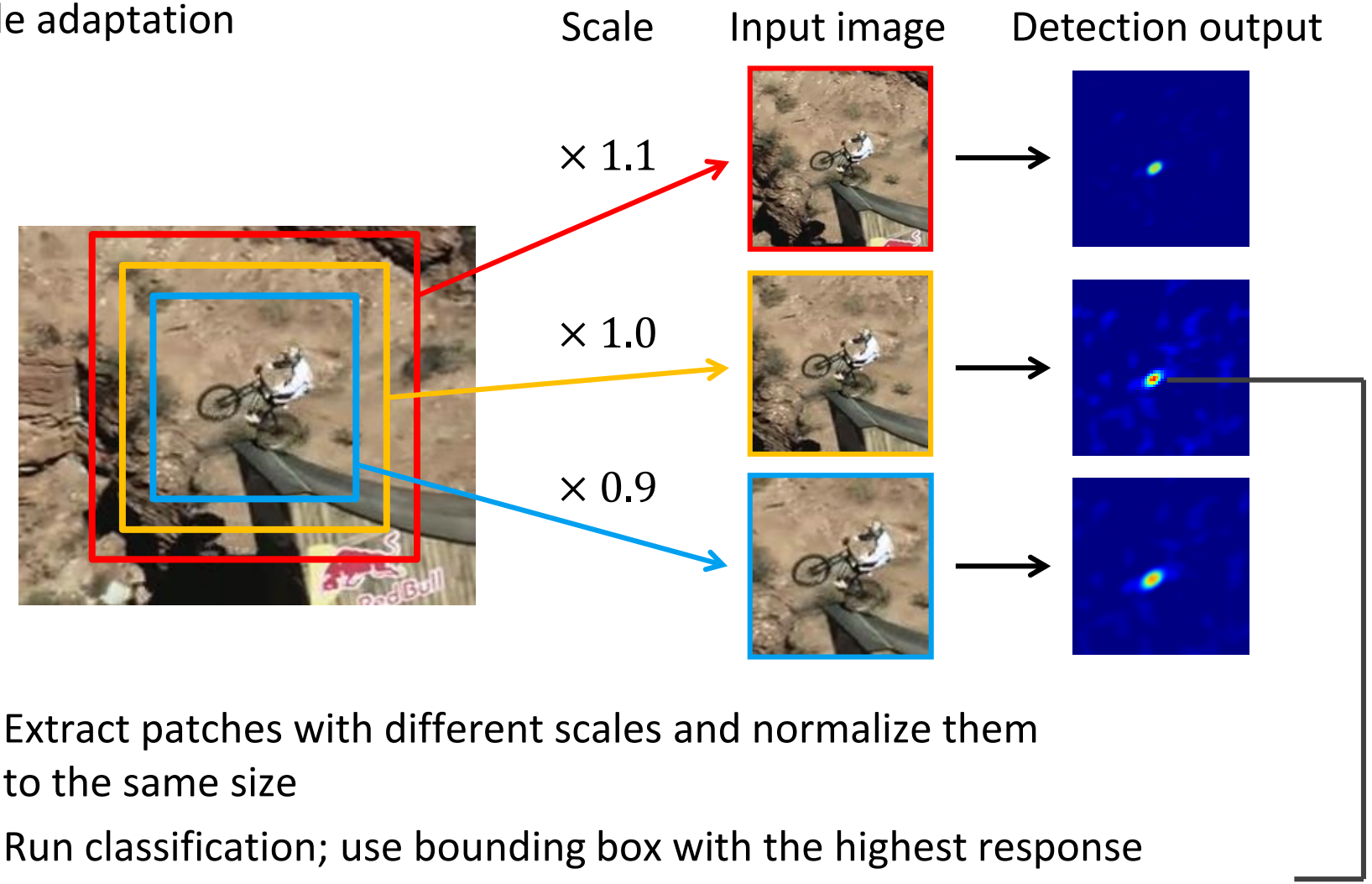
$$\hat{\mathbf{w}}_t = (1-\eta)\hat{\mathbf{w}}_{t-1} + \eta\hat{\mathbf{w}}_{\text{new}}$$

Update previous solution $\hat{\mathbf{w}}_{t-1}$ with $\hat{\mathbf{w}}_{\text{new}}$ by linear interpolation.

Breakthrough by MOSSE tracker

Implementation aspects

- Scale adaptation



- Extract patches with different scales and normalize them to the same size
- Run classification; use bounding box with the highest response

Why MOSSE works?

Ridge Regression Formulation

= Least-Squares with regularization (avoids overfitting!)

- Consider simple Ridge Regression (RR) problem:

$$\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2$$

has closed-form solution: $\mathbf{w} = \left(X^T X + \lambda I\right)^{-1} X^T \mathbf{y}$

We can replace $\mathrm{X} = C(\mathbf{x})$ (circulant data), and $\mathbf{y} = \mathbf{g}$ (Gaussian targets).

- **Diagonalizing** the involved circulant matrices with the DFT yields:

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}^* \times \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \times \hat{\mathbf{x}} + \lambda} \quad \Rightarrow$$

- Exactly the MOSSE solution!
- **good learning algorithm** (RR) with **lots of data** (circulant/shifted samples).

Kernelized Correlation Filters

- Circulant matrices are a **very general tool** which allows to replace standard operations with fast Fourier operations.
- The same idea can by applied e.g. to the **Kernel Ridge Regression**:

  with K kernel matrix $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and dual space representation

$$\boldsymbol{\alpha} = (K + \lambda I)^{-1}\mathbf{y}$$

- For many kernels, circulant data $\Rightarrow$ circulant K matrix

$$K = C(\mathbf{k}^{\mathbf{xx}}),$$

  where $\mathbf{k}^{\mathbf{xx}}$ is kernel auto-correlaton and the first row of K (small, and easy to compute)

- Diagonalizing with the DFT for learning the classifier yields:

$$\hat{\boldsymbol{\alpha}} = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda} \quad \Rightarrow$$

Fast solution in $\mathcal{O}(n \log n)$.
Typical kernel algorithms are $\mathcal{O}(n^2)$ or higher!

Kernelized Correlation Filters

- The $\mathbf{k}^{\mathbf{xx'}}$ is kernel correlation of two vectors **x** and **x'**

$$k_i^{\mathbf{xx'}} = \kappa(\mathbf{x}', P^{i-1}\mathbf{x})$$

- For Gaussian kernel it yields:

multiple channels can be concatenated to the vector **x** and then sum over in this term

$$\mathbf{k}^{\mathbf{xx'}} = \exp\left(-\frac{1}{\sigma^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\mathcal{F}^{-1}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}')\right)\right)$$

- Evaluation on subwindows of image **z** with classifier $\boldsymbol{\alpha}$ and model **x**:

1. $K^{\mathbf{z}} = C(\mathbf{k}^{\mathbf{xz}})$
2. $\mathbf{f}(\mathbf{z}) = \mathcal{F}^{-1}\left(\hat{\mathbf{k}}^{\mathbf{xz}} \odot \hat{\boldsymbol{\alpha}}\right)$

- Update classifier $\boldsymbol{\alpha}$ and model **x** by linear interpolation from the location of maximum response **f**(**z**)

- Kernel allows integration of more complex and multi-channel features

Kernelized Correlation Filters

KCF Tracker

- very few hyperparameters
- code fits on one slide of the presentation!
- Use HoG features (32 channels)
- ~300 FPS
- Open-Source (Matlab/Python/Java/C)

Training and detection (Matlab)

```
function alphaf = train(x, y, sigma, lambda)
  k = kernel_correlation(x, x, sigma);
  alphaf = fft2(y) ./ (fft2(k) + lambda);
end

function y = detect(alphaf, x, z, sigma)
  k = kernel_correlation(z, x, sigma);
  y = real(ifft2(alphaf .* fft2(k)));
end

function k = kernel_correlation(x1, x2, sigma)
  c = ifft2(sum(conj(fft2(x1)) .* fft2(x2), 3));
  d = x1(:)'*x1(:) + x2(:)'*x2(:) - 2 * c;
  k = exp(-1 / sigma^2 * abs(d) / numel(d));
end
```

Sum over channel dimension in kernel computation

From KCF to Discriminative CF trackers

Basic

- Henriques et al. – CSK
  - raw grayscale pixel values as features
- Henriques et al. – KCF
  - HoG multi-channel features

Further work

- Danelljan et al. – DSST:
  - PCA-HoG + grayscale pixels features
  - filters for translation and for scale (in the scale-space pyramid)
- Li et al. – SAMF:
  - HoG, color-naming and grayscale pixels features
  - quantize scale space and normalize each scale to one size by bilinear inter.
    → only one filter on normalized size

Discriminative Correlation Filters Trackers

- Danelljan et al. –SRDCF:
  - spatial regularization in the learning process
    → limits boundary effect
    → penalize filter coefficients depending on their spatial location
  - allows to use much larger search region
  - more discriminative to background (more training data)

CNN-based Correlation Trackers

- Danelljan et al. – Deep SRDCF, CCOT (best performance in VOT 2016)
- Ma et al.
  - features : VGG-Net pretrained on ImageNet dataset extracted from third, fourth and fifth convolution layer
  - for each feature learn a linear correlation filter

CNN-based Trackers (not correlation based)

- Nam et al. – MDNet, T-CNN:
  - CNN classification (3 convolution layers and 2 fully connected layers) learn on tracking sequences with bbox regression

Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking

Martin Danelljan, Andreas Robinson,
Fahad Shahbaz Khan, Michael Felsberg

LINKÖPING UNIVERSITY

Discriminative Correlation Filters (DCF)

Applications

- Object recognition
- Object detection
- Object tracking
  - Among state-of-the-art since 2014
  - KCF, DSST, HCF, SRDCF, Staple …

Discriminative Correlation Filters (DCF)

Limitations:

Single-resolution feature map

Coarse output scores

DCF Limitations:

1. Single-resolution feature map

- Why a problem?
  - Combine convolutional layers of a CNN
    - Shallow layers: low invariance – high resolution
    - Deep layers: high invariance – low resolution
- How to solve?
  - Explicit resampling?
    - Artefacts, information loss, redundant data
  - Independent DCFs with late fusion?
    - Sub-optimal, correlations between layers

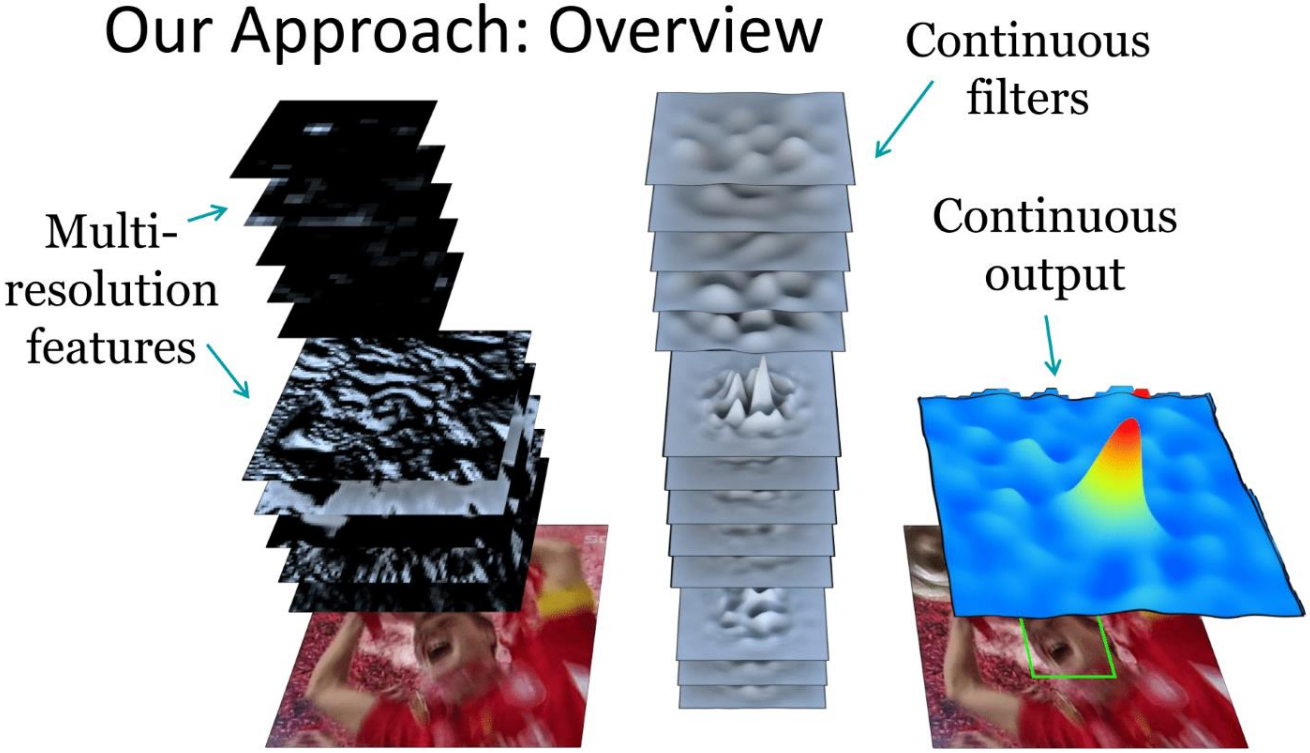
DCF Limitations:
2. Coarse output scores

- Why a problem?
  - Accurate localization
    - Sub-grid (e.g. HOG grid) or sub-pixel accuracy
    - More accurate annotations=> less drift
- How to solve?
  - Interpolation?
    - Which interpolation strategy?
  - Interweaving?
    - Costly

DCF Limitations:
3. Coarse labels

- Why a problem?
  - Accurate learning
    - Sub-grid or sub-pixel supervision
- How to solve?
  - Interweaving?
    - Costly
  - Explicit interpolation of features?
    - Artefacts

Our Approach: Overview



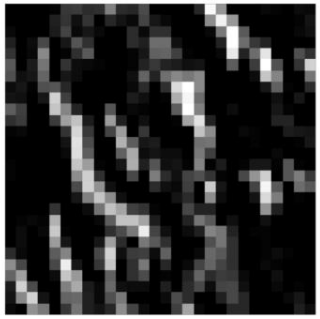
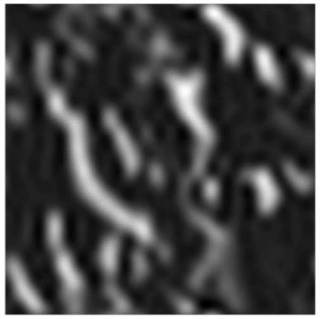
Multiresolution Features

$$x = \left\{ \right.$$

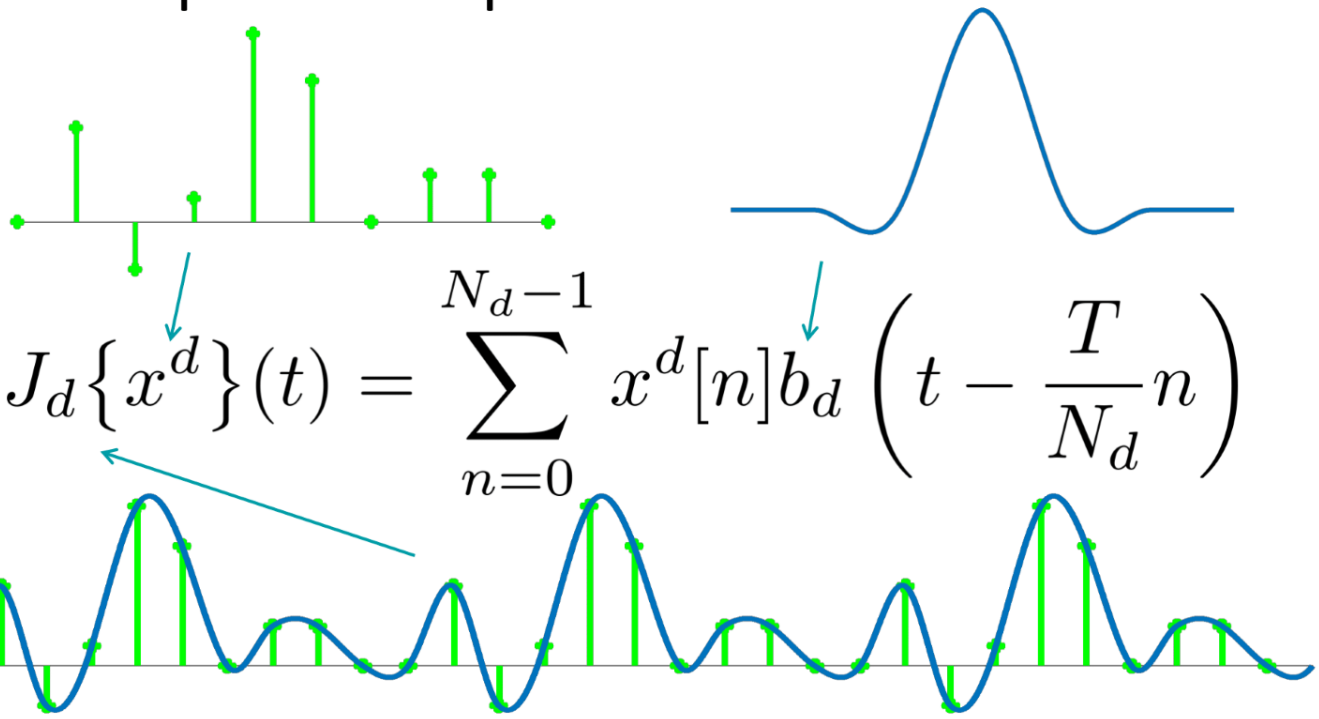
$$x^d \in \mathbb{R}^{N_d}$$

Interpolation Operator

$$J_d : \mathbb{R}^{N_d} \rightarrow L^2(T)$$

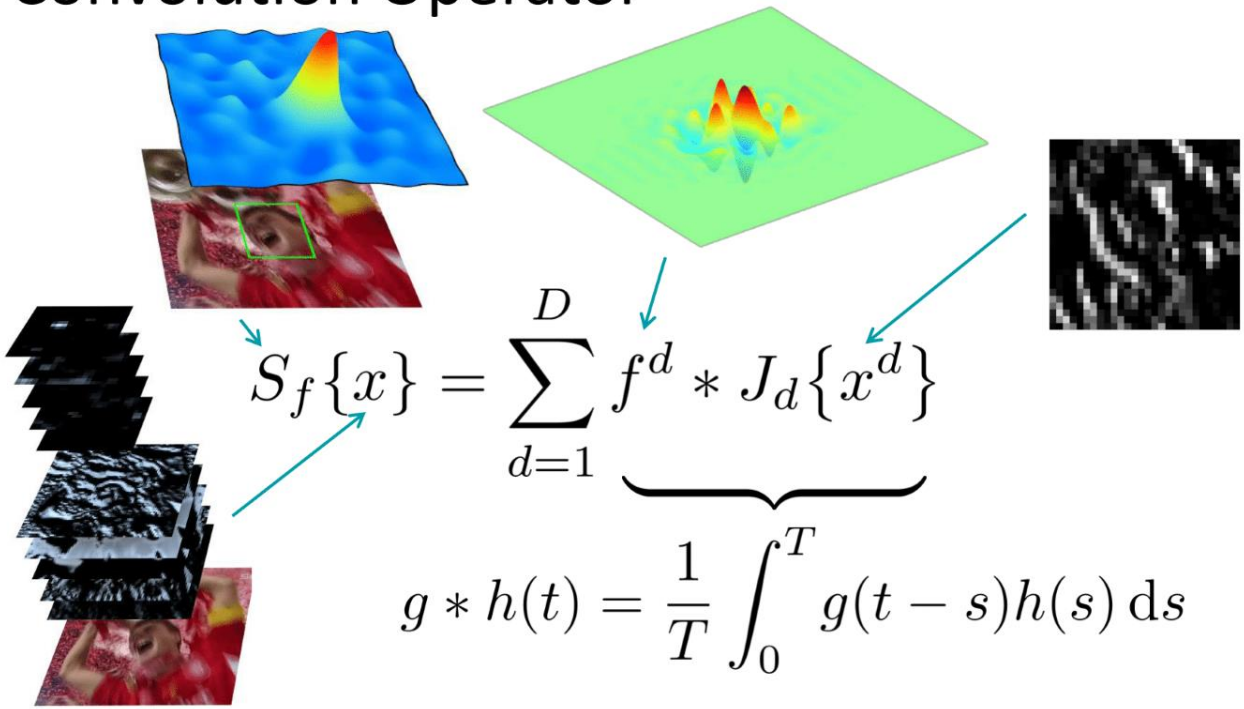
$J_d\{$  $\} =$ 

Interpolation Operator



$$J_d\{x^d\}(t) = \sum_{n=0}^{N_d-1} x^d[n] b_d \left(t - \frac{T}{N_d} n \right)$$

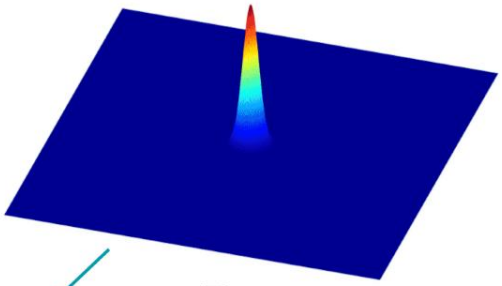
Convolution Operator



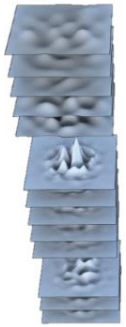
$$S_f\{x\} = \sum_{d=1}^{D} \underbrace{f^d * J_d\{x^d\}}$$

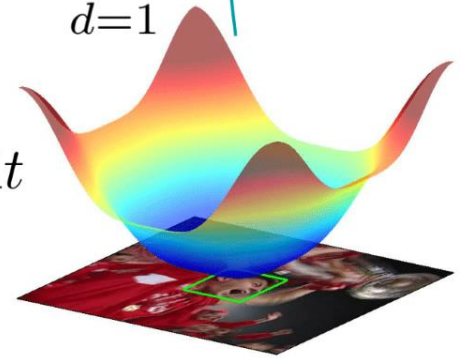
$$g * h(t) = \frac{1}{T} \int_0^T g(t-s)h(s)\,\mathrm{d}s$$

Training Loss



$$E(f) = \sum_{j=1}^{m} \alpha_j \underbrace{\left\| S_f\{x_j\} - y_j \right\|^2}_{\|g\|^2 = \frac{1}{T}\int_0^T |g(t)|^2 \, \mathrm{d}t} + \sum_{d=1}^{D} \left\| w f^d \right\|^2$$





[Danelljan et al., ICCV 2015]

Training Loss – Fourier Domain

$$E(f) = \sum_{j=1}^{m} \alpha_j \underbrace{\left\| \sum_{d=1}^{D} \hat{f}^d X_j^d \hat{b}_d - \hat{y}_j \right\|_{\ell^2}^2}_{\|\hat{g}\|_{\ell^2}^2 = \sum_{-\infty}^{\infty} |\hat{g}[k]|^2} + \sum_{d=1}^{D} \left\| \hat{w} * \hat{f}^d \right\|_{\ell^2}^2$$

$$\hat{g}[k] = \langle g, e_k \rangle = \frac{1}{T} \int_0^T g(t) e^{-i \frac{2\pi}{T} kt} \, \mathrm{d}t$$

$$X^d[k] = \sum_{n=0}^{N_d - 1} x^d[n] e^{-i \frac{2\pi}{N_d} nk}$$

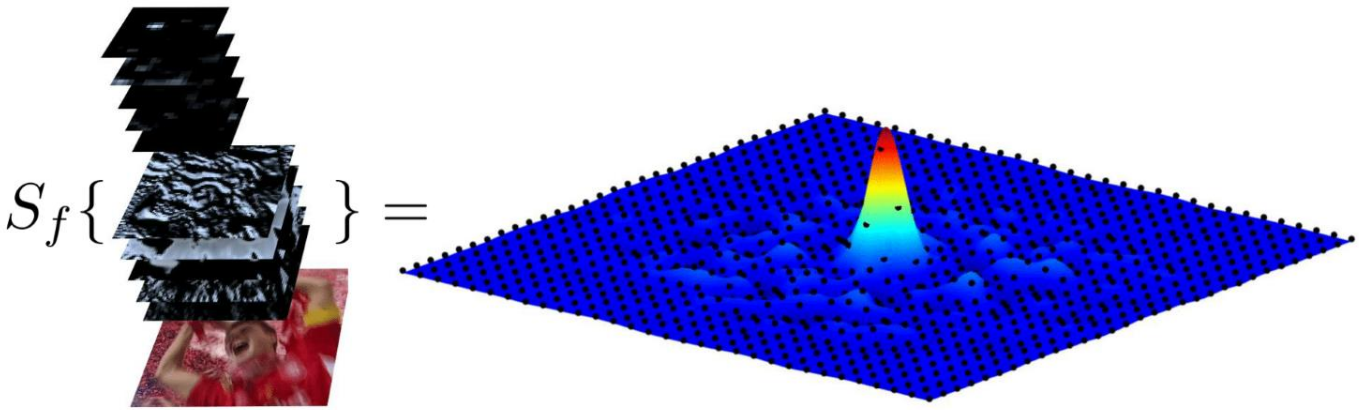
Training Loss – Fourier Domain

$$E(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{f}^d X_j^d \hat{b}_d - \hat{y}_j \right\|_{\ell^2}^2 + \sum_{d=1}^{D} \left\| \hat{w} * \hat{f}^d \right\|_{\ell^2}^2$$

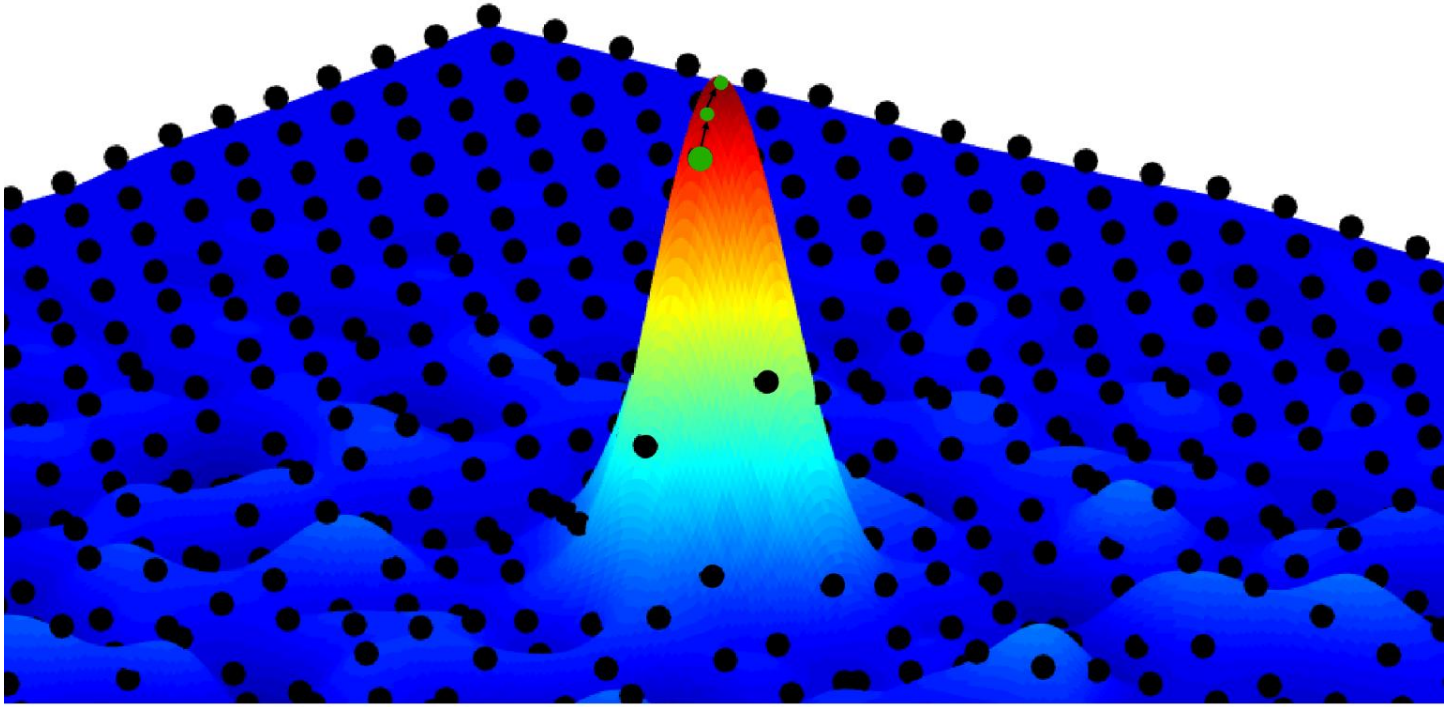
$$\Downarrow$$

$$\left(A^{\mathrm{H}} \Gamma A + W^{\mathrm{H}} W \right) \hat{\mathbf{f}} = A^{\mathrm{H}} \Gamma \hat{\mathbf{y}}$$

Localization



Localization



How to set y_j and b_d ?

- Use periodic summation of functions $g : \mathbb{R} \to \mathbb{R}$:

$$g_T(t) = \sum_{n=-\infty}^{\infty} g(t - nT)$$

- Gaussian function for y_j
- Cubic spline kernel for b_d
- Fourier coefficients $\hat{y}_j, \hat{b}_d$ with Poisson's summation formula:

$$\hat{g}_T[k] = \tfrac{1}{T}\hat{g}(\tfrac{k}{T})$$

Object Tracking Framework: Features

- VGG network
  - Pre-trained on ImageNet
  - No fine-tuning on application specific data

Object Tracking Framework: Optimization

Solving $\left(A^{\mathrm{H}}\Gamma A + W^{\mathrm{H}}W\right)\hat{\mathbf{f}} = A^{\mathrm{H}}\Gamma\hat{\mathbf{y}}$

SRDCF: **Gauss-Seidel**	C-COT: **Conjugate Gradient**
☹ Explicit computation of $\left(A^{\mathrm{H}}\Gamma A + W^{\mathrm{H}}W\right)$	☺ Only need to evaluate $\left(A^{\mathrm{H}}\Gamma A + W^{\mathrm{H}}W\right)\hat{\mathbf{f}}$
😐 Sparse matrix handling	☺ **No** sparse matrix handling
☹ $\mathcal{O}(D^2)$	☺ $\mathcal{O}(D)$
😐 "Infinite" memory	😐 Finite memory
☺ Warm starting: trivial	😐 Warm starting: **non**-trivial
	😐 Tuning of pre-conditioners

Object Tracking Framework: Pipeline

- Simple:

  … – Track – Train – Track – Train – …

- No thresholds
- No hidden “tricks”

Experiments: Object Tracking

- 3 datasets: OTB-100, TempleColor, VOT2015
- Layer fusion on OTB:

	Layer 0	Layer 1	Layer 5	Layers 0, 1	Layers 0, 5	Layers 1, 5	Layers 0, 1, 5
Mean OP	58.8	78.0	60.0	77.8	70.7	*81.8*	**82.4**
AUC	49.9	65.8	51.1	65.7	59.0	*67.8*	**68.2**

- Compared to explicit resampling in DCF
  - Performance gain: $+7.4\%$ AUC
  - Efficiency gain: -80% data size

Experiments: VOT2016

	Tracker	EAO	A	R	A_{rank}	R_{rank}	AO	EFO	Impl.
1.	C-COT	**0.331**	0.539	*0.238*	12.000	**1.000**	0.469	0.507	D M
2.	TCNN	*0.325*	0.554	0.268	4.000	*2.000*	*0.485*	1.049	S M
3.	SSAT	0.321	**0.577**	0.291	**1.000**	3.000	**0.515**	0.475	S M
4.	MLDF	0.311	0.490	**0.233**	36.000	**1.000**	0.428	1.483	D M
5.	Staple	0.295	0.544	0.378	5.000	10.000	0.388	11.144	D C

[Matej et al., ECCV VOT workshop 2016]

Object Tracking: Speed

- With CNN features: slow ~1 FPS (no GPU)
- With HOG features: ~ real time at SRDCF performance

Feature Point Tracking Framework

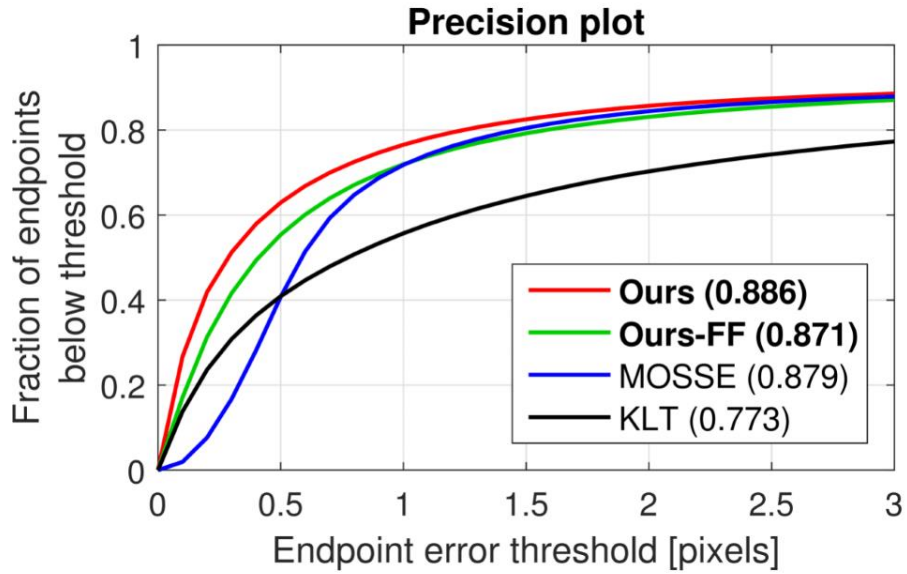
- Grayscale pixel features, $D = 1$
- Uniform regularization, $w(t) = \beta$

$$\Downarrow$$

$$\hat{f}[k] = \frac{\sum_{j=1}^{m} \alpha_j \overline{X_j[k]\hat{b}[k]}\hat{y}_j[k]}{\sum_{j=1}^{m} \alpha_j \left| X_j[k]\hat{b}[k] \right|^2 + \beta^2}$$

Experiments: Feature Point Tracking

- The Sintel dataset



Conclusions

- **Continuous domain** learning formulation
  - **Multi-resolution** deep feature maps
  - **Sub-pixel** accurate localization
  - **Sub-pixel** supervision
- Superior results for two applications
  - Object tracking
  - Feature point tracking

Discriminative Correlation Filter with Channel and Spatial Reliability

https://arxiv.org/abs/1611.08461

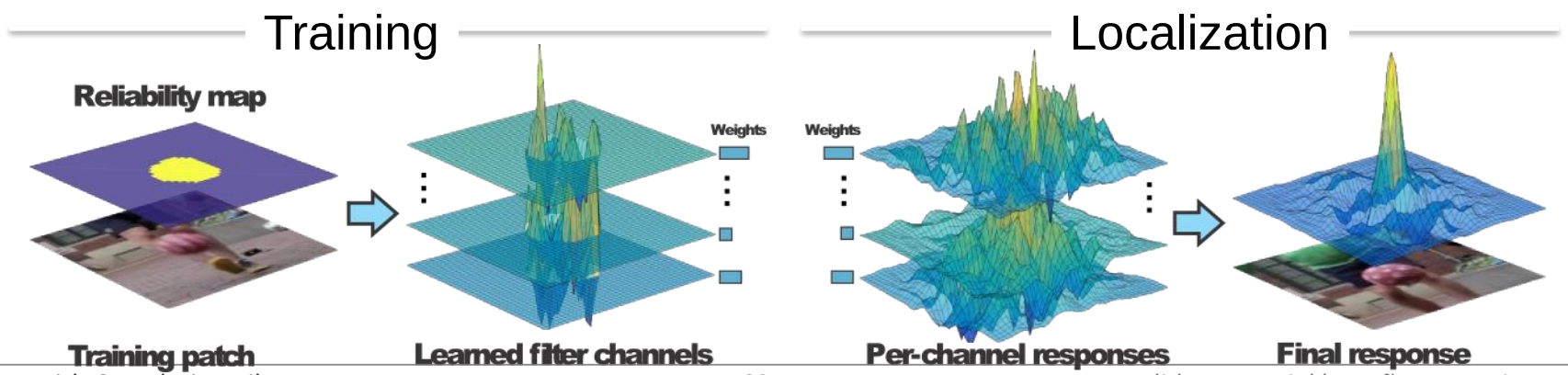
- State-of-the-art results, outperforms even trackers based on deep NN

- Simple features:
  - HoG features (18 contrast sensitive orientation channels)
  - binarized grayscale channel (1 channel)
  - color names (~mapping of RGB to 10 channels)

- Single-CPU single-thread, matlab implementation @13 fps

CSR-DCF

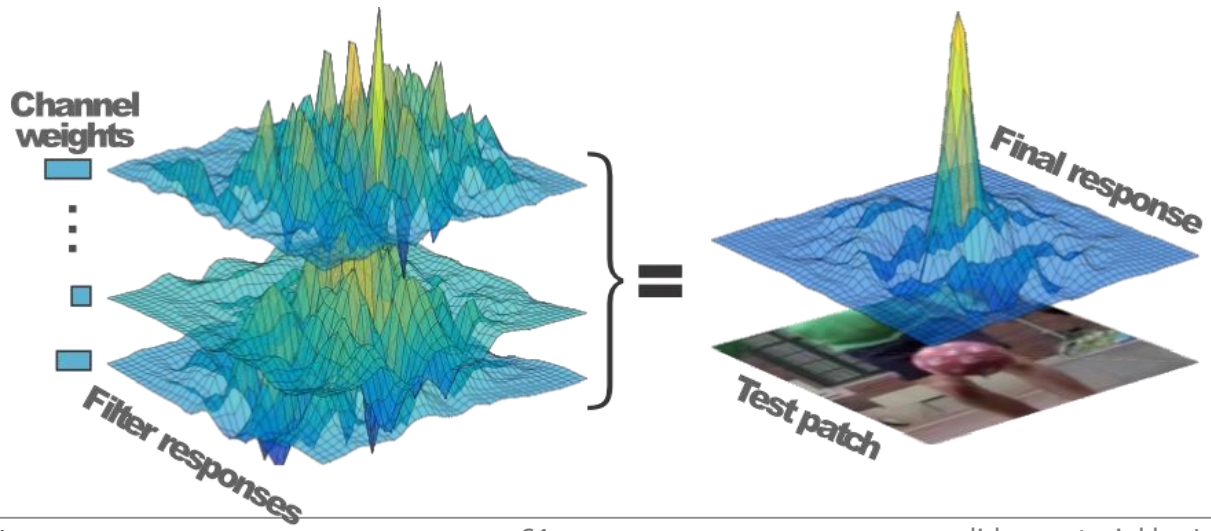
- Algorithm (repeats 1,2)
- Training:
  - Estimate object segmentation → object mask
  - **Learn correlation filter using the object mask as constraints**
  - Update generative weights for the feature channels
- Localization:
  - Compute response map from the weighted feature channels responses
  - Update discriminative weights for the feature channels
  - Estimate best position (max peak location + subpixel localization)
  - Estimate scale (standard approach used in correlation tracking)



Channel Regularized

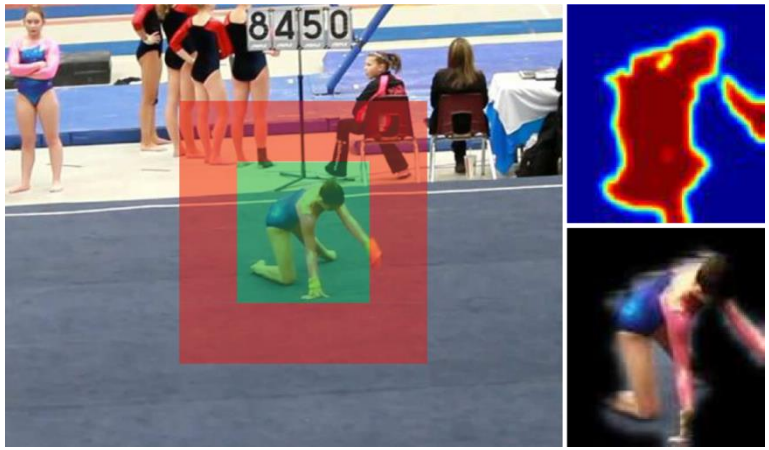
- Online weighting scheme of features
- The feature channels are weighted by:
  - their absolute contribution to the correct label response during filter learning, i.e. generative weighting
    (the higher contribution to the correct response the better)
  - ratio of first and second max peaks of the filter response during tracking, i.e. discriminative weighting
    (the larger difference between first and second peak the better)

Localization:

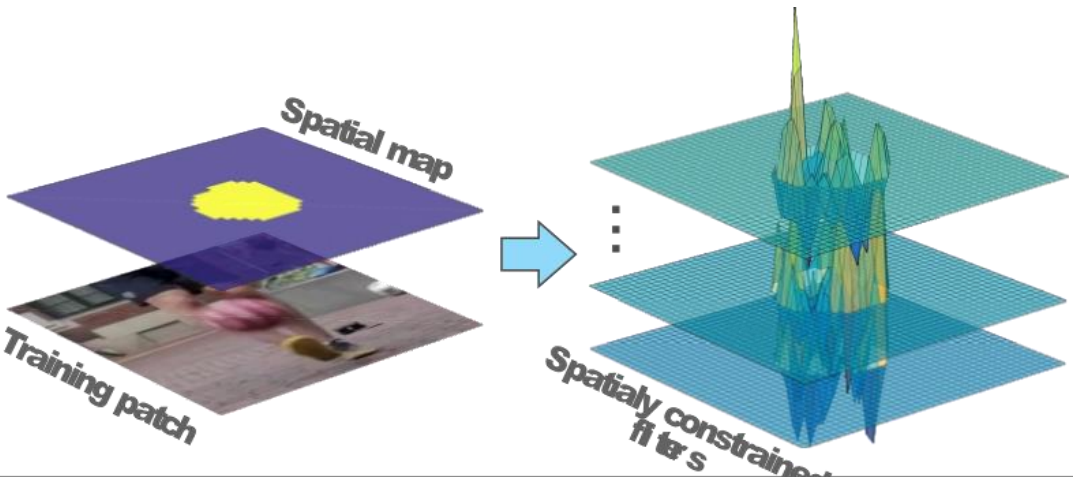


Spatial Regularization

- GrabCut based segmentation on estimated location (or initial position)
  → pisel-wise object mask
- Correlation filter is trained using the object mask, i.e. pixels that does not belong to the target are disabled
- Advantages:
- Reduces influence of bounding box object representation for object that undergoes e.g. rotation, deformation or aspect ratio change

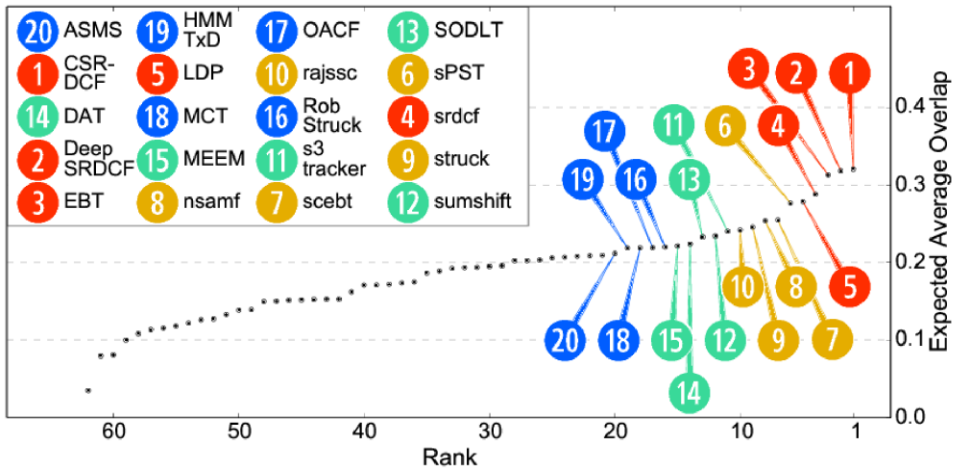


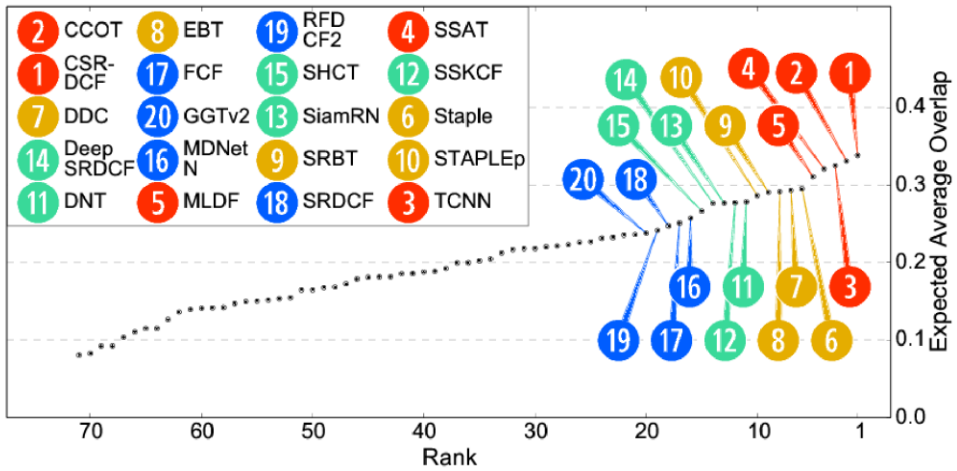
- Allows for large search region (i.e. large movement), since the filter training is focused by the object mask



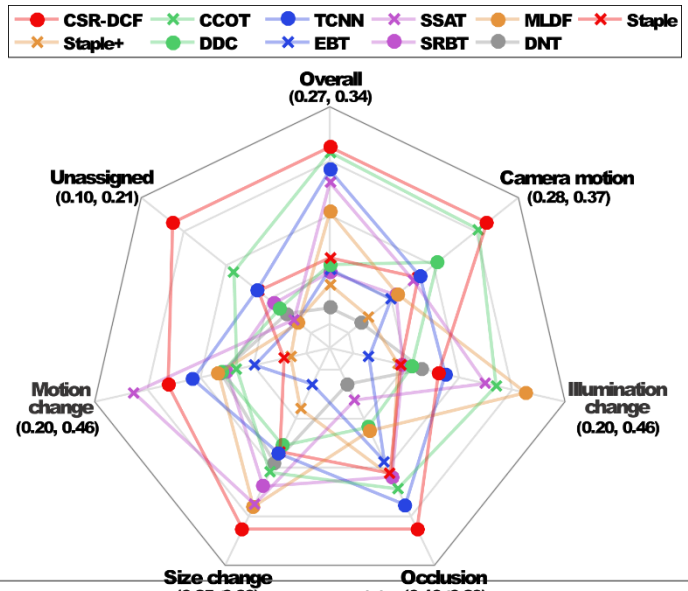
CSR-DCF

- Results for standard benchmarks: VOT2015 (left) and VOT2016 (right)



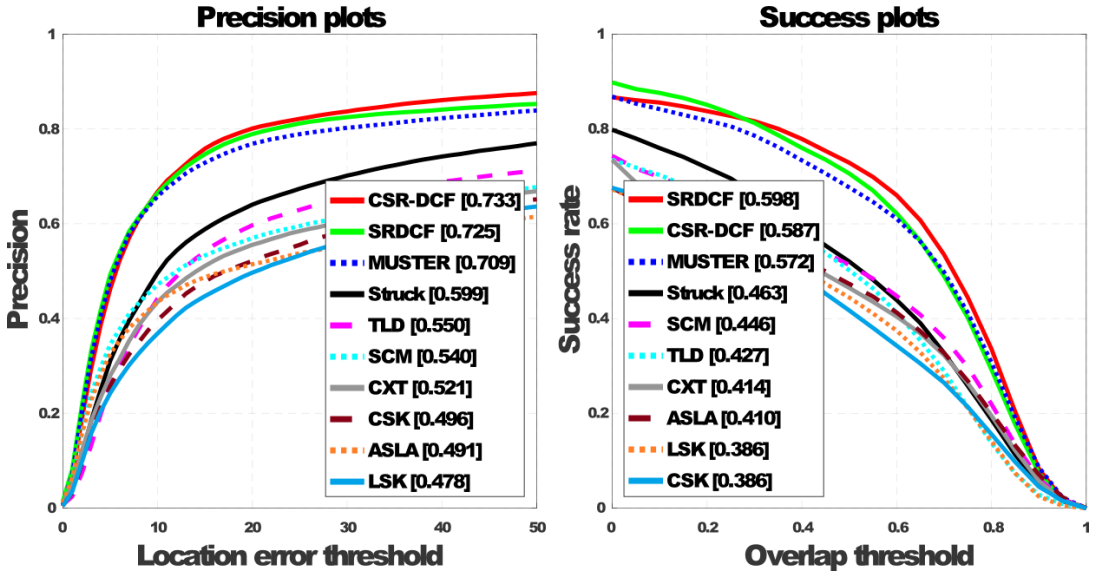


Tracker	Published at	EAO	A_{av}	R_{av}	fps
CSR-DCF	This work.	**0.338**	*0.51*	**0.85**	13.0
CCOT	ECCV2016	*0.331*	**0.52**	**0.85**	0.55
CCOT*	ECCV2016	0.274	**0.52**	*1.18*	1.0
SRDCF	ICCV2015	0.247	**0.52**	1.50	7.3
KCF	PAMI2015	0.192	0.48	2.03	**115.7**
DSST	PAMI2016	0.181	0.48	2.52	*18.6*
Struck	ICCV2011	0.142	0.42	3.37	8.5

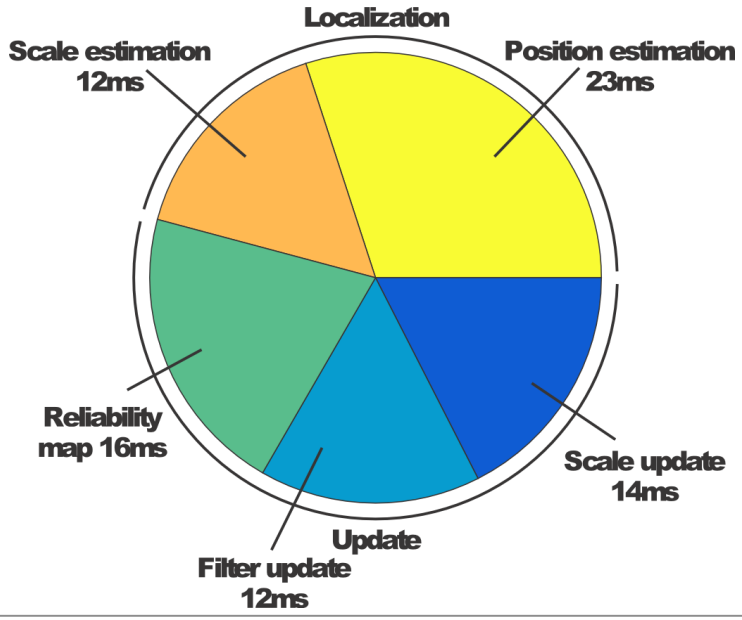


CSR-DCF

- Results for standard benchmark: OTB2015



- Speed analysis



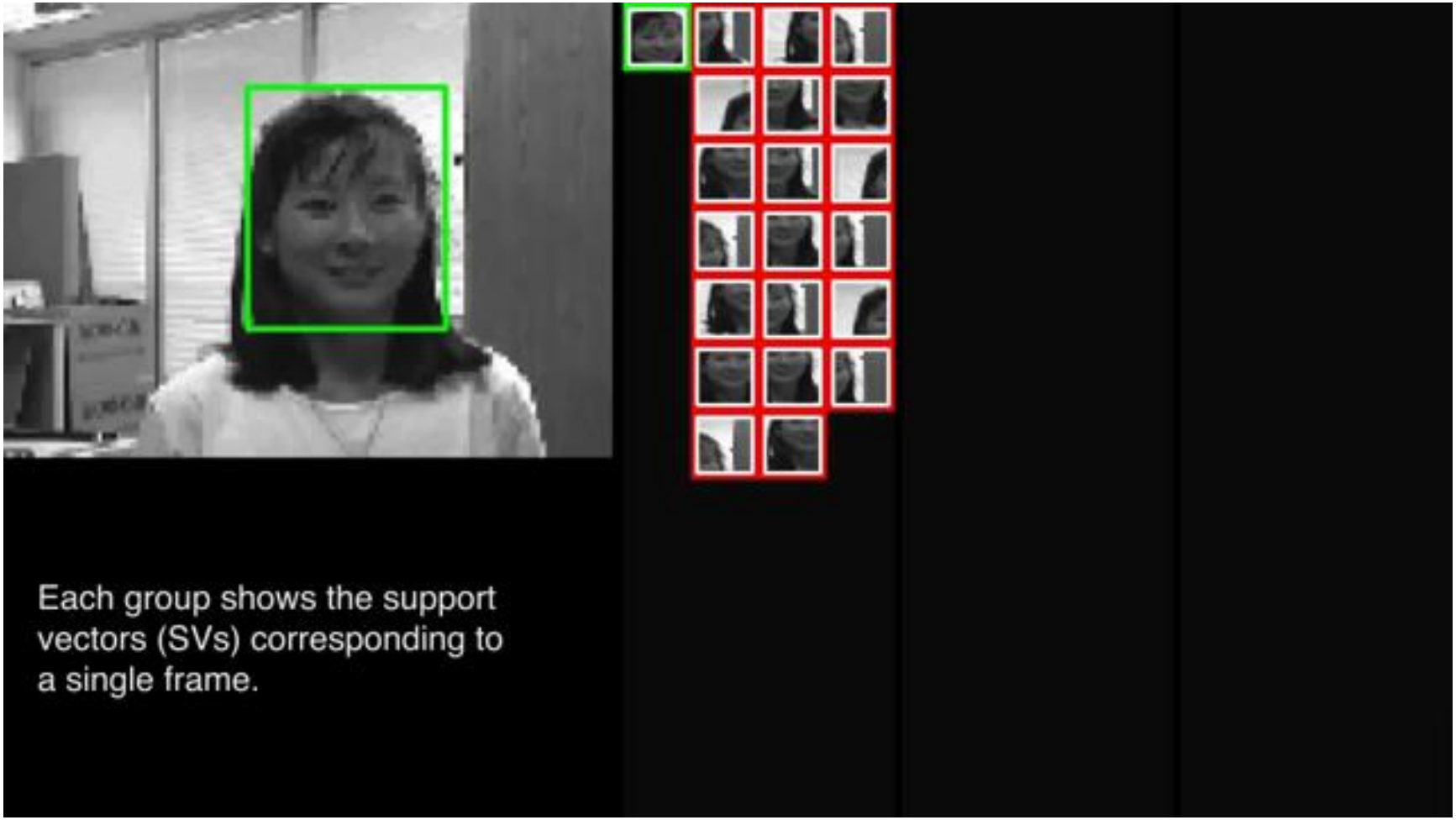
Discriminative Correlation Filters - Summary

- state-of-the-art performance on standard benchmark
- more efficient than competing DNN approaches

- cost function: discriminative, kernel based
- optimization:
  - efficient for translation
  - response not only at the location of the maximum

- issues with non-square objects
- transformations beyond translation handled ad-hoc
- outputs a global transformation:
  - providing only an approximate flow field
  - segmentation not part of the standard formulation

Other Tracking-by-Detection Trackers: Struck

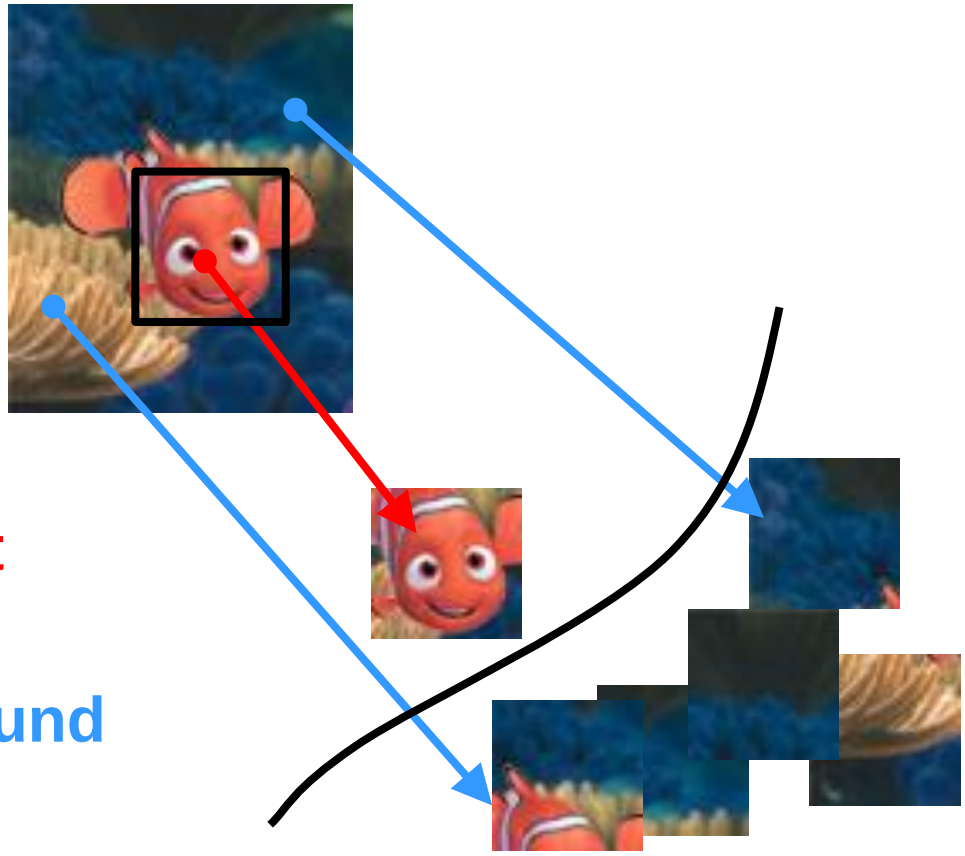


Sam Hare, Amir Saffari, Philip H. S. Torr, Struck: Structured Output Tracking with Kernels, ICCV 2011

Tracking as Detection:

- Tracking as binary classification

S. Avidan. **Ensemble tracking**. CVPR 2005.
J.Wang, et al. **Online selecting discriminative tracking features using particle filter**. CVPR 2005.



object

vs.

background

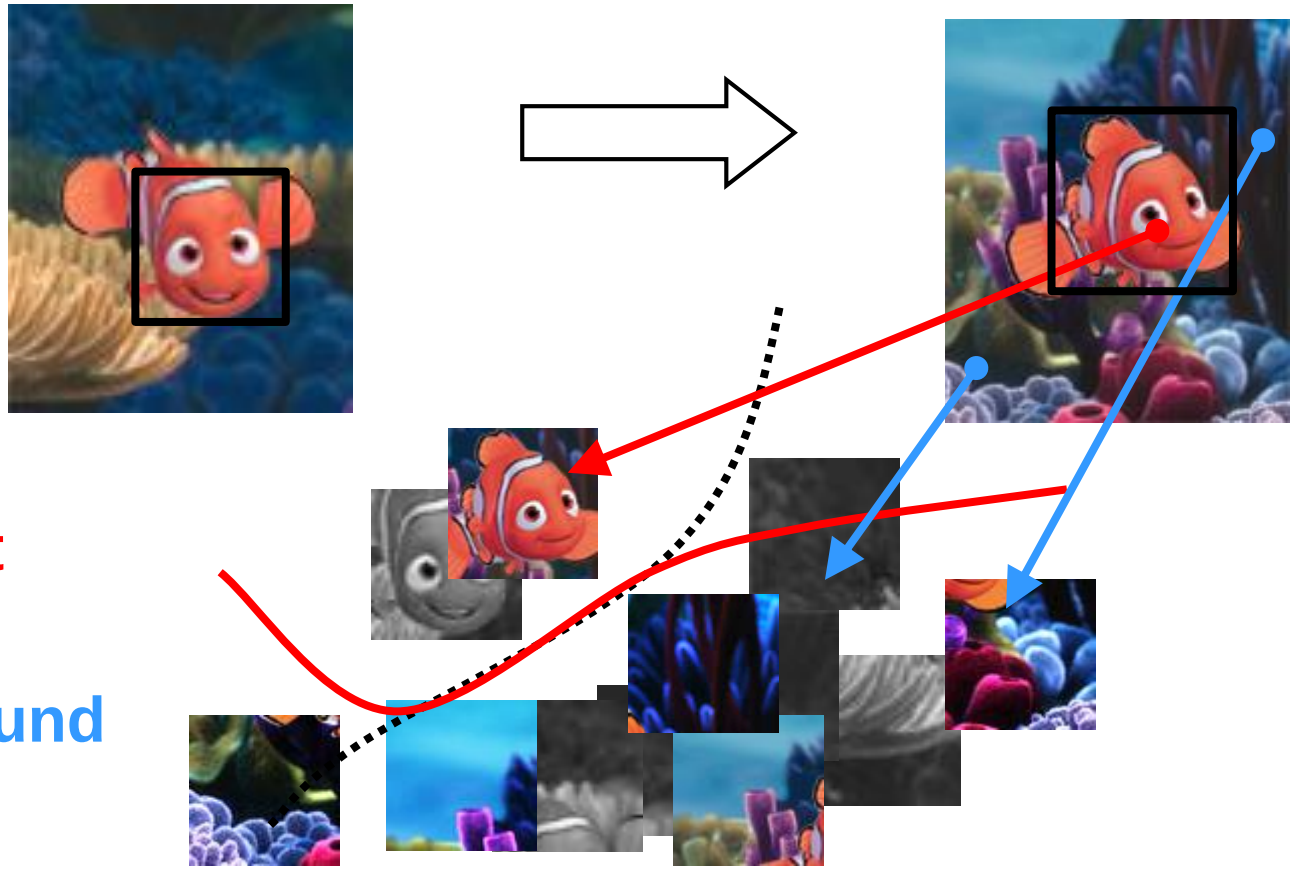
Slide credit: Helmut Grabner

Online discriminative tracking

- Tracking as binary classification

S. Avidan. **Ensemble tracking**. CVPR 2005.
J.Wang, et al. **Online selecting discriminative tracking features using particle filter**. CVPR 2005.

- Object and background changes are robustly handled by on-line updating!



object

vs.

background

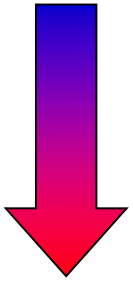
Slide credit: Helmut Grabner

Boosting for Feature Selection

Object Detector

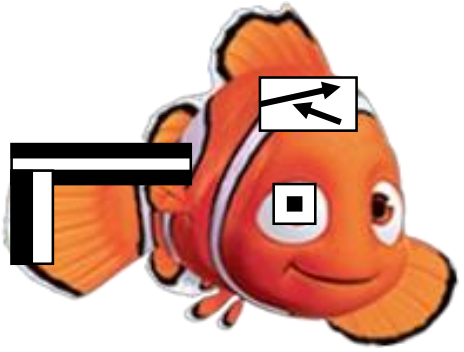
P. Viola and M. Jones. **Rapid object detection using a boosted cascade of simple features.** CVPR 2001.

Fixed Training set
General object detector



Object Tracker

On-line update
Object vs. Background



$$\text{sign}(\alpha_1 \cdot \boxed{} + \alpha_2 \cdot \boxed{} + \alpha_3 \cdot \boxed{} + \ldots)$$

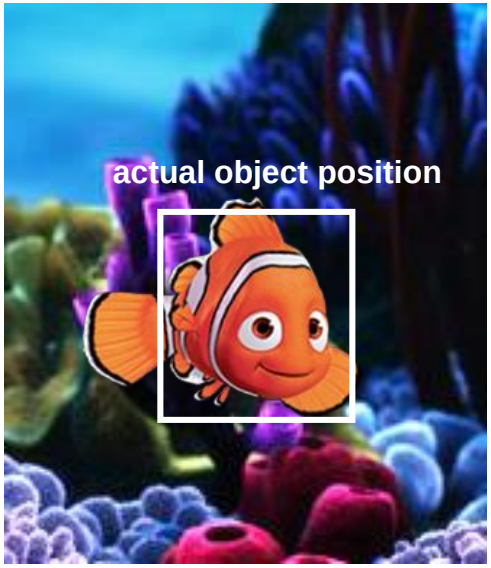
Combination of simple image features using Boosting as Feature Selection

On-Line Boosting for Feature Selection

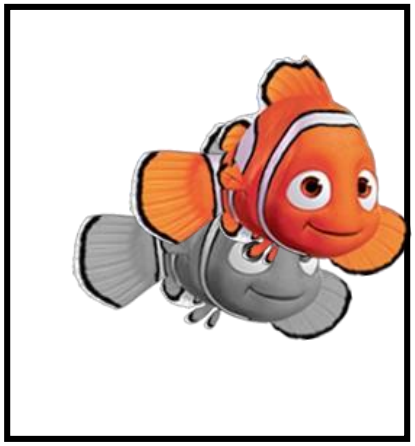
H. Grabner and H. Bischof. **On-line boosting and vision.** CVPR, 2006.

Slide credit: Helmut Grabner

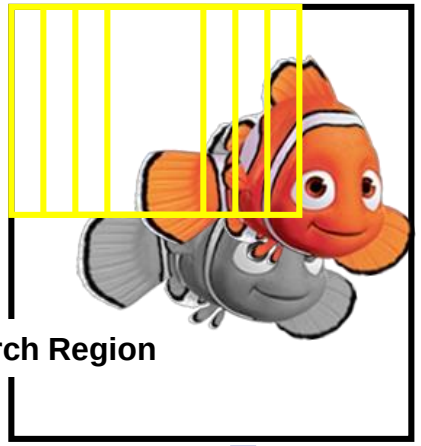
Tracking by online Adaboost



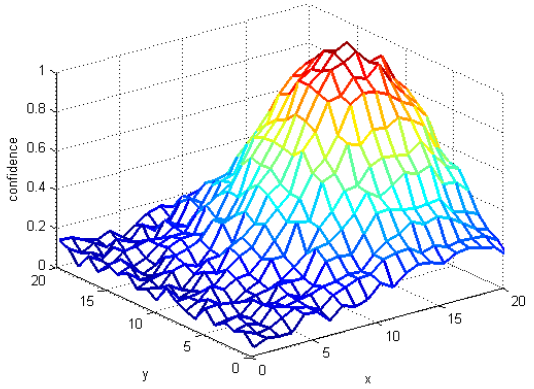
from time t to t+1



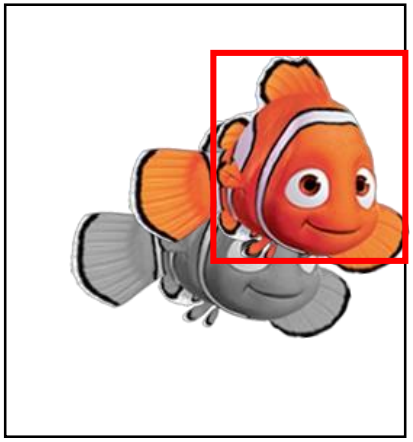
evaluate classifier on sub-patches



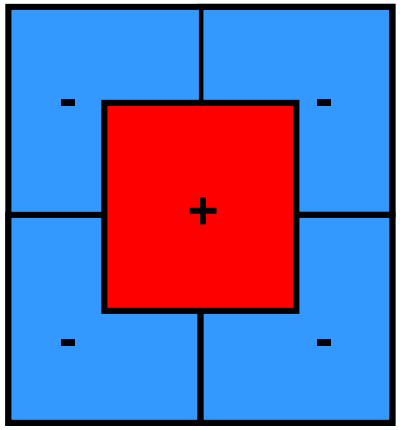
create confidence map



analyze map and set new object position



update classifier (tracker)

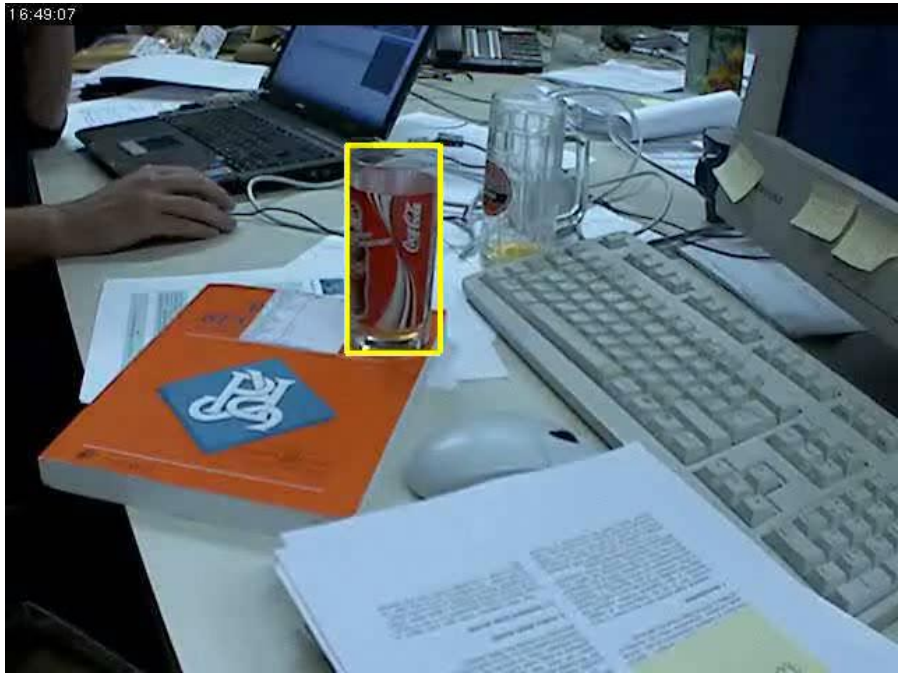


H. Grabner et. al., Real-Time Tracking via On-line Boosting . BMVC, 2006.

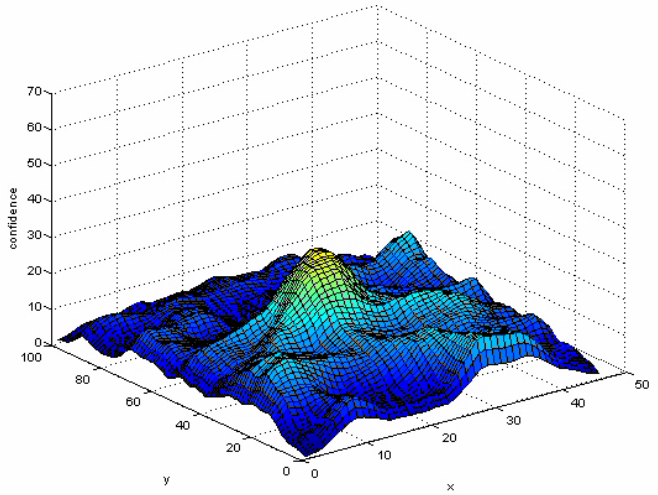
Slide credit: Helmut Grabner

Tracking by online Adaboost

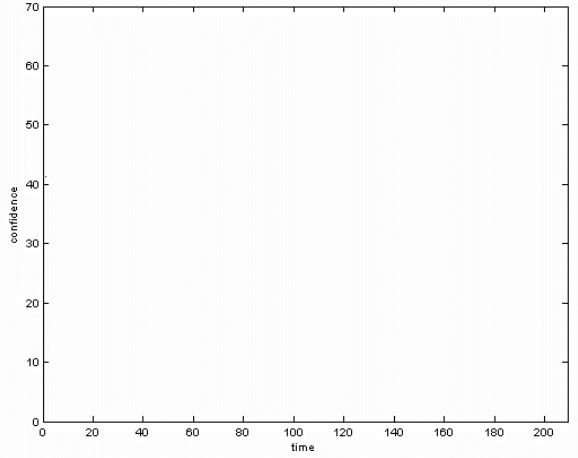
- Realtime performance
  - Fast feature computation
  - Efficient update of classifier



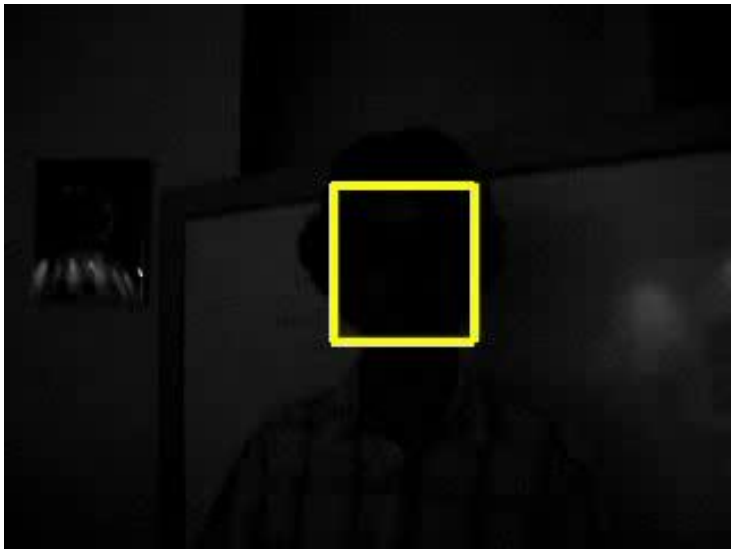
Confidence Map

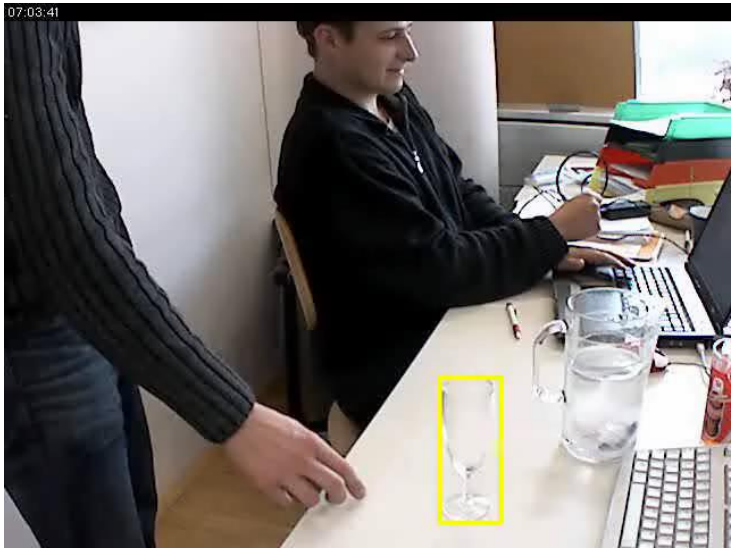


Max. Confidence Value



Tracking by online Adaboost

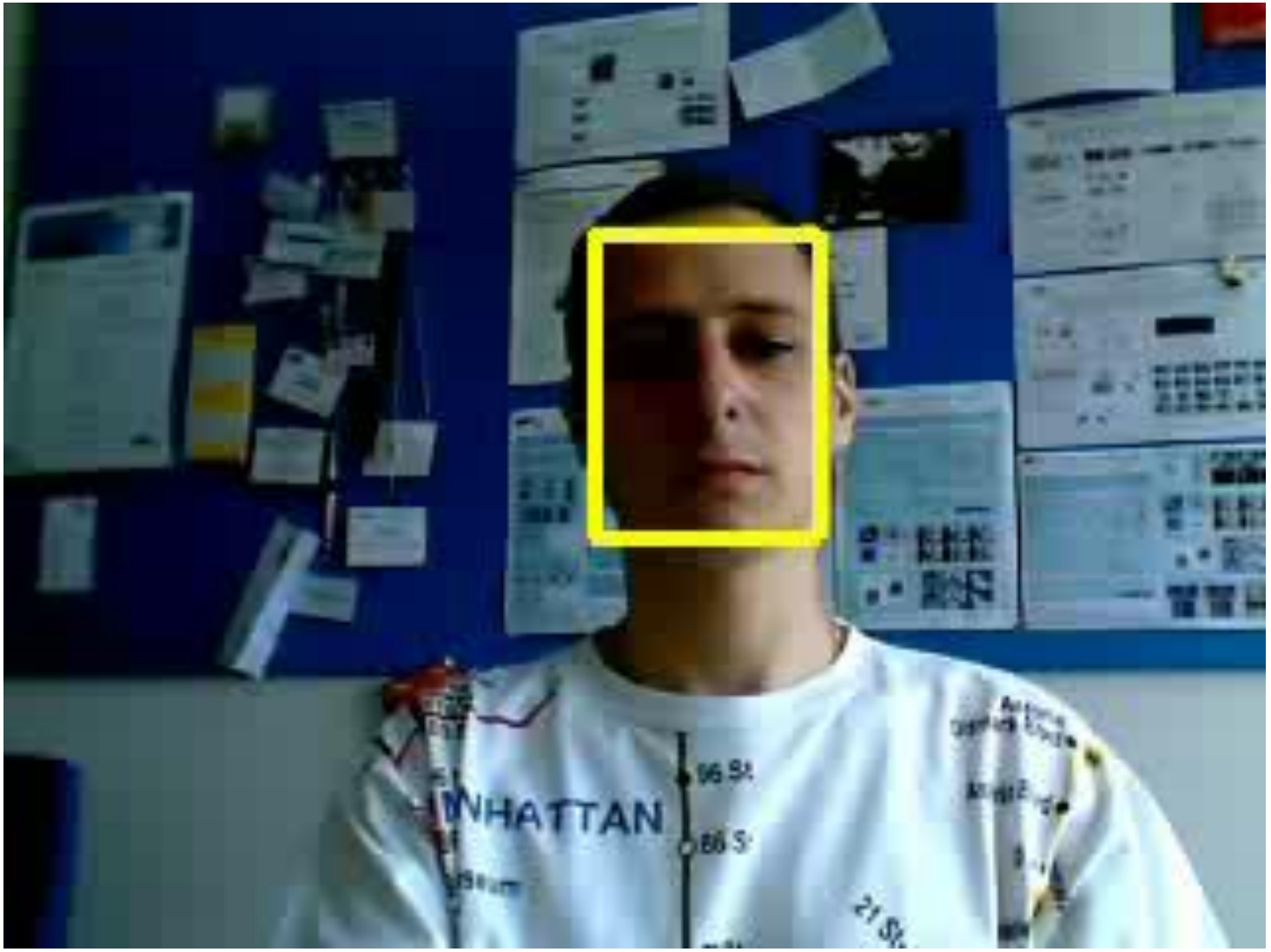




H. Grabner et. al., Real-Time Tracking via On-line Boosting . BMVC, 2006.

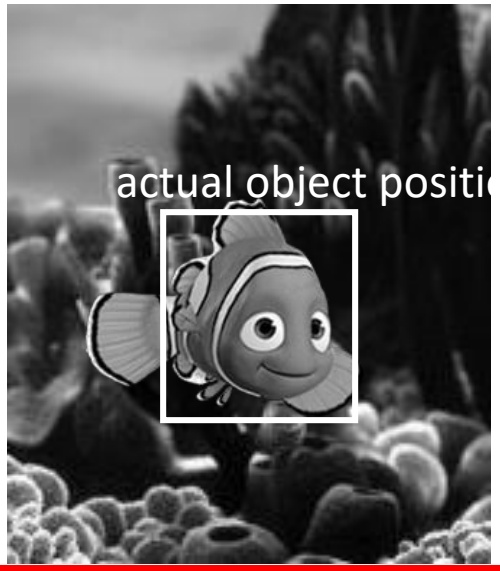
Slide credit: Helmut Grabner

Failure modes

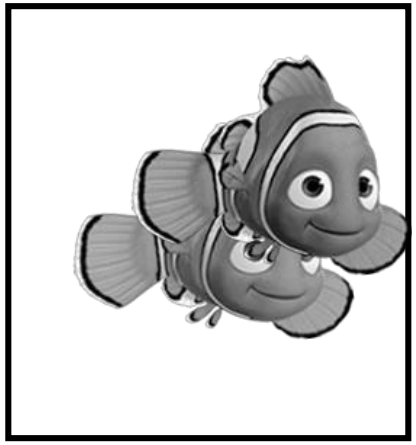


Slide credit: Helmut Grabner

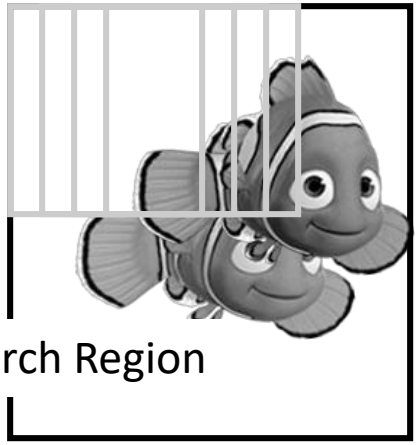
Why does it fail…



from time t to t+1

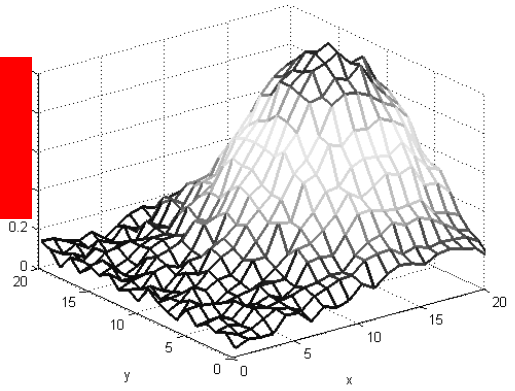


evaluate classifier on sub-patches

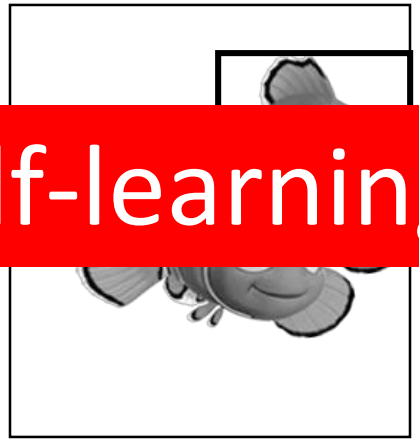


search Region

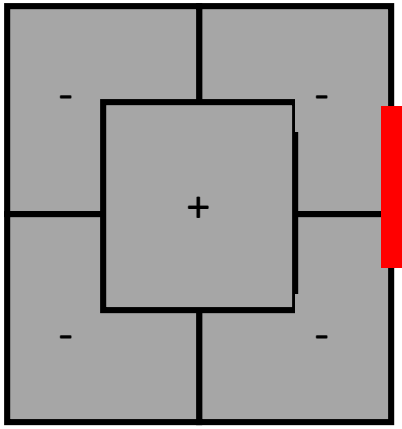
create confidence map



analyze map and set new object position



update classifier (tracker)

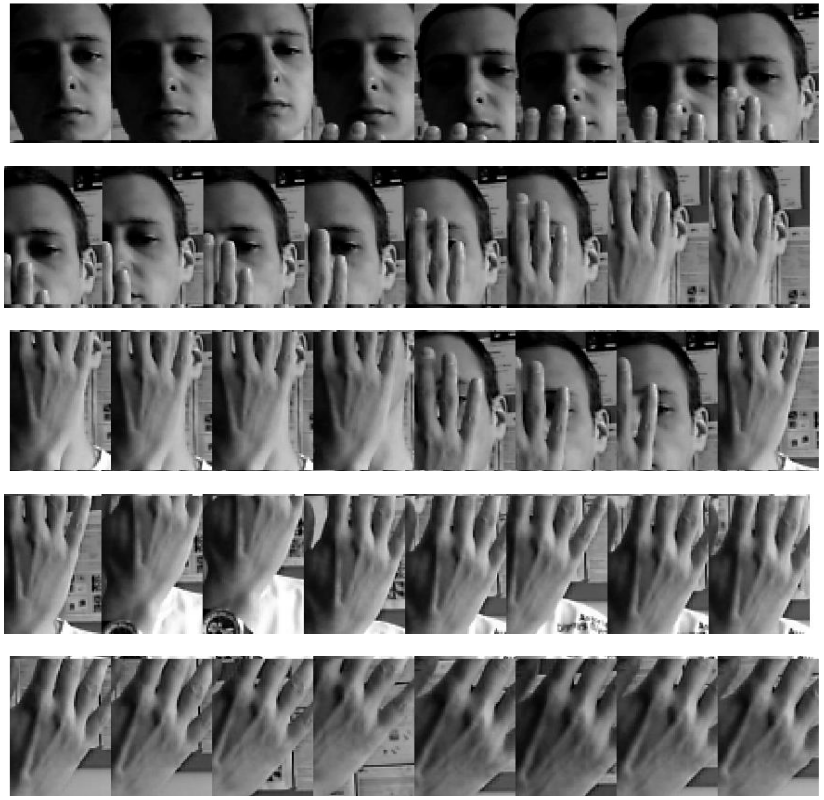


Self-learning!

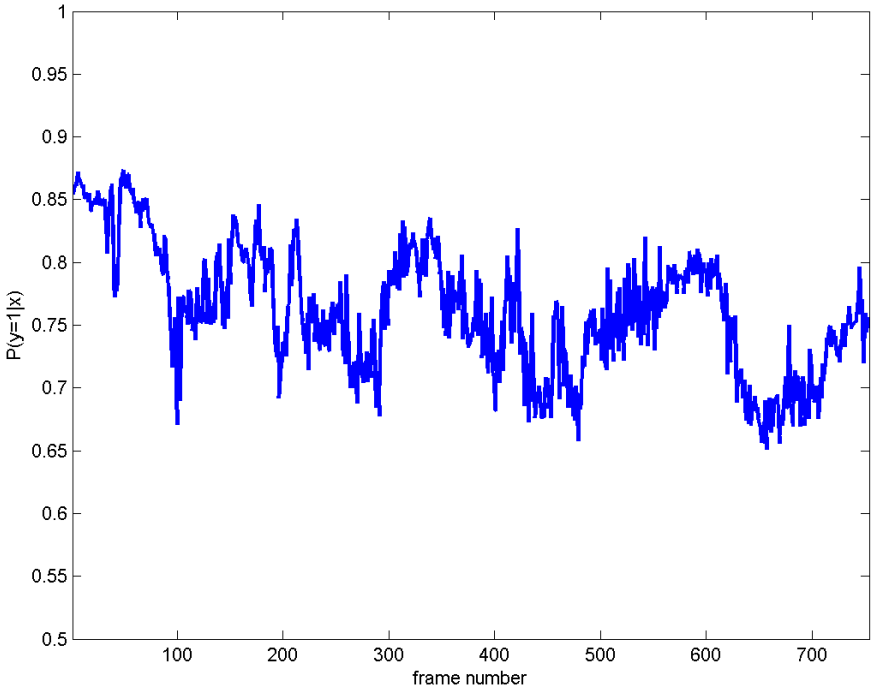
Slide credit: Helmut Grabner

Constant self-adaptation leads to drifting

Tracked Patches



Confidence



Slide credit: Helmut Grabner

Constant self-adaptation leads to drifting

- A poor update at time-step k may lead to poor localization at k+1
- This leads to even a poorer update, etc.

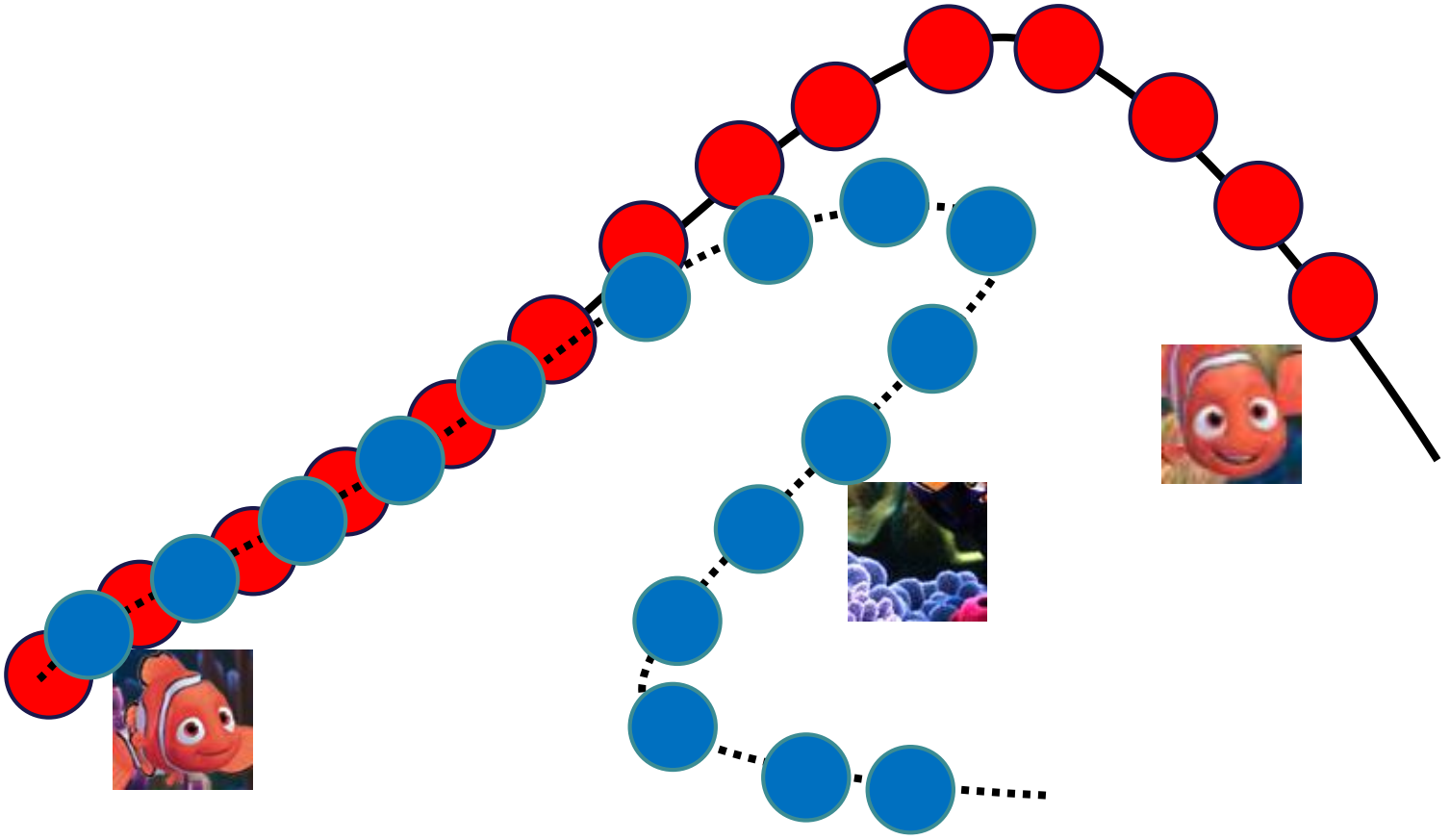
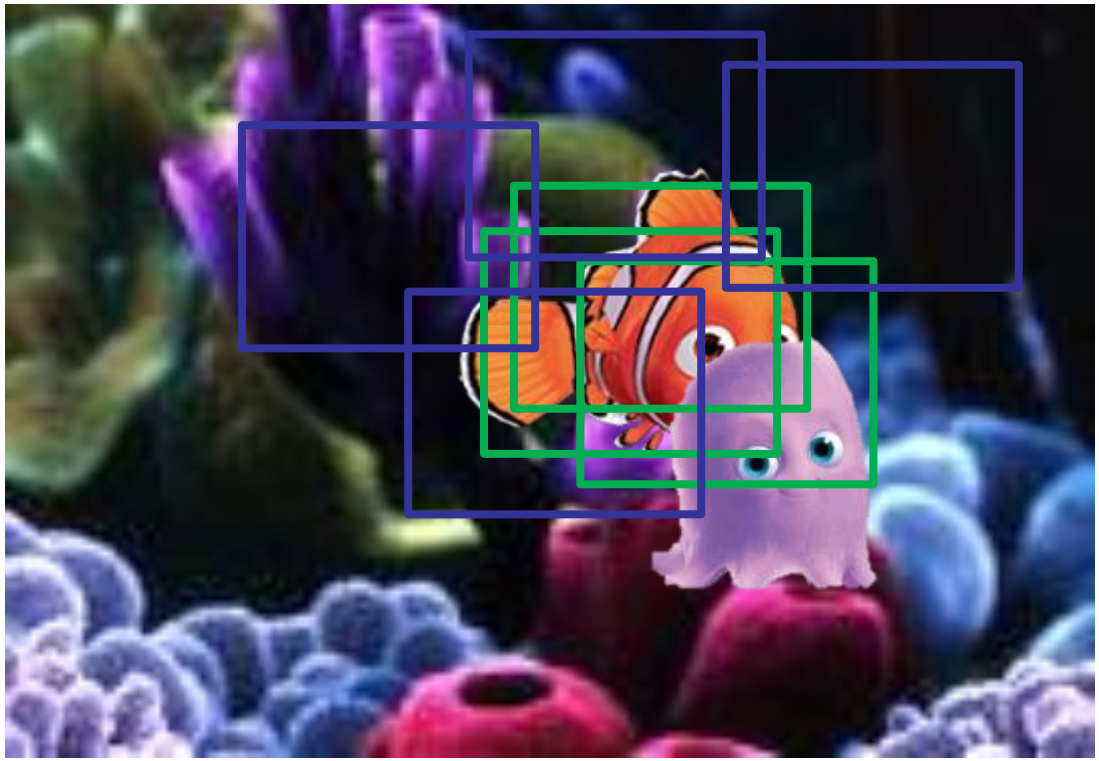
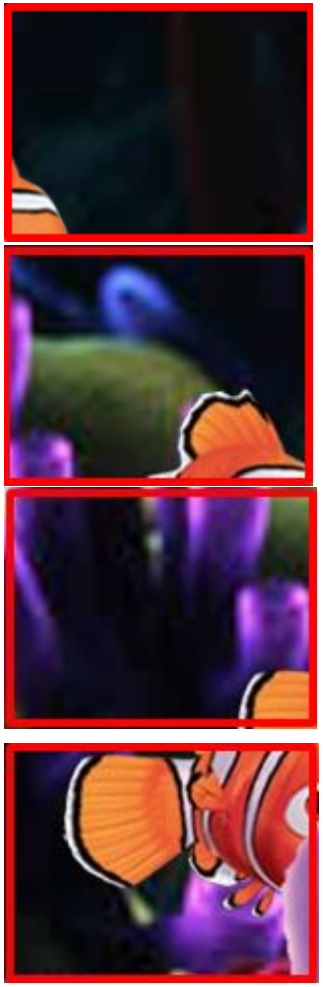


Image credit: Helmut Grabner

Do not trust all learning examples



negatives

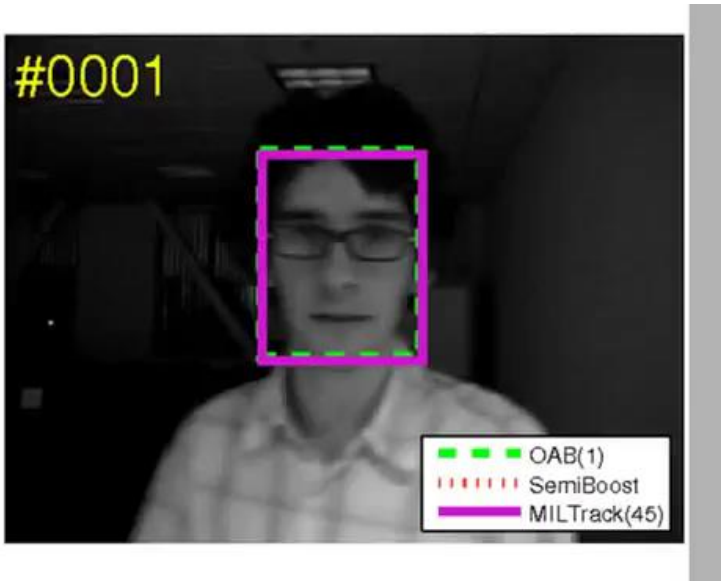


positives

Assume all negative examples are really negative

Assume positive examples might contain some negatives

Do not trust all learning examples



- Note that the online Adaboost failed *in this run* on the David sequence!
- Be sure that TMIL authors worked to show this, but it also says a lot about robustness of oAB to initialization!
- Code for TMIL available here.

Babenko et al.,"Robust Object Tracking with Online Multiple Instance Learning", TPAMI2011

References

- Key-point-based tracking:
  - [1] Özuysal, Calonder, Lepetit, Fua: Fast Keypoint Recognition Using Random Ferns. TPAMI2010
- Online Adaboost for tracking:
  - [2] H. Grabner et. al., Real-Time Tracking via On-line Boosting . BMVC, 2006.
- Multiple instance learning for tracking:
  - [3] Babenko et al.,"Robust Object Tracking with Online Multiple Instance Learning", TPAMI2011
- Structured SVM tracking:
  - [4] Hare, Saffari, Torr, Struck: Structured Output Tracking with Kernels, ICCV 2011
- Correlation filter tracking:
  - [5] Bolme, Beveridge, Draper, and Y. M. Lui. Visual Object Tracking using Adaptive Correlation Filters, CVPR 2010.
  - [6] Henriques, Caseiro, Martins, Batista, High-Speed Tracking with Kernelized Correlation Filters, TPAMI2015
  - [7] Danelljan, M., Hager, G., Khan, F.S., Felsberg, M.: Accurate scale estimation for robust visual tracking, BMVC2014

The TLD (PN) Long-Term Tracker

The TLD (PN) Long-Term Tracker

includes:

- adaptive tracker(s) (FOT)
- object detector(s)
- P and N event recognizers for unsupervised learning generating (*possibly incorrectly)* labelled samples
- an (online) supervised method that updates the detector(s)

Operation:

1. Train **Detector on** the first patch
2. Runs **TRACKER** and **DETECTOR** in parallel
3. Update the object **DETECTOR using P-N learning**

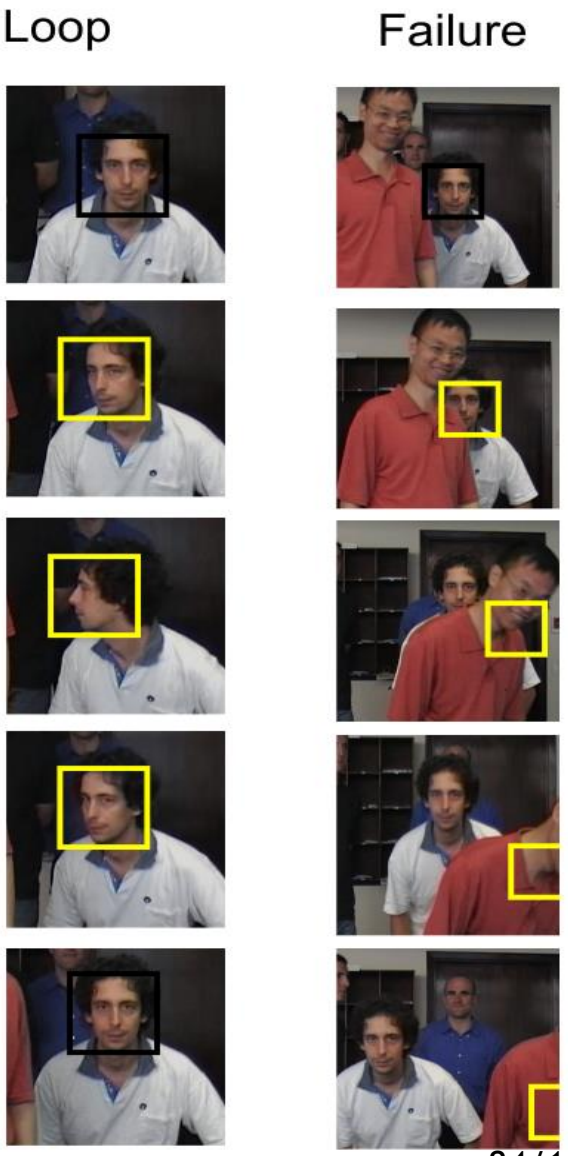
TLD a.k.a. PN Tracker a.k.a. “The Predator”

Z. Kalal, K.Mikolajczyk, J. Matas: Tracking-Learning-Detection. IEEE T PAMI 34(7): 1409-1422 (2012)

P-event: “Loop”

- exploits **temporal** structure
- turns drift of adaptive trackers into a
- **Assumption:**
  If an adaptive tracker fails, it is unlike
- **Rule:**
  Patches from a track starting and enc
  model (black), ie. are validated by th
  added to the model

Tracker responses

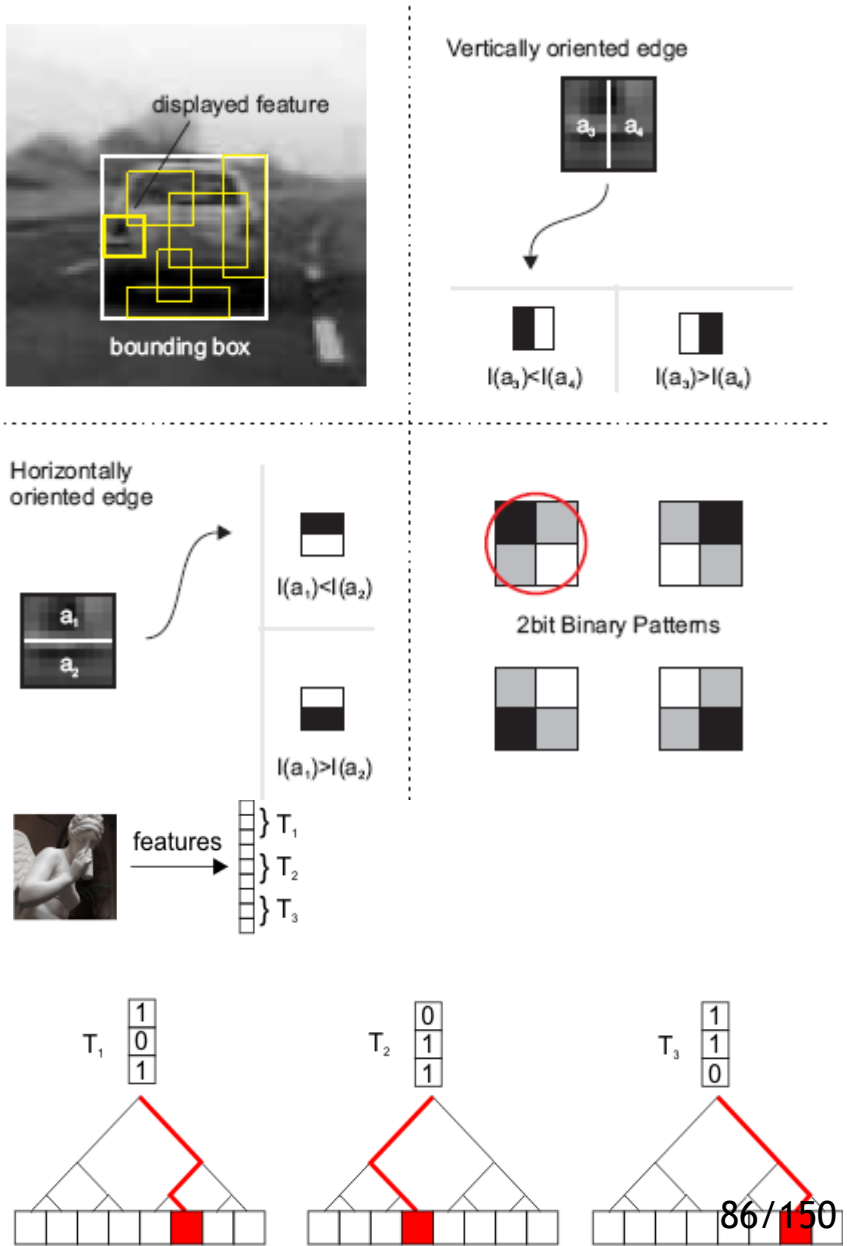


N-event: Uniqueness Enforcement

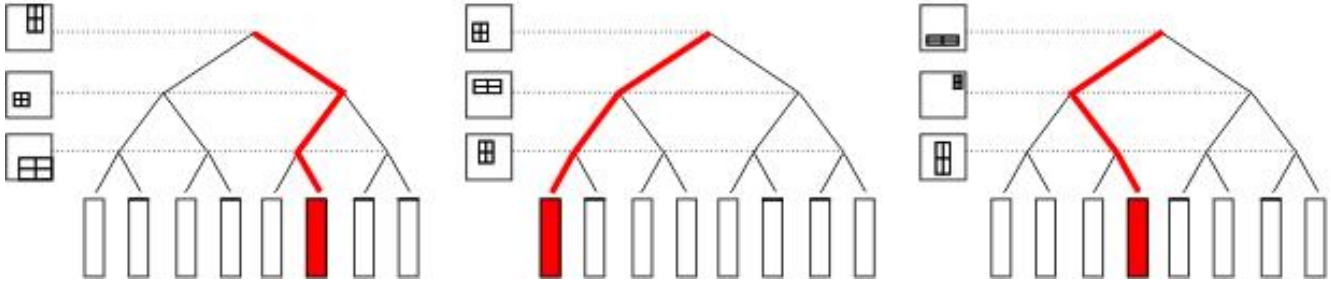
- exploits **spatial** structure
- **Assumption:**
  Object is unique in a single frame.
- **Rule:**
  If *the tracker is in model*, all other detections within the current frame (red) are assumed wrong → prune from the model

The Detector

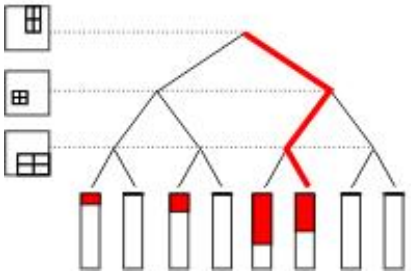
- Scanning window
- Randomized forest
- Trees implemented as ferns [Lepetit 2005]
- Real-time training/detection 20 fps on 320x240 image
- High accuracy, 8 trees of depth 10
- 2bit Binary Patterns Combined Haar and LBP features
- Tree depth controls complexity & discriminability; currently not adaptive

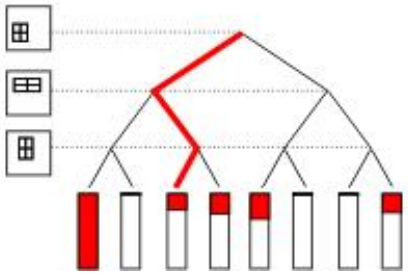


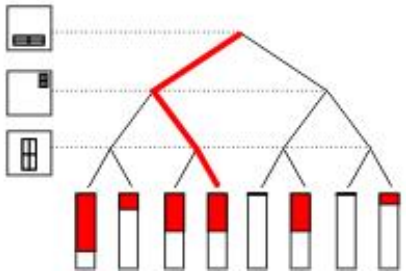

Growing event

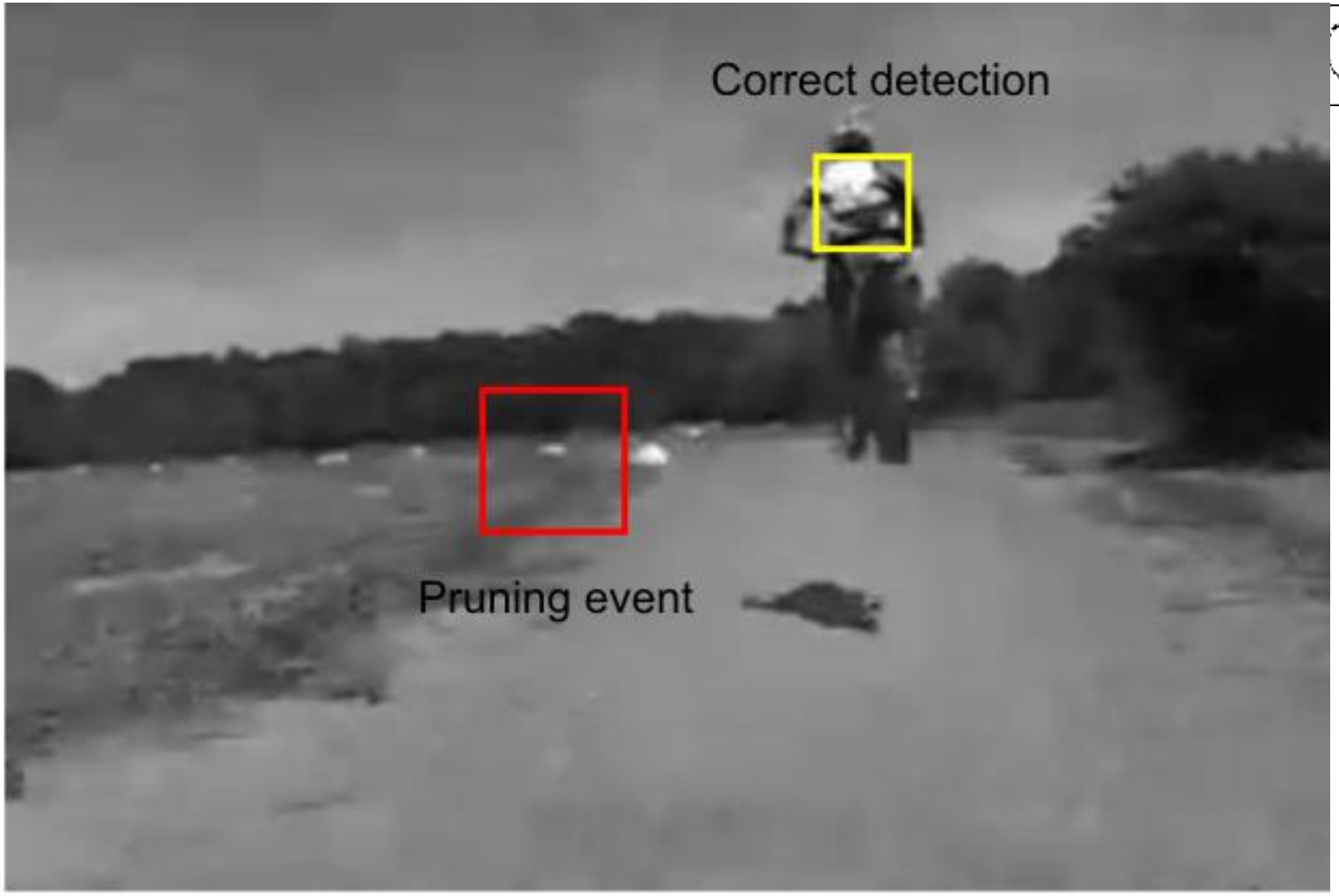


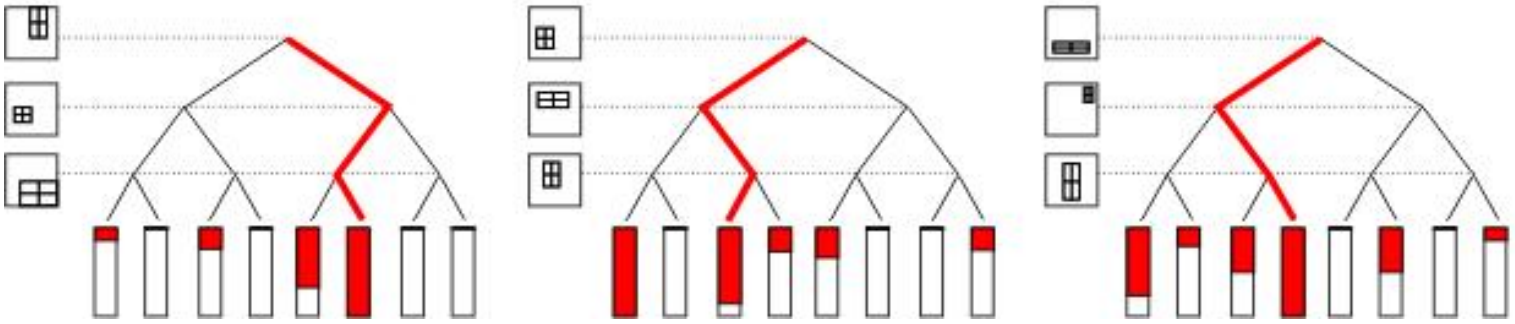

Correct detection



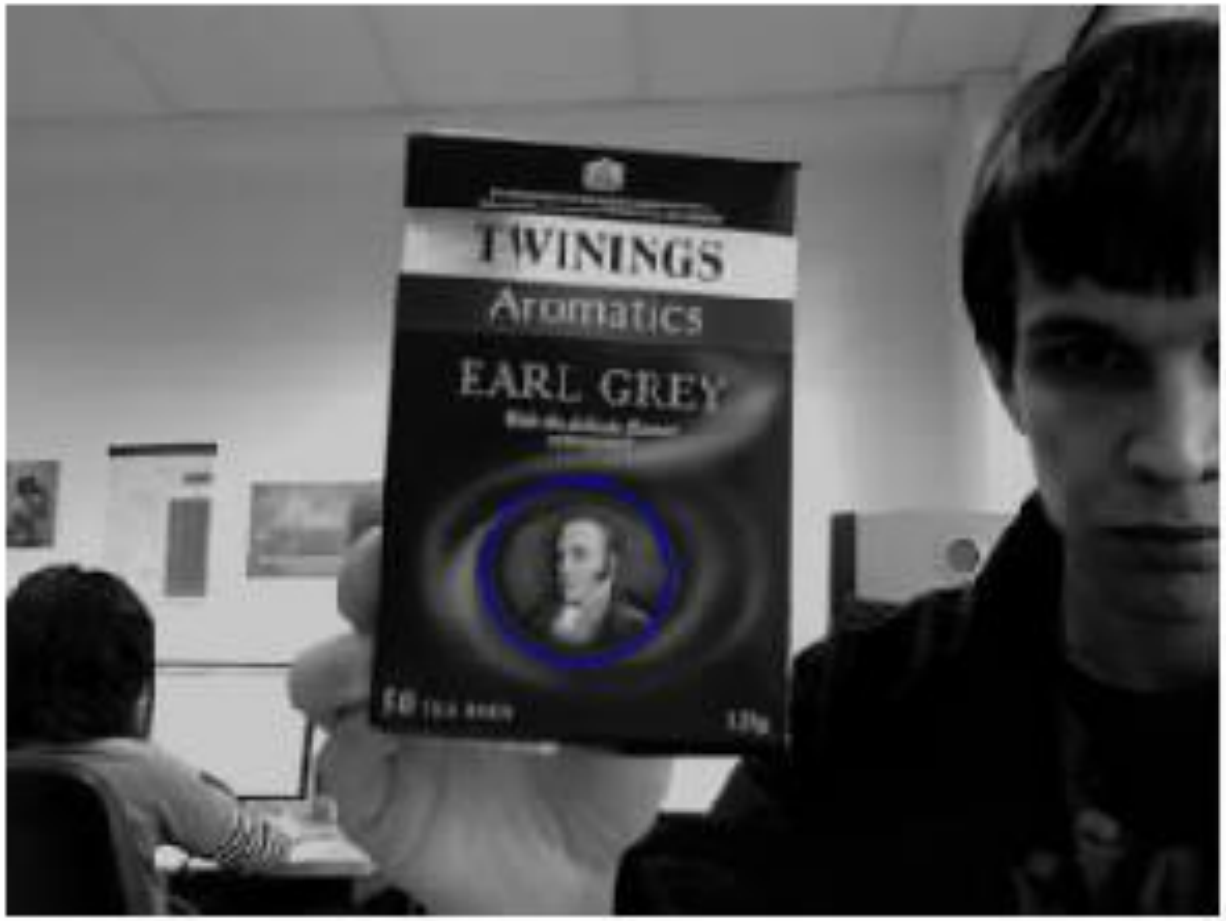


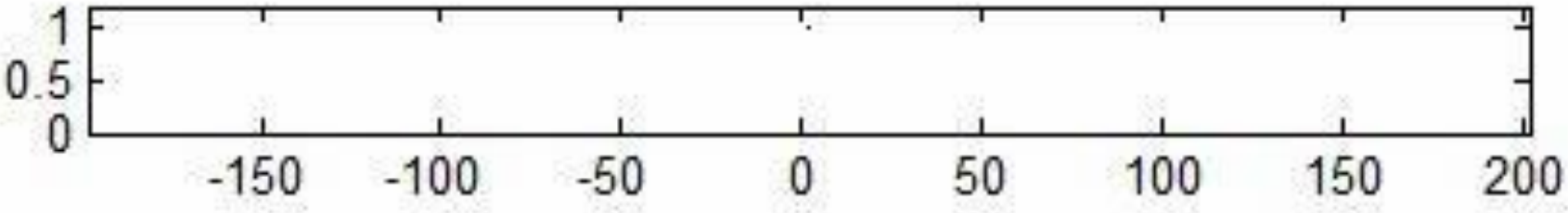



Correct detection
Pruning event

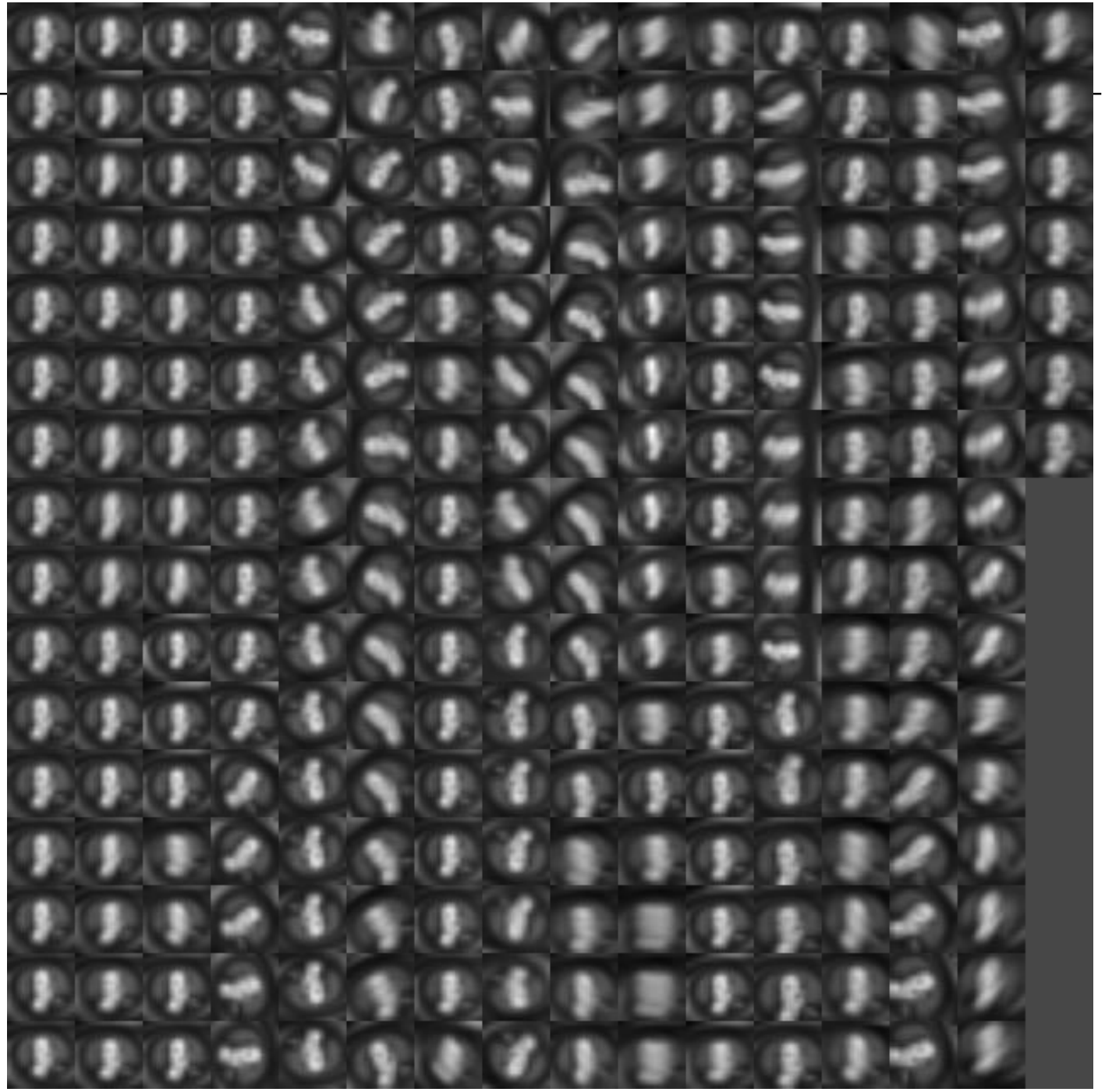


2





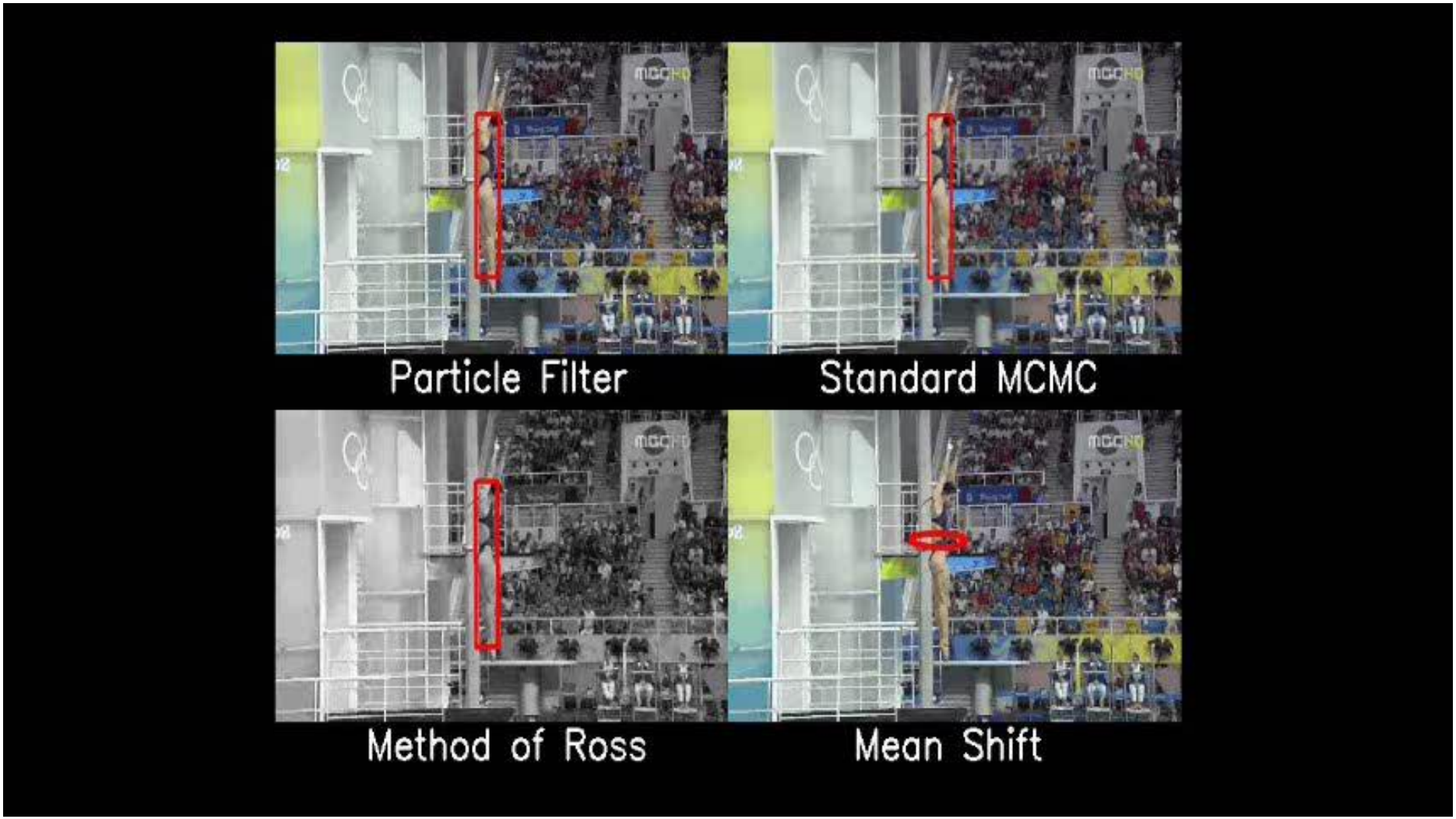
More TLD videos



Evaluation of Trackers

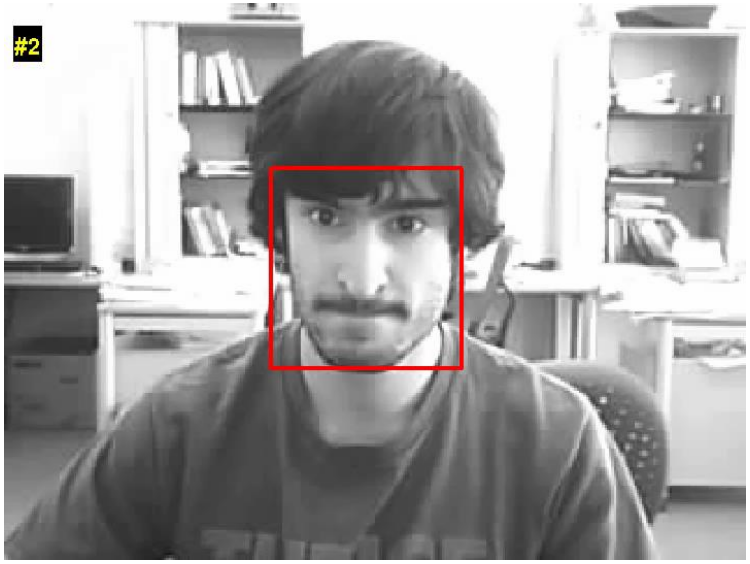
Tracking: Which methods work?

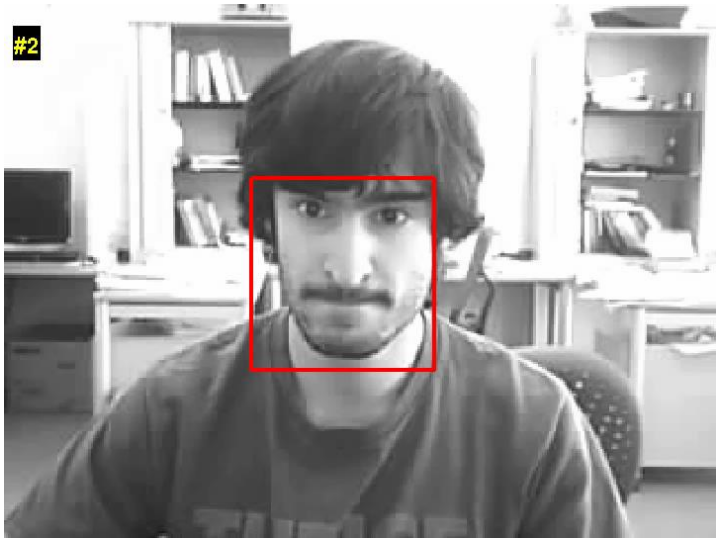
Tracking: Which methods work?

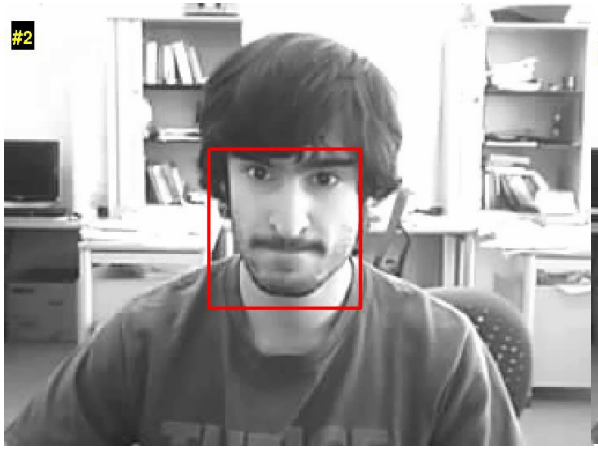


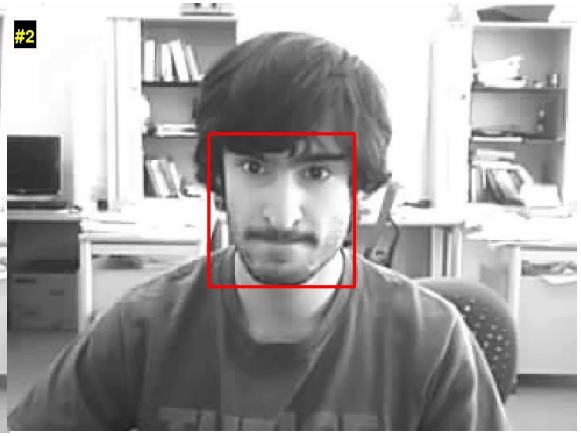
What works? "The zero-order tracker" ☺

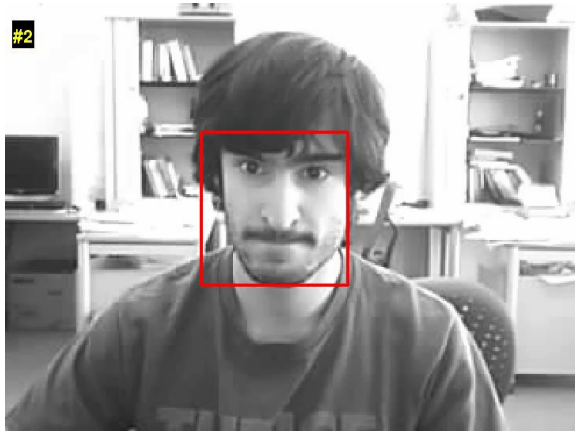
Compressive Tracker (ECCV'12). Different runs.



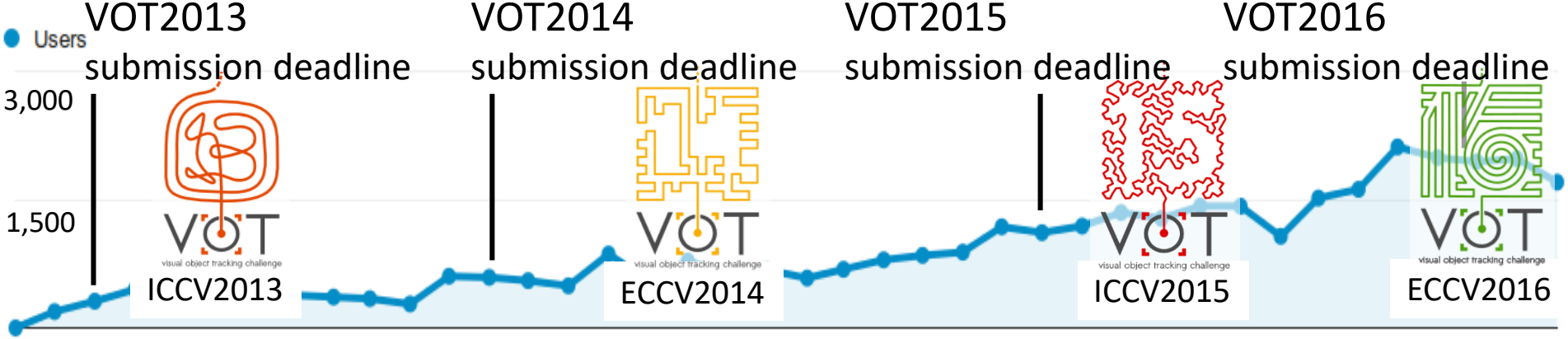


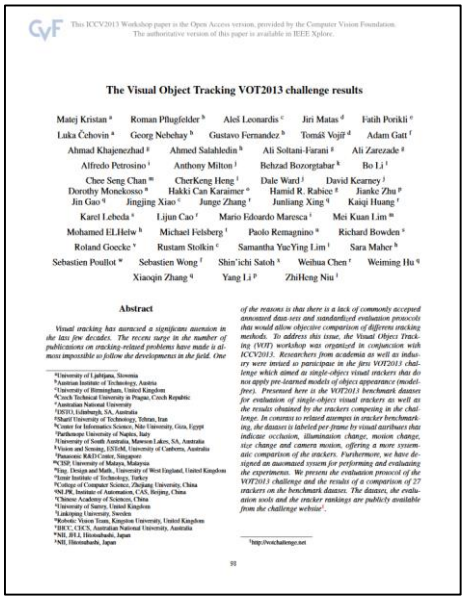






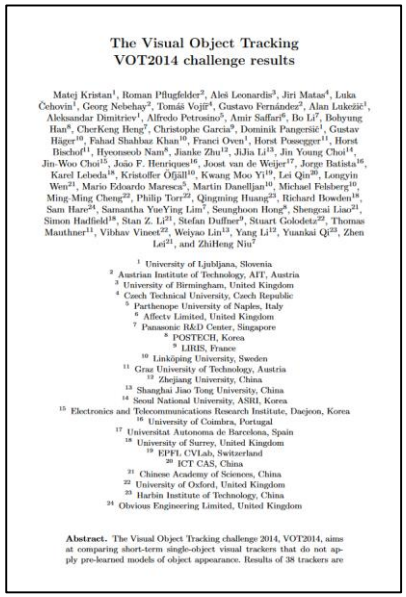
VOT community evolution





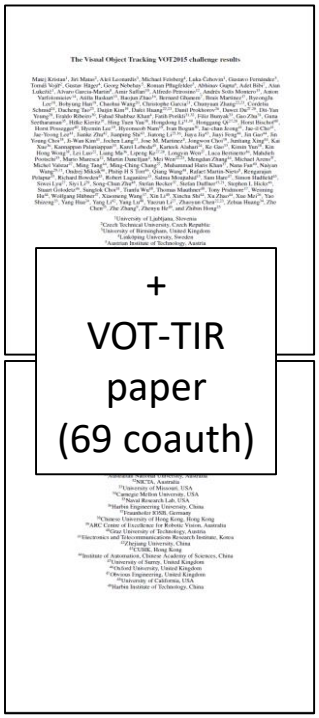
The Visual Object Tracking VOT2013 challenge results

51 Coauthors, 14pgs



The Visual Object Tracking VOT2014 challenge results

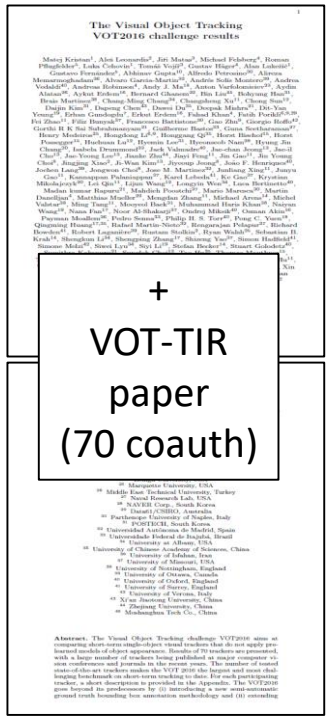
57 Coauthors, 27pgs



The Visual Object Tracking VOT2015 challenge results

+ VOT-TIR paper (69 coauth)

128 Coauthors, 24pgs



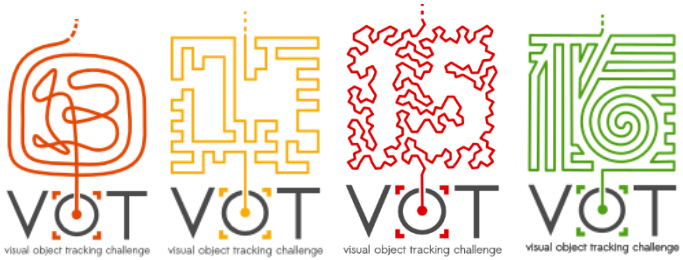
The Visual Object Tracking VOT2016 challenge results

+ VOT-TIR paper (70 coauth)

141 Coauthors, 44pgs

VOT challenge evolution

	Perf. Measures	Dataset size	Target box	Property	Trackers tested
VOT2013	ranks, A, R	16, s. manual	manual	per frame	27
VOT2014	ranks, A, R, EFO	25, s. manual	manual	per frame	38
VOT2015	EAO, A, R, EFO	60, fully auto	manual	per frame	62 VOT, 24 VOT-TIR
VOT2016	EAO, A, R, EFO	60, fully auto	auto	per frame	70 VOT, 24 VOT-TIR



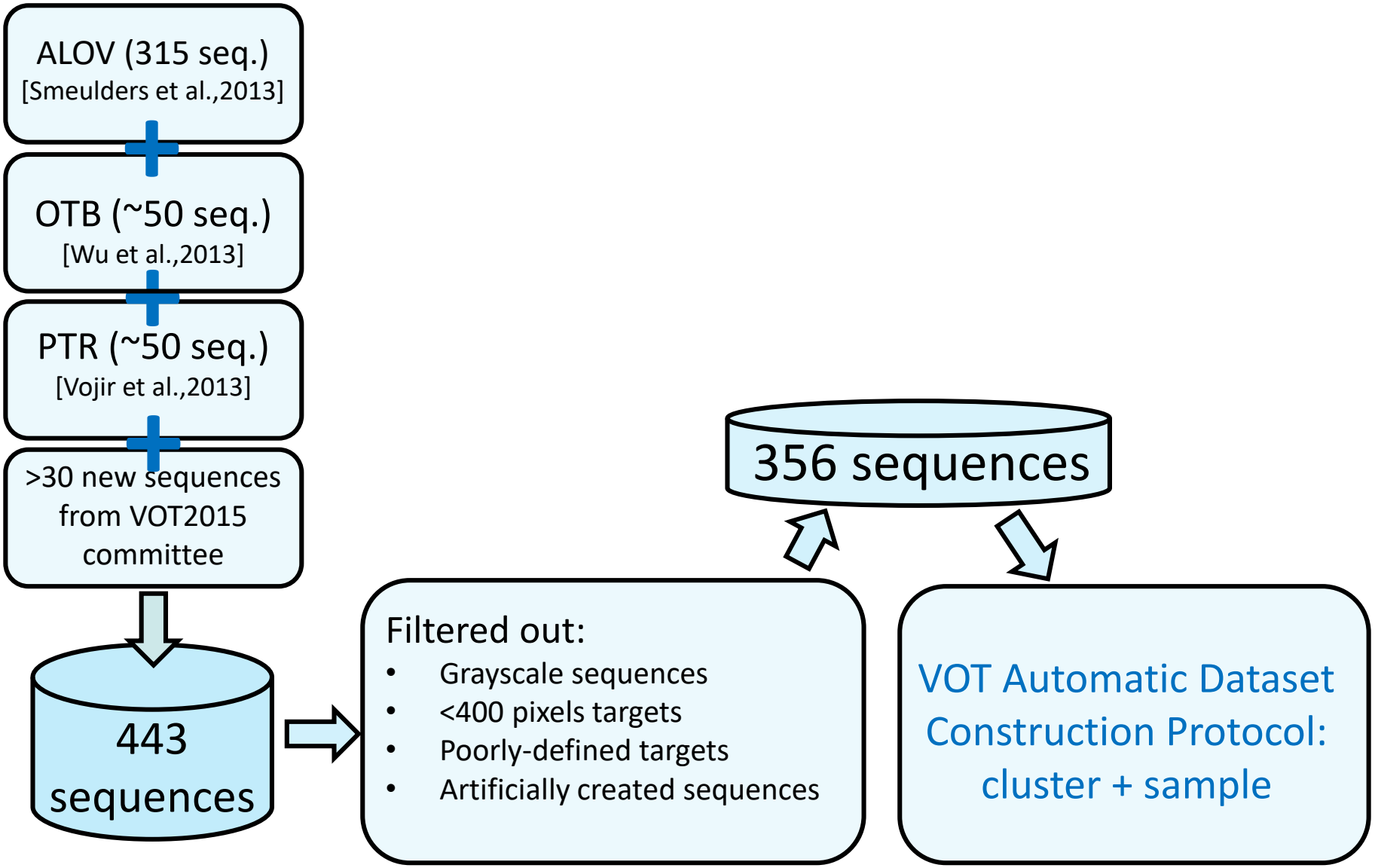
- Gradual increase of dataset size
- Gradual refinement of dataset construction
- Gradual refinement of performance measures
- Gradual increase of tested trackers

Class of trackers tested

- Single-object, single-camera

- Short-term:
  - Trackers performing without re-detection
- Causality:
  - Tracker is not allowed to use any future frames
- No prior knowledge about the target
  - Only a single training example – BBox in the first frame
- Object state encoded by a bounding box

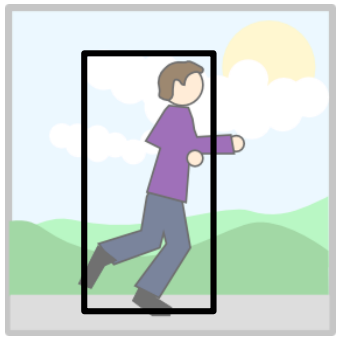
Construction (1/3): Sequence candidates



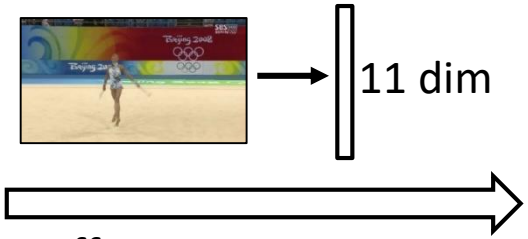
Construction (2/3): Clustering

- Approximately annotate targets
- 11 global attributes estimated automatically for 356 sequences

  (e.g., blur, camera motion, object motion)



Feature encoding

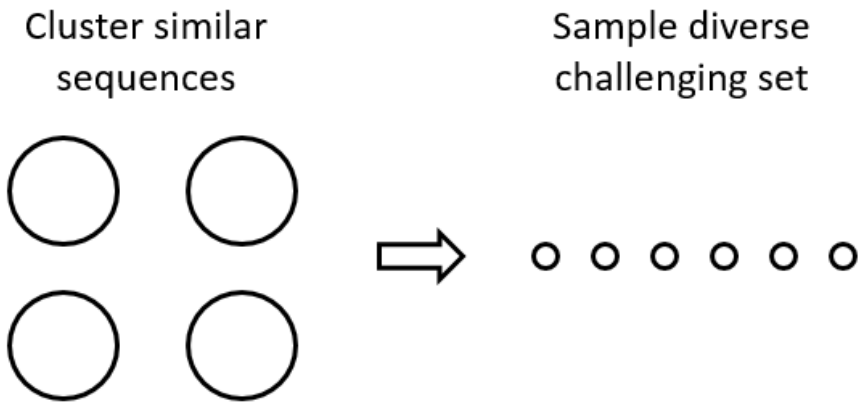


Cluster similar sequences

- Cluster into K = 28 groups (automatic selection of K)

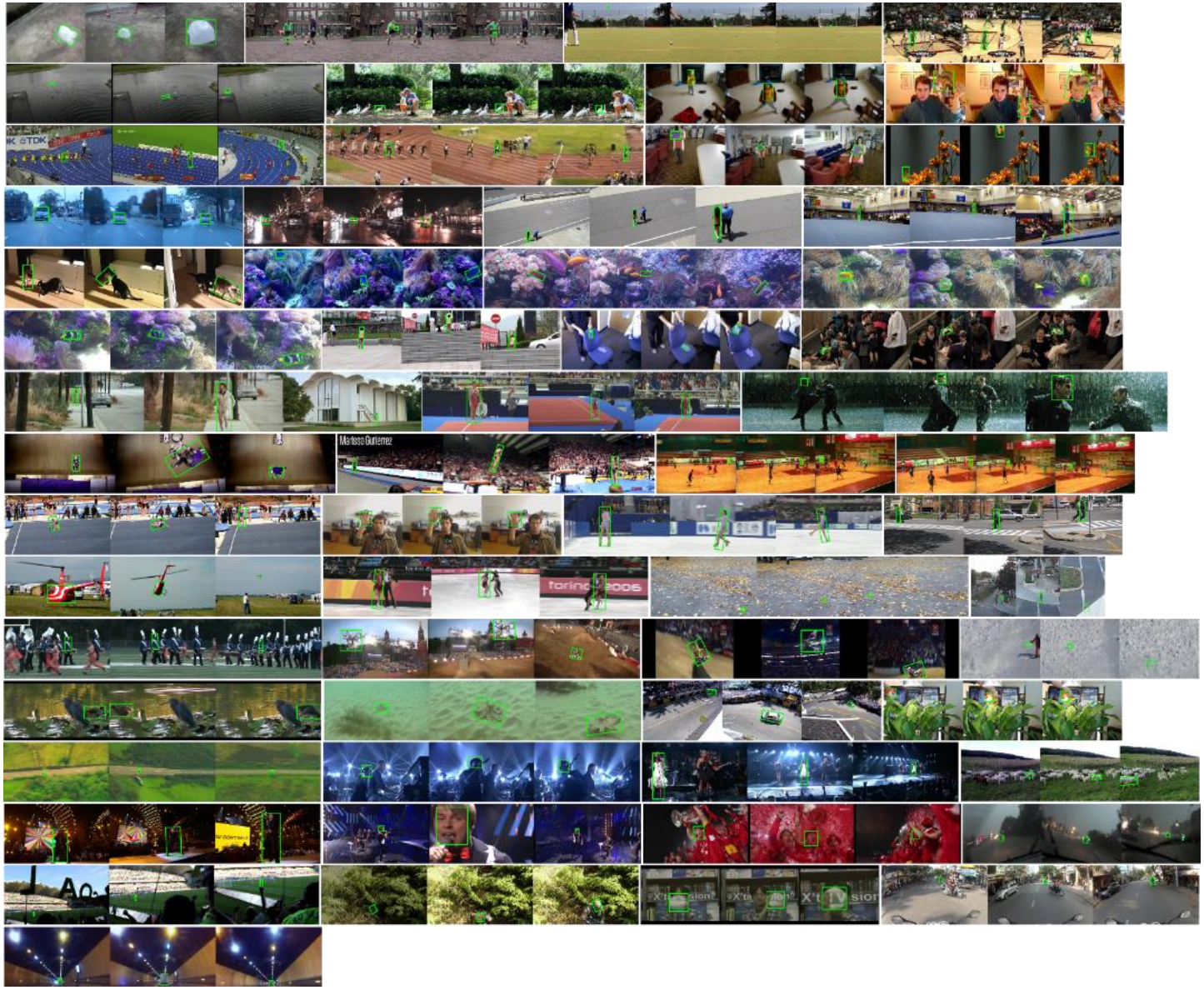
Construction (3/3): Sampling

- Requirement:
  - Diverse visual attributes
  - Challenging subset



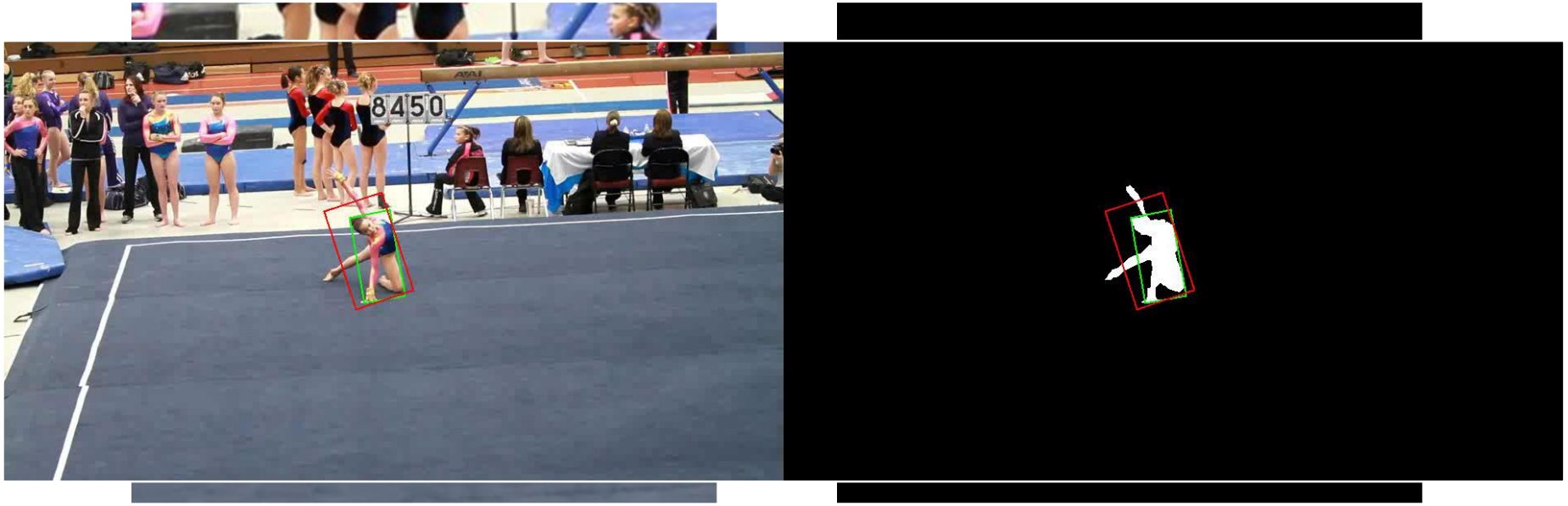
- Global visual attributes: computed
- Tracking difficulty attribute: Applied FoT, ASMS, KCF trackers
- Developed a sampling strategy that sampled challenging sequences while keeping the global attributes diverse.

VOT2015/16 dataset: 60 sequences



Object annotation

- Automatic bounding box placement
  1. Segment the target (semi-automatic)
  2. Automatically fit a bounding box by optimizing a cost function



Sequence ranking

- Among the most challenging sequences

Matrix (A_f = 0.33, M_f = 57)

Rabbit(A_f = 0.31, M_f = 43)

Butterfly ($A_f = 0.22, M_f = 45$)

- Among the easiest sequences

Singer1 ($A_f = 0.02, M_f = 4$)

Octopus ($A_f = 0.01, M_f = 5$)

Sheep ($A_f = 0.02, M_f = 15$)

VOT visual object tracking

Home | Challenges | Support | Publications

VOT2016 | Home | Dataset | Participate | Program | People | ECCV2016, Amsterdam, the Netherlands

VOT2016 Challenge

News and updates

July 14th, 2016: - Workshop day

The **VOT workshop** will be held on October 10th.

You find the old news here.

Call for participation and for papers

We are happy to announce the 4th VOT Workshop, that will take place in conjunction with ECCV 2016. The event follows the three highly sucessful workshops VOT2013 (ICCV2013), VOT2014 (ECCV2014), and VOT2015 (ICCV2015).

Researchers from industry as well as academia are invited to participate. The challenge aims at **single-object short-term trackers** that do not apply pre-learned models of object appearance (**model-free**). Trackers do not necessarily need to be capable of automatic re-initialization, as the objects are visible over the whole course of the sequences.

We are also announcing the second VOT thermal imagery tracking sub-challenge VOT-TIR2016. The details of the VOT2016 and VOT-TIR2016 sub-challenge will be available soon.

The results of the VOT2016 and VOT-TIR2016 challenges will be presented at the ECCV2016 VOT workshop.

The VOT committee also solicits full-length papers describing:

VOT visual object tracking challenge

Main novelty – better ground truth.

- Each frame manually per-pixel segmented
- B-boxes automatically generated from the segmentation

VOT Results: Realtime

2013

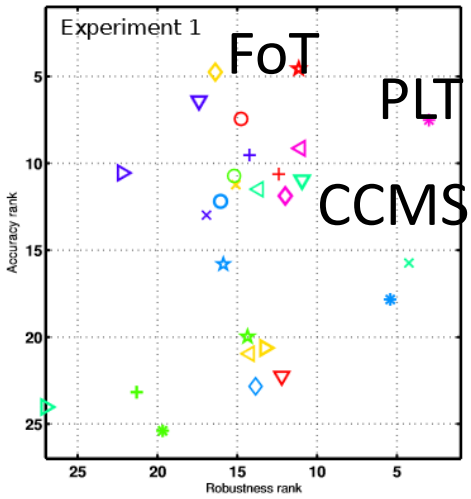
PLT (~169 fps)
FoT (~156 fps)
CCMS(~57 fps)

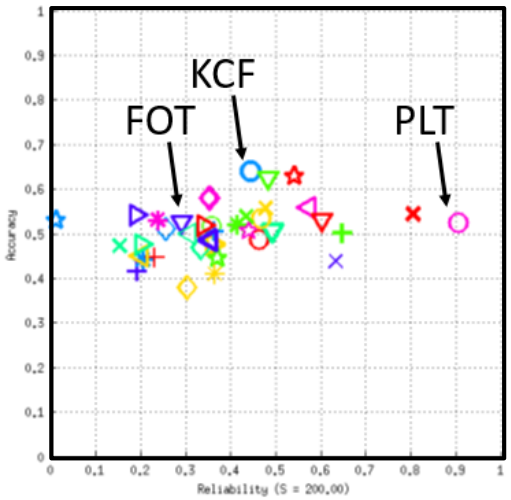
2014

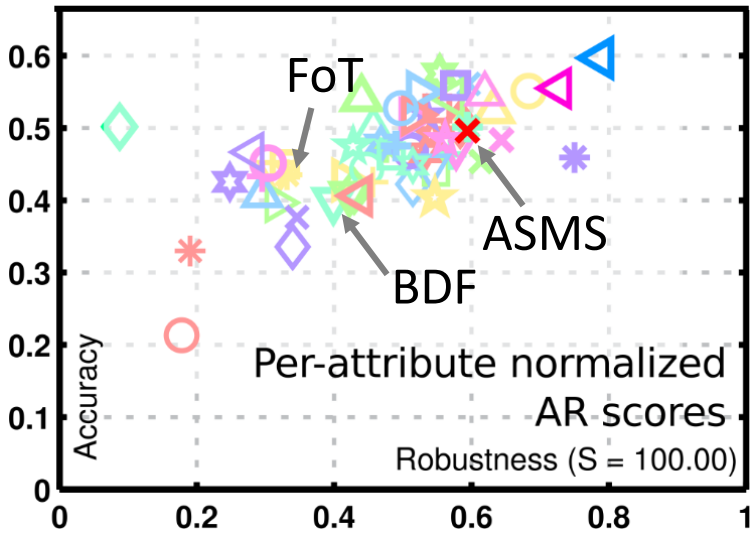
FoT (~190 fps)
PLT (~112 fps)
KCF (~36 fps)

2015

ASMS (~172 fps)
BDF (~300 fps)
FoT (~190 pfs)



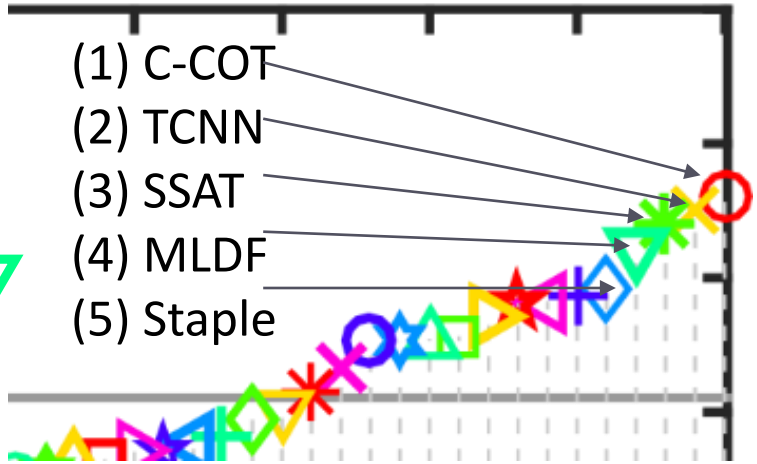




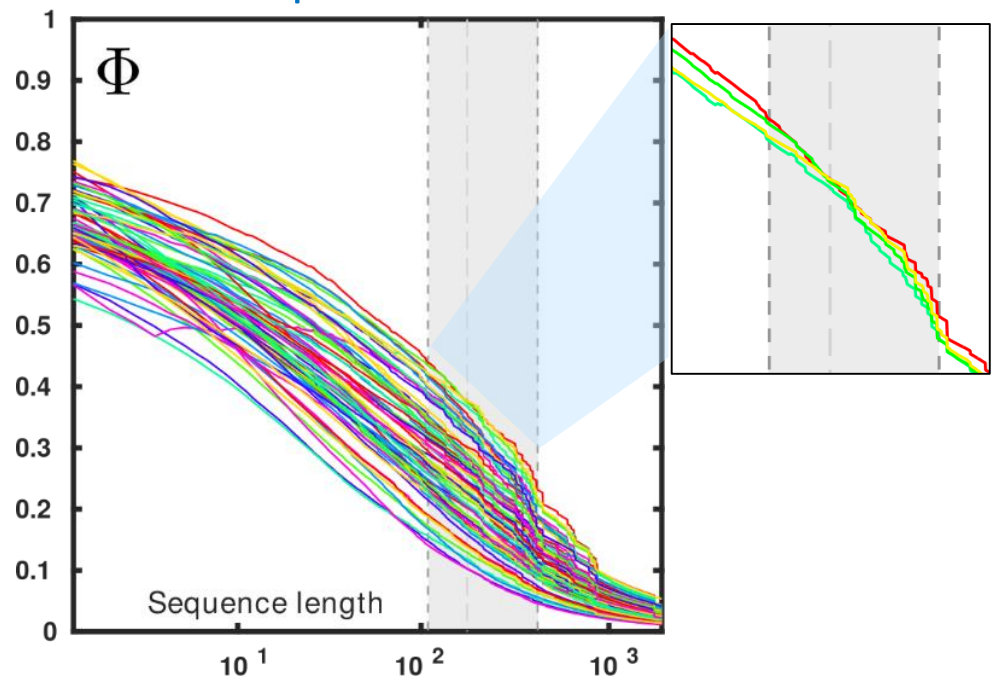
- Flow-based, Mean Shift-based, Correlation filters
- Engineering, use of basic features

VOT 2016: Results

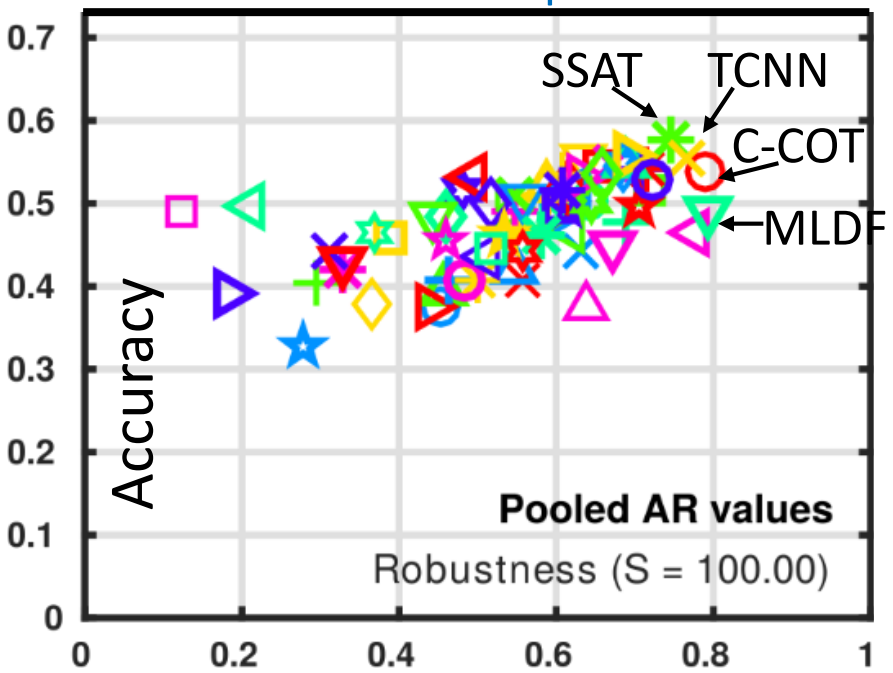
- C-COT slightly ahead of TCNN
- Most accurate: SSAT
- Most robust: C-COT and MLDF



Overlap curves



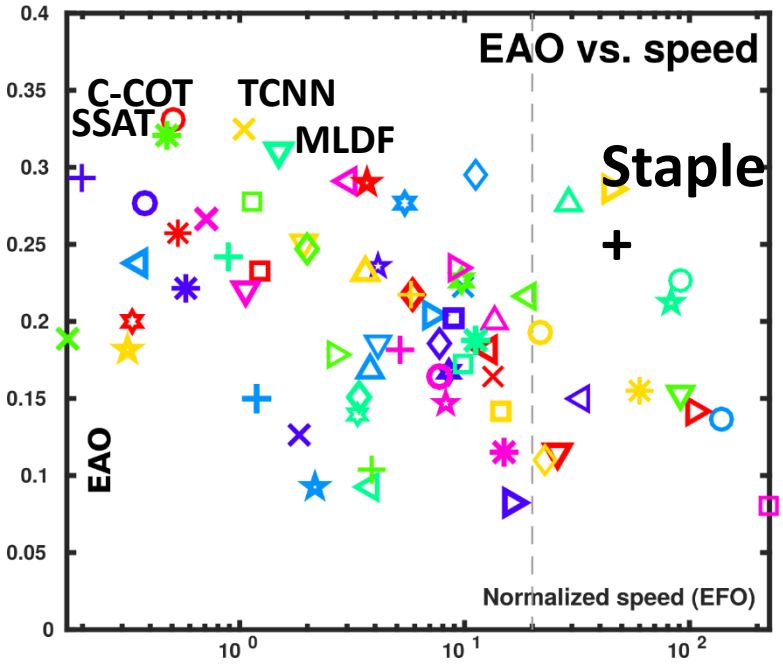
AR-raw plot

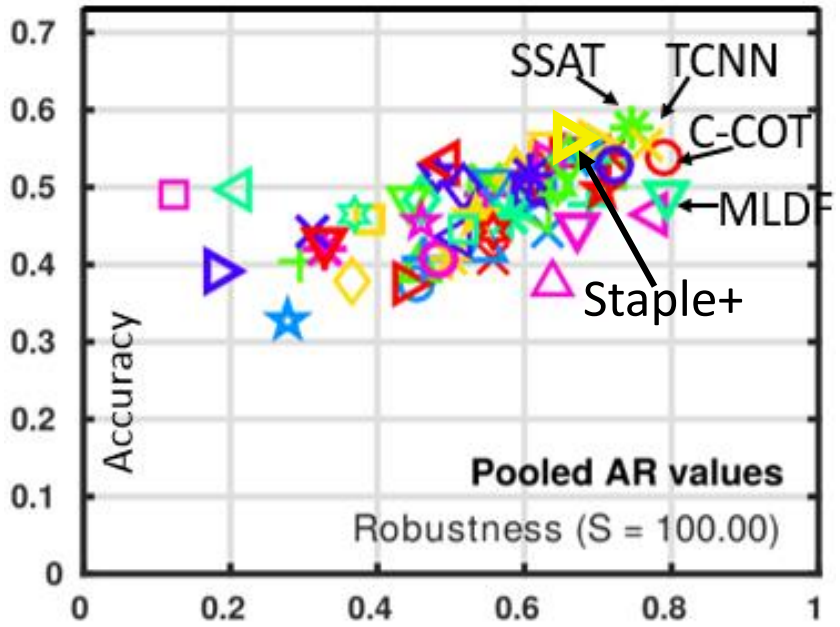


VOT 2016: Tracking speed

- Top-performers slowest
  - Plausible cause: CNN
- Real-time bound: Staple+
  - Decent accuracy,
  - Decent robustness

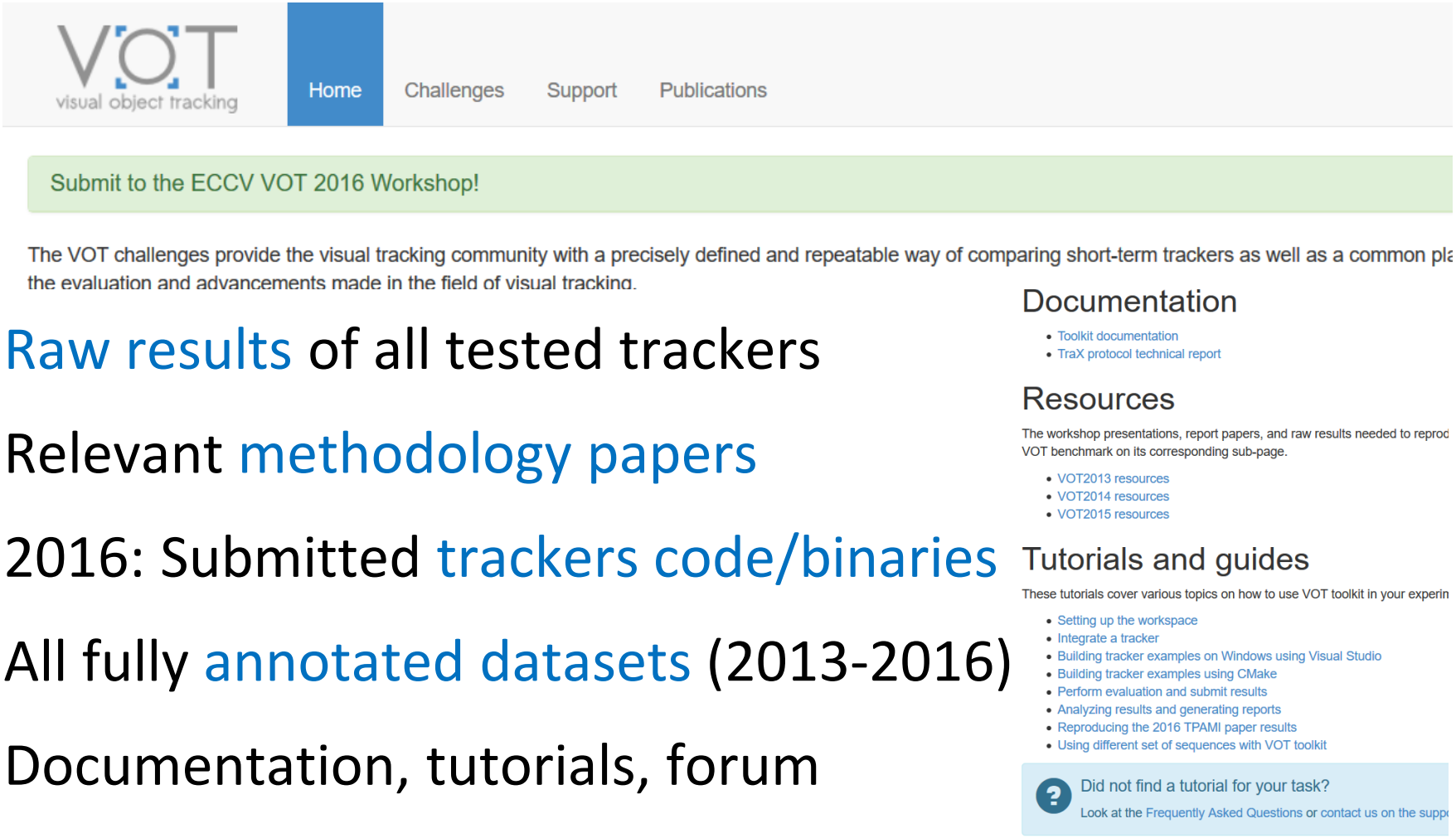
Note: the speed in some Matlab trackers has been significantly underestimated by the toolkit since it was measuring also the Matlab restart time. The EFOs of Matlab trackers are in fact higher than stated in this





VOT public resources

- Resources publicly available: VOT page



- Raw results of all tested trackers
- Relevant methodology papers
- 2016: Submitted trackers code/binaries
- All fully annotated datasets (2013-2016)
- Documentation, tutorials, forum

Summary

- “Visual Tracking” may refer to quite different problems.
- The area is just starting to be affected by CNNs.
- Robustness at all levels is the road to reliable performance
- Key components of trackers:
  - target learning (modelling, “template update”)
  - integration of detection and temporal smoothness assumptions
  - representation of the image and target
- Be careful when evaluating tracking results

What if there are several objects to track?

Secure | https://motchallenge.net

Multiple Object Tracking Benchmark

home | data | results | vis | VQA | submit | FAQ | people | login | sign up

Welcome to MOTChallenge: The Multiple Object Tracking Benchmark!

In the recent past, the computer vision community has relied on several centralized benchmarks for performance evaluation of numerous tasks including object detection, pedestrian detection, 3D reconstruction, optical flow, single-object short-term tracking, and stereo estimation. Despite potential pitfalls of such benchmarks, they have proved to be extremely helpful to advance the state-of-the-art in the respective research fields. Interestingly, there has been rather limited work on the standardization of multiple target tracking evaluation. One of the few exceptions is the well-known PETS dataset, targeted primarily at surveillance applications. Even for this widely used benchmark, a common technique for presenting tracking results to date involves using different subsets of the available data, inconsistent model training and varying evaluation scripts.

With this benchmark we would like to pave the way for a unified framework towards more meaningful quantification of multi-target tracking.

Milan et.al, 2016. MOT16: A Benchmark for Multi-Object Tracking

THANK YOU.
Questions, please?