Visual Object Tracking

Jianan Wu
Megvii (Face++) Researcher
wjn@megvii.com
Dec 2017

Applications

- From image to video:
  - Augmented Reality
  - Motion Capture
  - Surveillance
  - Sports Analysis
  - ……

Wait. What is visual tracking?

- When we talk about visual tracking, we may refer to something completely different.

- Main topics covered in this lesson:
  1. Motion estimation / optical flow
  2. Single object tracking
  3. Multiple object tracking

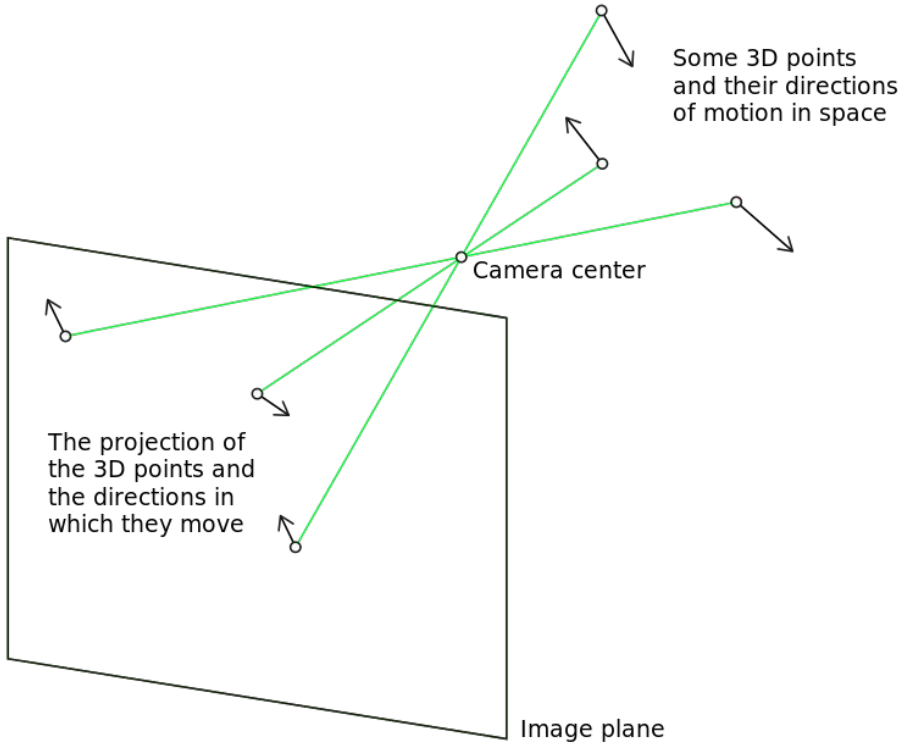
- We will also glance at other variants:
  - fast moving, multi-camera, …

Outline

1. **Motion Estimation / Optical Flow**
2. Single Object Tracking
3. Multiple Object Tracking
4. Other

Motion Field

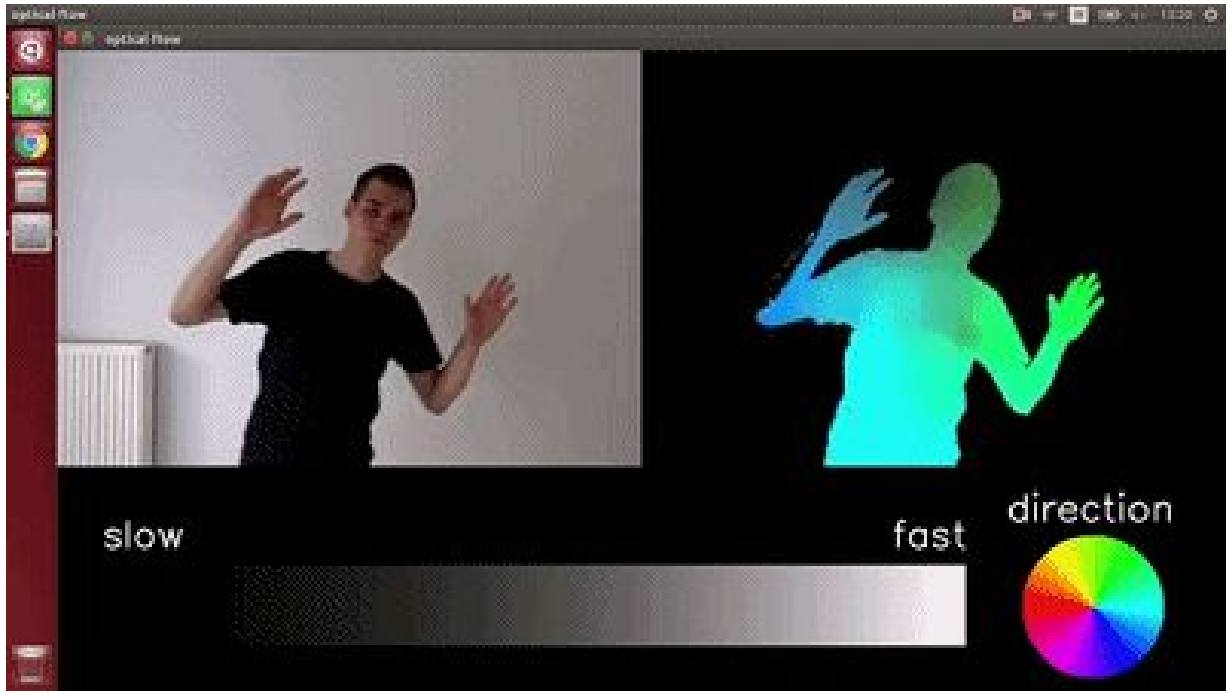
- The projection of the 3D motion onto a 2D image.
- However, the true motion field can only be approximated based on measurements on image data.



motion field (from wiki)

Optical Flow

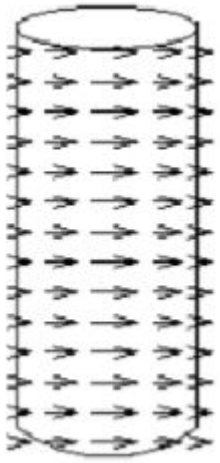
- Optical flow: the pattern of apparent motion in images.
  - Approximation of the motion field $I(x, y, t) = I(x + dx, y + dy, t + dt)$
  - Usually adjacent frames
  - Pixel level
  - Either dense or sparse



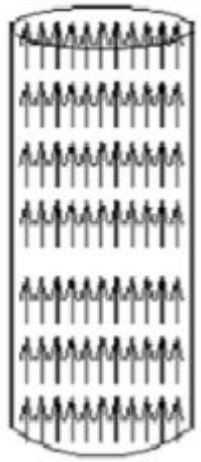
Motion Field ≈ Optical Flow

- Not always the same.

Barber's pole



Motion field



Optical flow

- Such cases are unusual. In most cases we will assume that optical flow corresponds to the motion field.

Image from: Gary Bradski slides

Kanade-Lucas-Tomasi Feature Tracker

- Steps:
  1. Find good feature points
     - E.g. Shi-Tomasi corner points
  2. Calculate optical flow
     - Lucas-Kanade method (Assume all the neighboring pixels have similar motion)

the local image flow (velocity) vector (V_x, V_y) must satisfy

$$\begin{aligned} \mathrm{I}_x(q_1)V_x + \mathrm{I}_y(q_1)V_y &= -I_t(q_1) \\ \mathrm{I}_x(q_2)V_x + \mathrm{I}_y(q_2)V_y &= -I_t(q_2) \\ &\cdots \\ \mathrm{I}_x(q_n)V_x + \mathrm{I}_y(q_n)V_y &= -I_t(q_n) \end{aligned}$$

where $q_1, q_2, \cdots, q_n$ are the pixels inside the window, and $\mathrm{I}_x(q_i), \mathrm{I}_y(q_i), \mathrm{I}_t(q_i)$ are the partial derivatives of the Image I with respect to position x, y and time t, evaluated at the current time.

  3

- Fre
- Also available in OpenCV

Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. *IJCAI.* 1981.
Carlo Tomasi and Takeo Kanade. “Detection and Tracking of Point Features”. Carnegie Mellon University Technical Report. 1991.
Jianbo Shi and Carlo Tomasi, “Good Features to Track”. *CVPR.* 1994.

Kanade-Lucas-Tomasi Feature Tracker

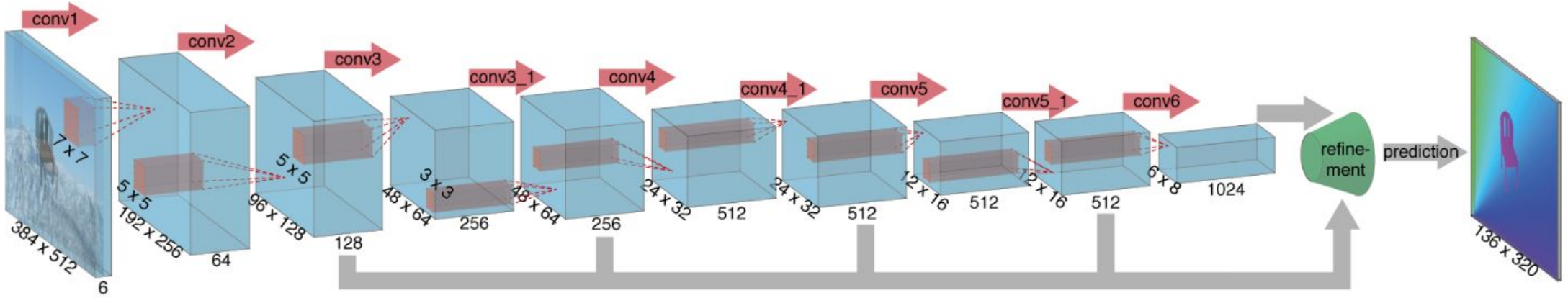
Optical Flow with CNN

- FlowNet / FlowNet 2.0
  - Learn optical flow directly from image pairs.
  - Lack of training data? Let's synthesize!
    - Flying Chairs / ChairsSDHom
    - Flying Things 3D
  - Train with simple datasets first.
  - Combine multiple FlowNets for large displacem
- https://github.com/lmb-freiburg/flownet2

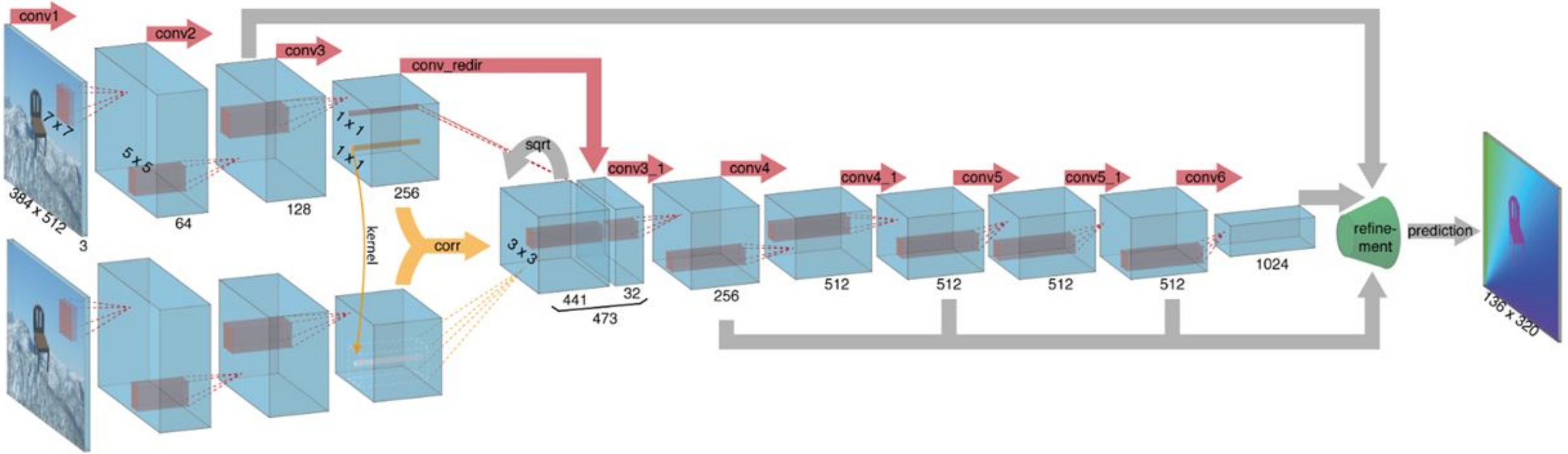
Dosovitskiy A, Fischer P, Ilg E, et al. "Flownet: Learning optical flow with convolutional network". *ICCV.* 2015.
Ilg E, Mayer N, Saikia T, et al. "Flownet 2.0: Evolution of optical flow estimation with deep networks". *CVPR.* 2017.

FlowNet: Structure

FlowNetS



FlowNetC



Optical Flow: Summary

- Establishing point to point correspondences in consecutive frames of an image sequence.

- Issues:
  - Missing concept of object
  - Large displacement handling
  - Occlusion handling
  - Failure (assumption validity) not easy to detect

Outline

1. Motion Estimation / Optical Flow
2. **Single Object Tracking**
3. Multiple Object Tracking
4. Other

Single Object Tracking

- Single object, single camera
- Model free:
  - Nothing but a single training example is provided by the bounding box in the first frame
- Short term:
  - Tracker does not perform re-detection
  - Fail if tracking drifts off the target
- Subject to Causality:
  - Tracker does not use any future frames

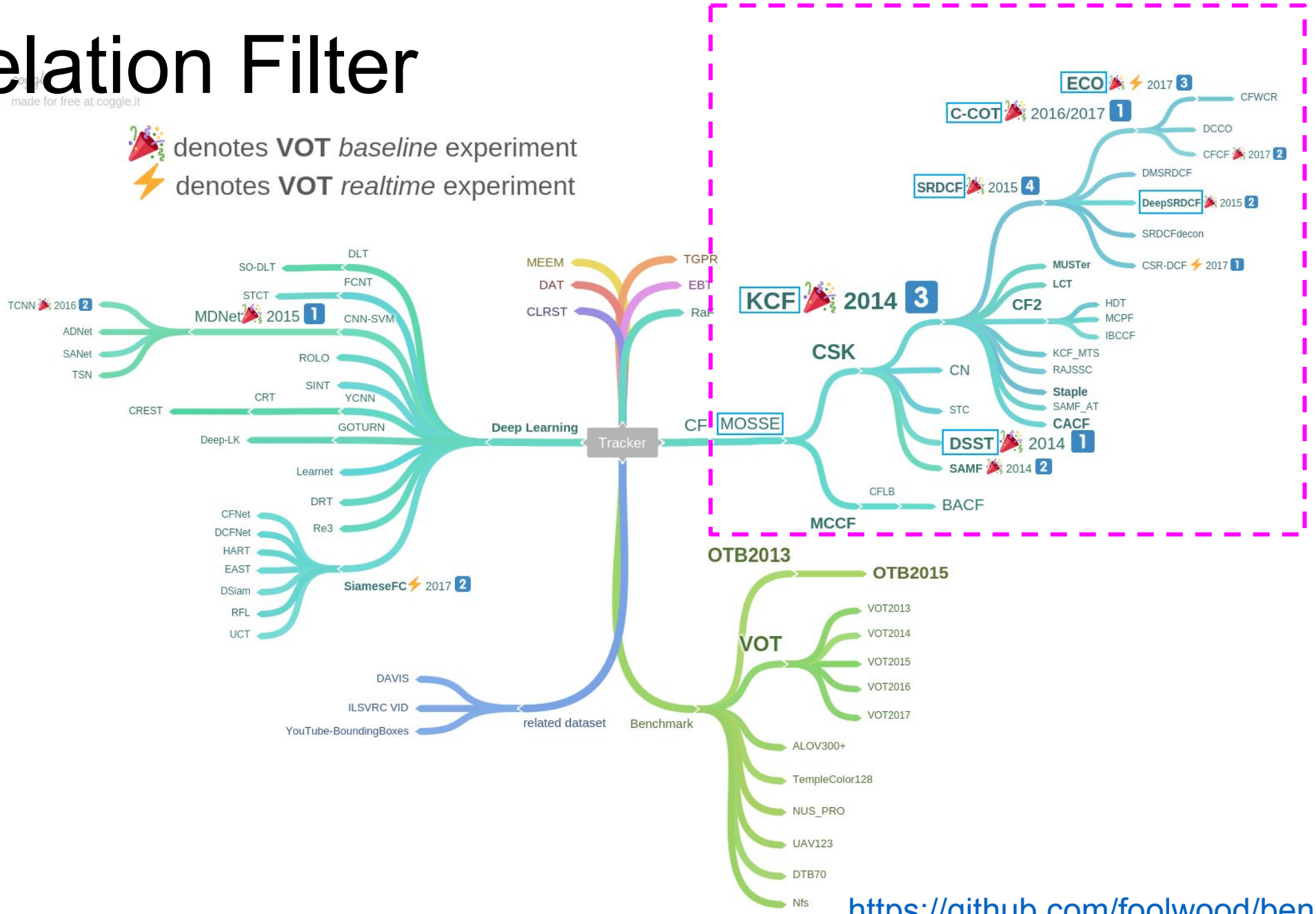
Single Object Tracking

- Protocol:

  Setup tracker
  Read initial object region and first image
  Initialize tracker with provided region and image
  loop
    Read next image
    if image is empty **then**
      Break the tracking loop
    end if
    Update tracker with provided image
    Write region to file
  end loop
  Cleanup tracker

Luka Čehovin, TraX. “The visual Tracking eXchange Protocol and Library”. *Neurocomputing*. 2017

Correlation Filter



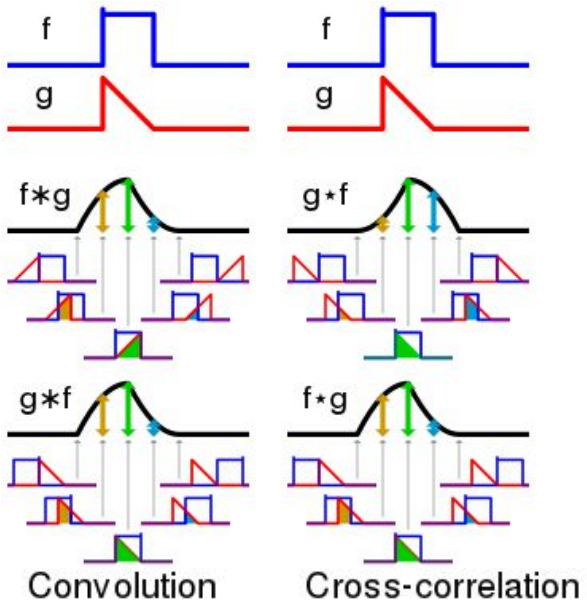
https://github.com/foolwood/benchmark_results

Correlation Filter

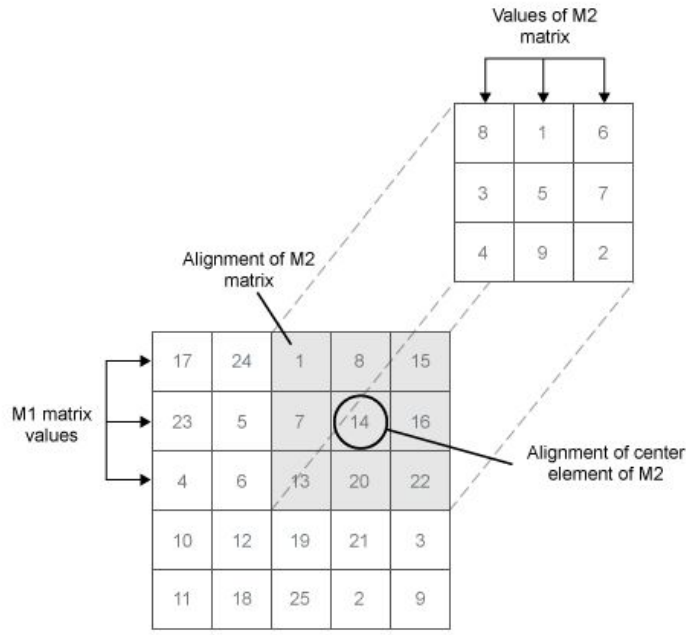
- Cross-correlation:
  - Cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other

$$(f \star g)(\tau) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f^*(t)\, g(t+\tau)\, dt$$

  - Similar to convolution



2D cross-correlation



$$1\times 8 + 7\times 3 + 13\times 4 + 8\times 1 + 14\times 5 + 20\times 9 + 15\times 6 + 16\times 7 + 22\times 2 = 585.$$

Convolution Theorem

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

$$g = f \star h \Leftrightarrow \mathcal{F}(g) = \mathcal{F}^*(f) \odot \mathcal{F}(h)$$

where $\mathcal{F}(g)$ is the Discrete Fourier Transform (DFT) of g (likewise for f and h)

$\star$ is cross-correlation and $\odot$ means element-wise product

* is complex-conjugate

- Computation is much faster in Fourier Domain: O(P^2) -> O(PlogP), P is number of pixels in tracking window.

Minimum Output Sum of Squared Error Filter

- Initialization: learn filter by applying random small affine perturbations to the tracking window for the first frame of the video

$$\min_{H^*} \sum_i |F_i \odot H^* - G_i|^2$$

where F_i is training image (gray scale, weighted by cosine window), G_i is generated from ground truth: a compact 2D Gaussian shaped peak centered on the target in training image F_i.

- Closed form solution:

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*}$$

David S. Bolme et al. “Visual Object Tracking using Adaptive Correlation Filters”. *CVPR*. 2010

Minimum Output Sum of Squared Error Filter

- Filter Update:

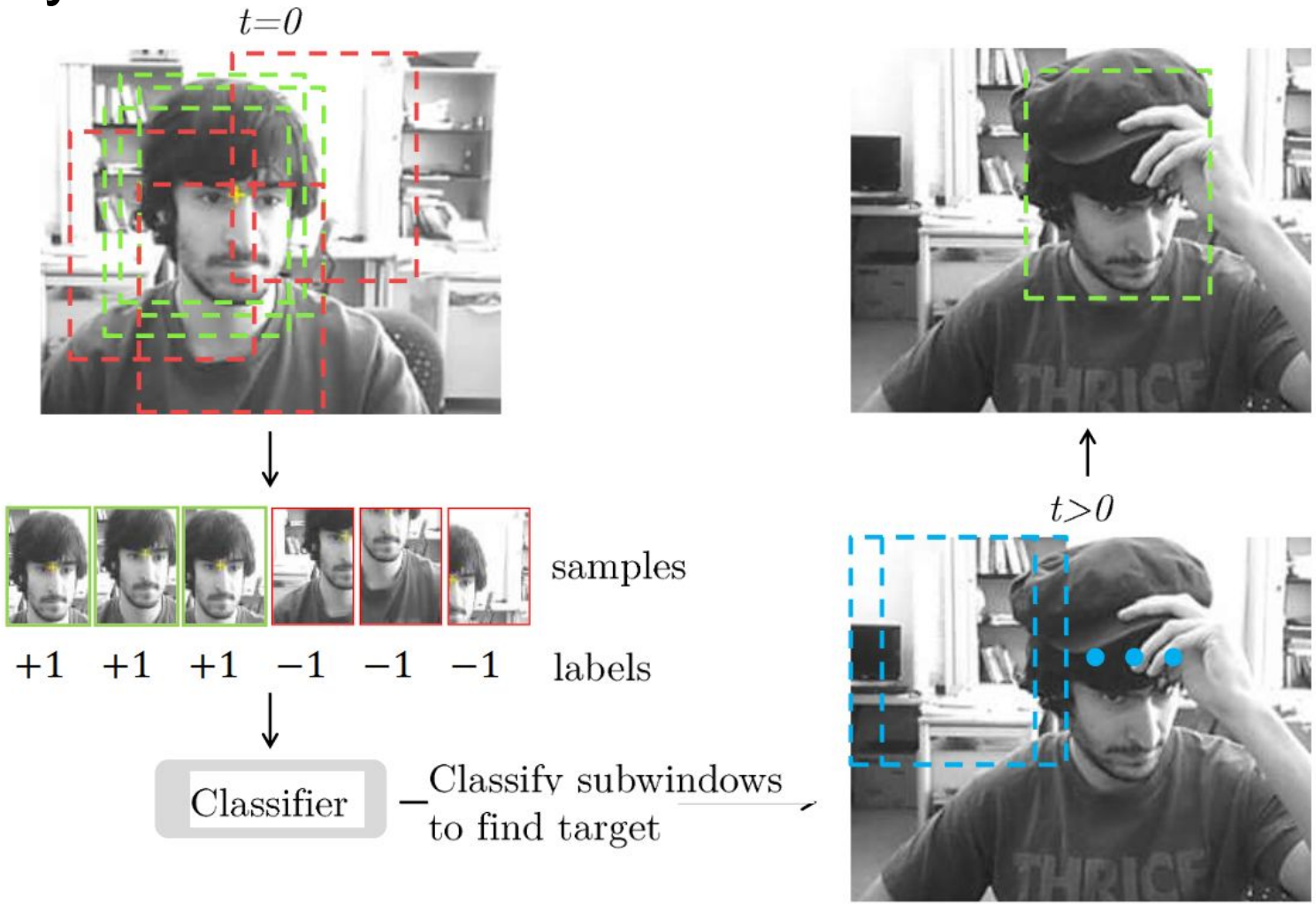
$$
\begin{aligned}
H_i^* &= \frac{A_i}{B_i} \\
A_i &= \eta G_i \odot F_i^* + (1-\eta) A_{i-1} \\
B_i &= \eta F_i \odot F_i^* + (1-\eta) B_{i-1}
\end{aligned}
$$

where η is learning rate

- Simple
- Extremely Fast! 600+ fps

Discriminative Tracking

- Tracking by Detection



Kernelized Correlation Filter

- Circulant matrices

$$X = C(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix}.$$

- Diagonalization:

$$X = F\,\mathrm{diag}\,(\hat{\mathbf{x}})\;F^H$$

- Where F is a constant matrix (DFT matrix) that does not depend on x, and $\hat{x} = \mathcal{F}(x)$

João F. Henriques, Rui Caseiro, Pedro Martins, Jorge Batista. “Kernelized Correlation Filters”. *TPAMI* . 2015

Kernelized Correlation Filter

- Ridge regression as classifier:

$$\min_{\mathbf{w}} \sum_i (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|^2 \qquad \Rightarrow \qquad \mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

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

- The regression target of the additional samples follow a Gaussian function, which takes a value of 1 for a centered target and smoothly decays to 0 for any other shifts, according to a bandwidth s.

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}$$

- Exactly the MOSSE solution !
- Good learning algorithm with lots of data.

Kernelized Correlation Filter

- Circulant matrices are very general tool which allows to replace standard operations with fast Fourier operations.
- Kernel Ridge Regression as classifier:
  - with K kernel matrix $K_{ij} = \kappa(x_i, x_j)$ and dual space representation

$$\mathbf{w} = \sum_i \alpha_i \varphi(\mathbf{x}_i) \qquad \varphi^T(\mathbf{x})\varphi(\mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}') \qquad K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

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

  - For many kernels, circulant data ⇒ circulant K matrix

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

  - k^{xx} is the first row of the kernel matrix $\Rightarrow \hat{\boldsymbol{\alpha}} = \dfrac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}$

Kernelized Correlation Filter

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

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

For Gaussian kernel it yields:

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

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

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

$$K^{\mathbf{z}} = C(\mathbf{k}^{\mathbf{xz}})$$

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

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

Kernel allows integration of more complex and multi-channel features

slides material by João F. Henriques

Kernelized Correlation Filter

very few hyperparameters

code fits on one slide!

Use HoG features (32 channels)

~300 FPS

Open-Source (Matlab/Python/Java/C)

Algorithm 1 : Matlab code, with a Gaussian kernel.
Multiple channels (third dimension of image patches) are supported. It is possible to further reduce the number of FFT calls.
Implementation with GUI available at:
http://www.isr.uc.pt/~henriques/

Inputs
- x: training image patch, $m \times n \times c$
- y: regression target, Gaussian-shaped, $m \times n$
- z: test image patch, $m \times n \times c$

Output
- responses: detection score for each location, $m \times n$

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

function responses = detect(alphaf, x, z, sigma)
  k = kernel_correlation(z, x, sigma);
  responses = 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
```

slides material by João F. Henriques

From KCF to Discriminative CF Trackers

- Martin Danelljan et al. – DSST
  - PCA-HoG + grayscale pixels features
  - Filters for translation and for scale (in the scale-space pyramid)

$$\varepsilon = \left\| g - \sum_{l=1}^{d} h^l \star f^l \right\|^2 + \lambda \sum_{l=1}^{d} \|h^l\|^2$$

- Li et al. – SAMF
  - HoG, color-naming(CN) and grayscale pixels features
  - Quantize scale space and normalize each scale to one size by bilinear inter.
- Martin Danelljan et al. – SRDCF
  - Spatial regularization in the learning process
    - limits boundary effect
    - penalize filter coefficients depending on their spatial location
  - Allow to use much larger search region
  - More discriminative to background (more training data)

(a) Original image. (b) Periodicity in correlation filters.

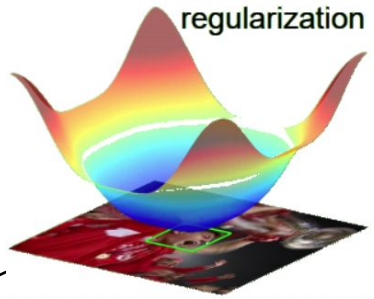
- Martin Danelljan et al. – Deep SRDCF
  - CNN features

$$S_f(x) = \sum_{l=1}^{d} x^l * f^l.$$

Sample weights

$$\varepsilon(f) = \sum_{k=1}^{t} \alpha_k \left\| S_f(x_k) - y_k \right\|^2 + \sum_{l=1}^{d} \left\| w \cdot f^l \right\|^2$$

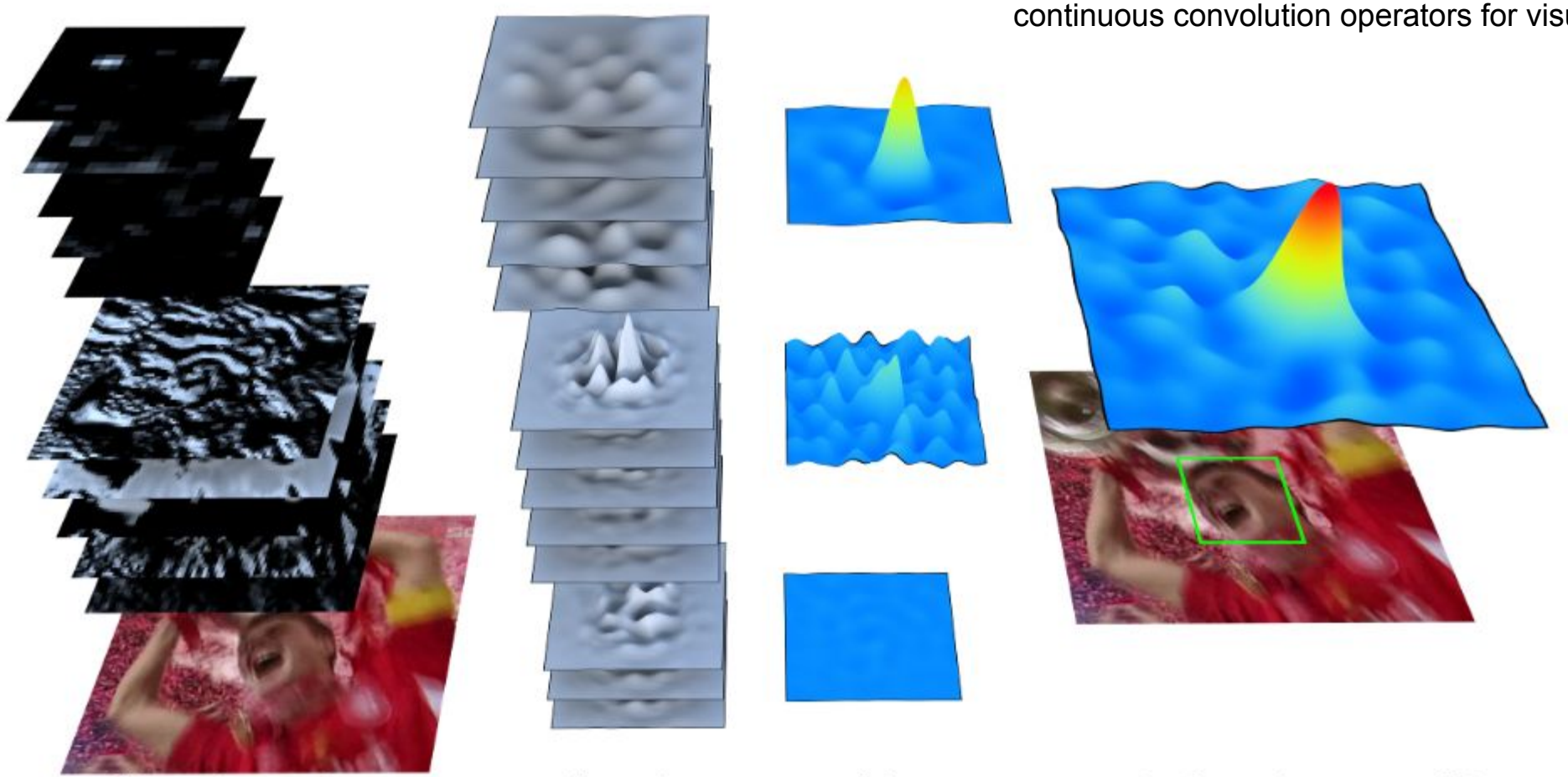
Spatial regularization



Continuous-Convolution Operator Tracker

- Multi-resolution CNN features

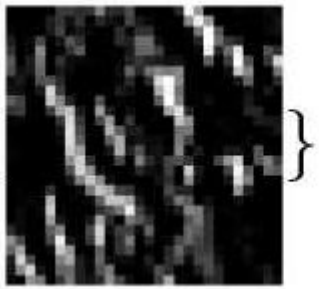
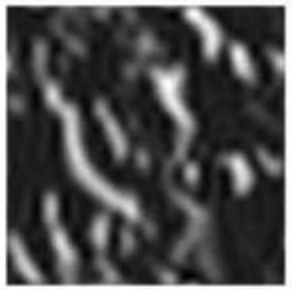
Danelljan, Martin, et al. "Beyond correlation filters: Learning continuous convolution operators for visual tracking." *ECCV*, 2016.



Continuous-Convolution Operator Tracker

$$E(f) = \sum_{j=1}^{m} \alpha_j \left\| S_f\{x_j\} - y_j \right\|^2 + \sum_{d=1}^{D} \left\| w f^d \right\|^2$$

$$S_f\{x\} = \sum_{d=1}^{D} f^d * J_d\{x^d\}, \quad x \in \mathcal{X}$$

- Interpolation operator
- Optimized $J_d\{x^d\}(t) = \sum_{n=0}^{N_d-1} x^d[n] b_d \left(t - \frac{T}{N_d} n \right)$ ı conju $J_d\{$  $\} =$ 
- Implementation: https://github.com/martin-danelljan/Continuous-ConvOp
- Very Slow, ~ 1fps
- A lot of parameters, easy to overfitting

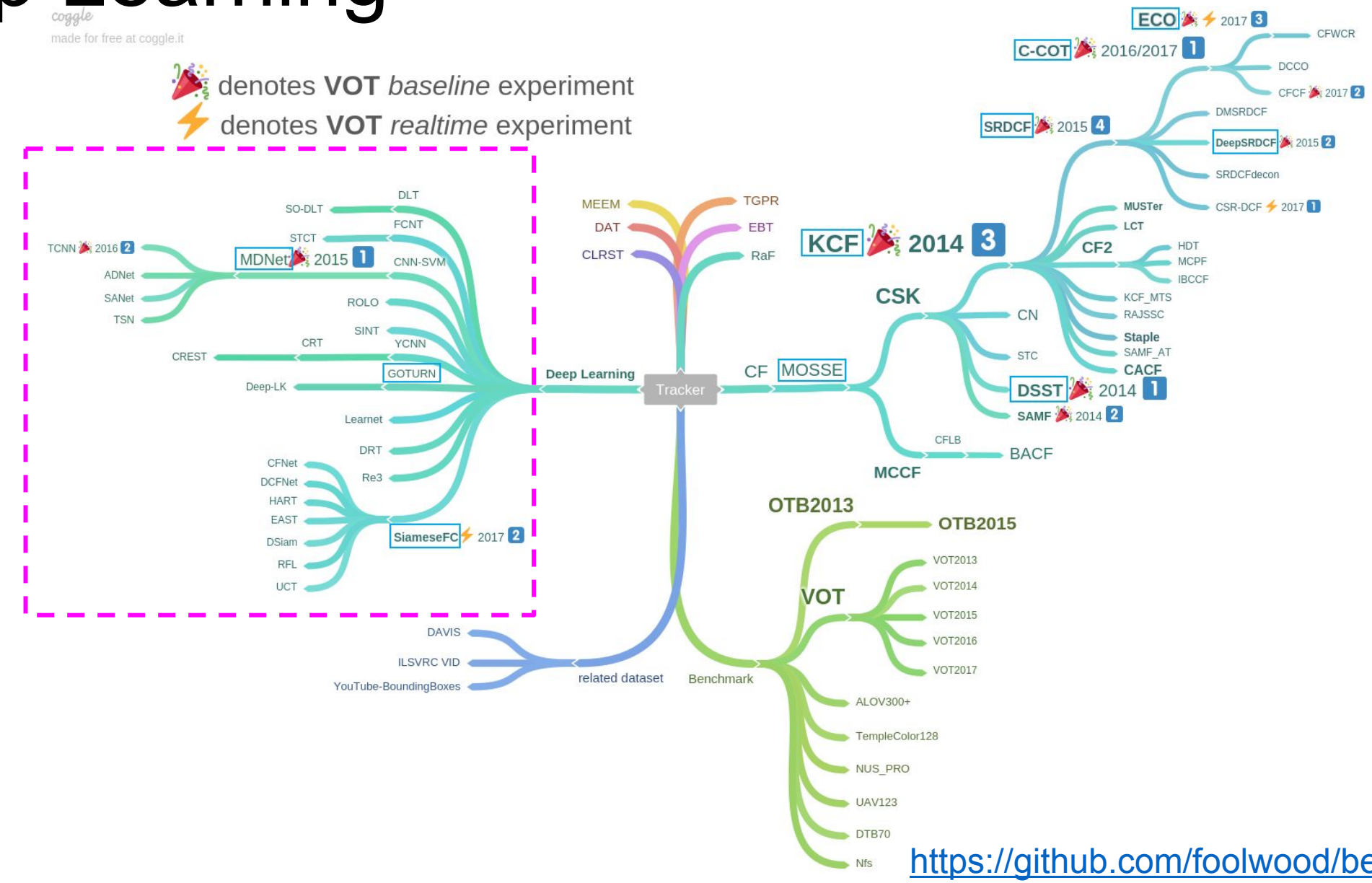
Efficient Convolution Operators

- Based on C-COT
- Main Improvements:
  1. Introduce a factorized convolution operator that dramatically reduces the number of parameters in the DCF model.
  2. A Gaussian mixture model to reduce the number of samples in the learning, while maintaining their diversity.
  3. Only optimize every N frames for faster tracking.

- Implementation: https://github.com/martin-danelljan/l
- ~ 15 FPS on GPU

Danelljan, Martin, et al. "ECO: Efficient Convolution Operators for Tracking." *CVPR*. 2017

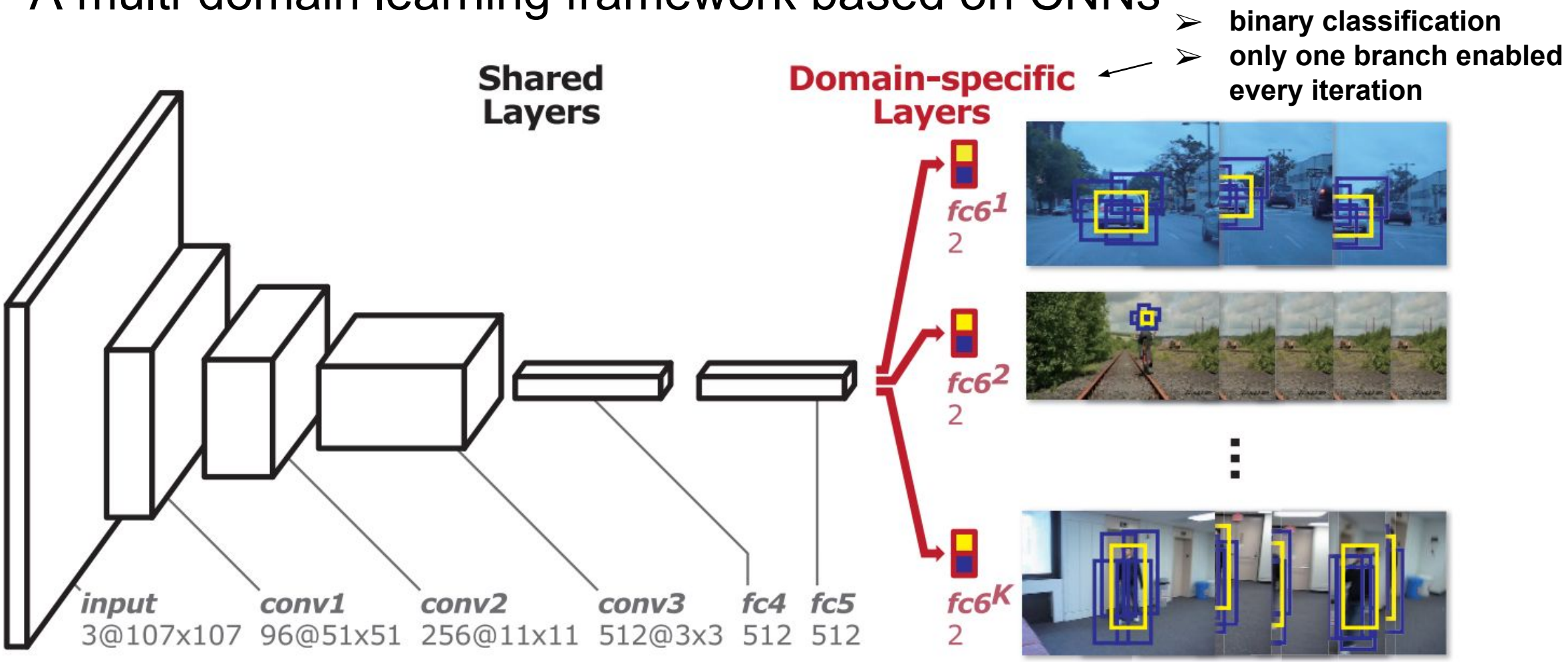
Deep Learning



https://github.com/foolwood/benchmark_results

Multi-Domain Convolutional Neural Network Tracker

- A multi-domain learning framework based on CNNs



Hyeonseob Nam, Bohyung Han. “Learning Multi-Domain Convolutional Neural Networks for Visual Tracking”. *CVPR*. 2016

Multi-Domain Convolutional Neural Network Tracker

- Online tracking:
  - Replace fc1-fc6 to a single branch with random initialization
  - Sample positive (iou>0.7) and negative (iou<0.5) samples for online training
  - Multi scale target candidate samples from Gaussian
- Hard minibatch mining
- Bounding box regression
- ~ 1 fps
- https://github.com/HyeonseobNam/MDNet

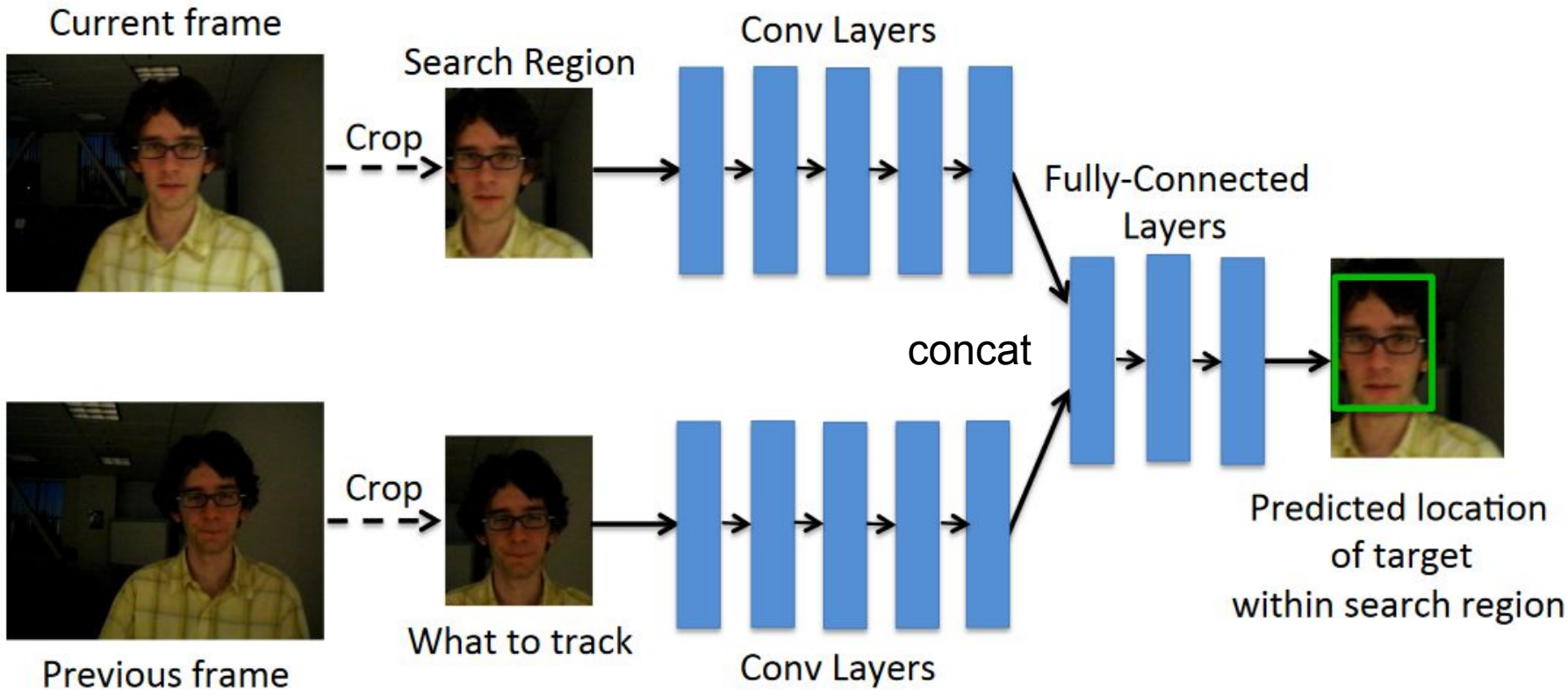
Algorithm 1 Online tracking algorithm

Input : Pretrained CNN filters $\{\mathbf{w}_1, \ldots, \mathbf{w}_5\}$
Initial target state $\mathbf{x}_1$
Output: Estimated target states $\mathbf{x}_t^*$

Randomly initialize the last layer $\mathbf{w}_6$.
Train a bounding box regression model.
Draw positive samples S_1^+ and negative samples S_1^-.
Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using S_1^+ and S_1^-;
$\mathcal{T}_s \leftarrow \{1\}$ and $\mathcal{T}_l \leftarrow \{1\}$.
repeat
  Draw target candidate samples $\mathbf{x}_t^i$.
  Find the optimal target state $\mathbf{x}_t^*$ by Eq. (1).
  if $f^+(\mathbf{x}_t^*) > 0.5$ **then**
    Draw training samples S_t^+ and S_t^-.
    $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{t\}$, $\mathcal{T}_l \leftarrow \mathcal{T}_l \cup \{t\}$.
    if $|\mathcal{T}_s| > \tau_s$ **then** $\mathcal{T}_s \leftarrow \mathcal{T}_s \setminus \{\min_{v \in \mathcal{T}_s} v\}$.
    if $|\mathcal{T}_l| > \tau_l$ **then** $\mathcal{T}_l \leftarrow \mathcal{T}_l \setminus \{\min_{v \in \mathcal{T}_l} v\}$.
    Adjust $\mathbf{x}_t^*$ using bounding box regression.
  if $f^+(\mathbf{x}_t^*) < 0.5$ **then**
    Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S^+_{v \in \mathcal{T}_s}$ and $S^-_{v \in \mathcal{T}_s}$.
  else if $t \bmod 10 = 0$ **then**
    Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S^+_{v \in \mathcal{T}_l}$ and $S^-_{v \in \mathcal{T}_s}$.
until end of sequence

GOTURN

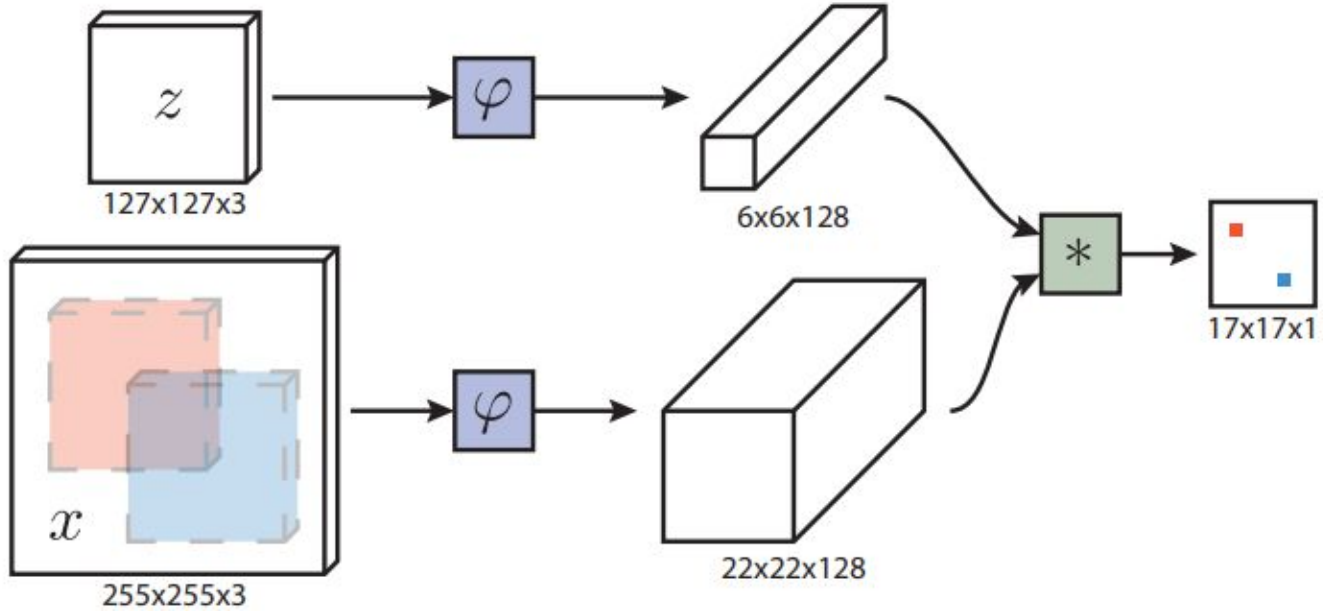
- Simple and no online model update
- http://davheld.github.io/GOTURN/GOTURN.html
- ~ 100 fps



Held, David, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." *ECCV*. 2016.

SiameseFC

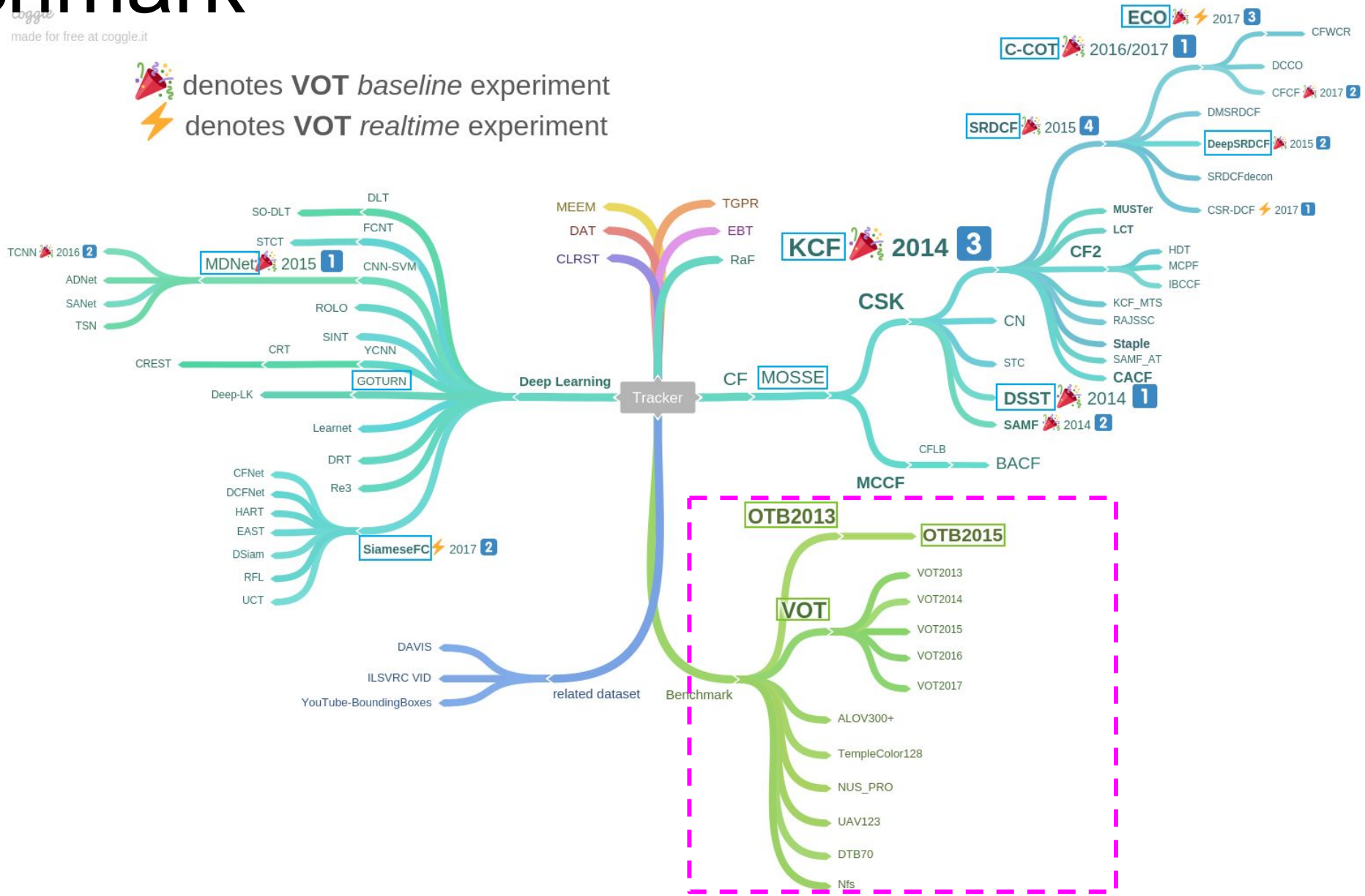
- A deep FCN is trained to address a more general similarity learning problem in an initial offline phase
- Training from ImageNet Video dataset
  - >> online learning methods
- No online model update
- https://github.com/bertinetto/siamese-fc
- ~ 60 fps



Bertinetto, Luca, et al. "Fully-convolutional siamese networks for object tracking." *ECCV*. 2016.

SiameseFC

Benchmark

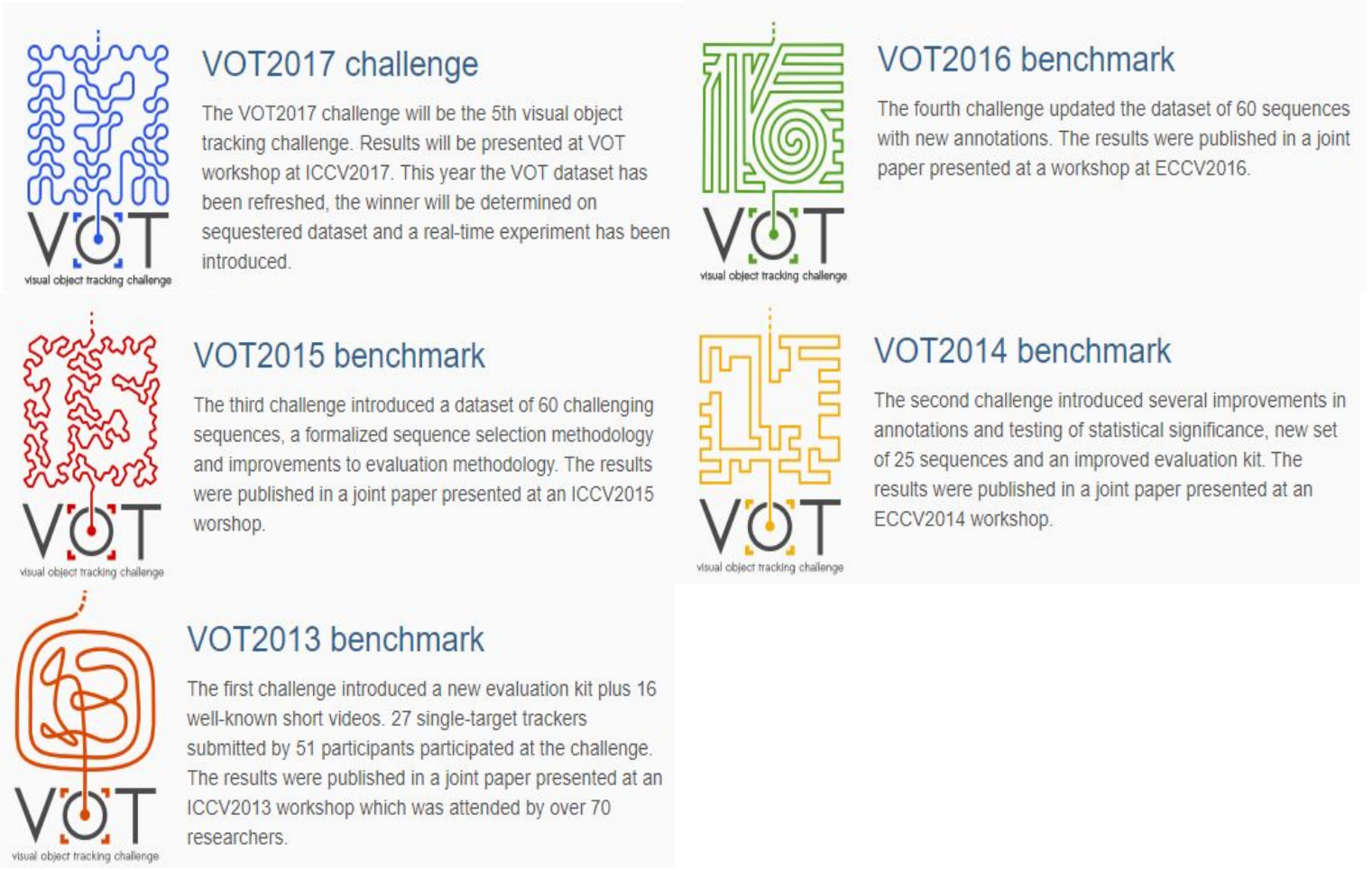


https://github.com/foolwood/benchmark_results

Benchmark: VOT

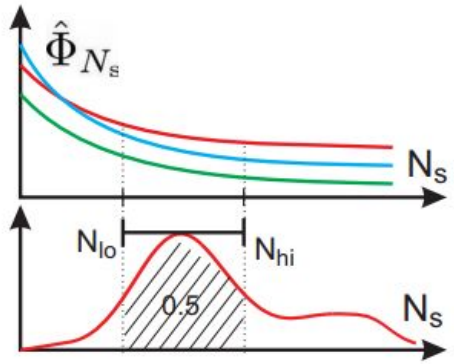
- http://www.votchallenge.net/index.html

- VOT 2017:
  - 60 sequences (50 from VOT 2016 and 10 new)
  - An additional sequestered dataset for top trackers.



Evaluation Metrics: VOT

- Accuracy:
  - Average overlap during successful tracking
- Robustness:
  - Number of times a tracker drifts off the target
- Expected Average Overlap(EAO):

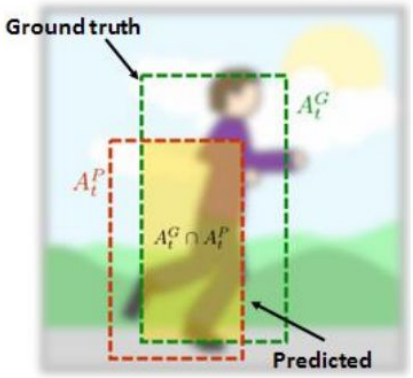


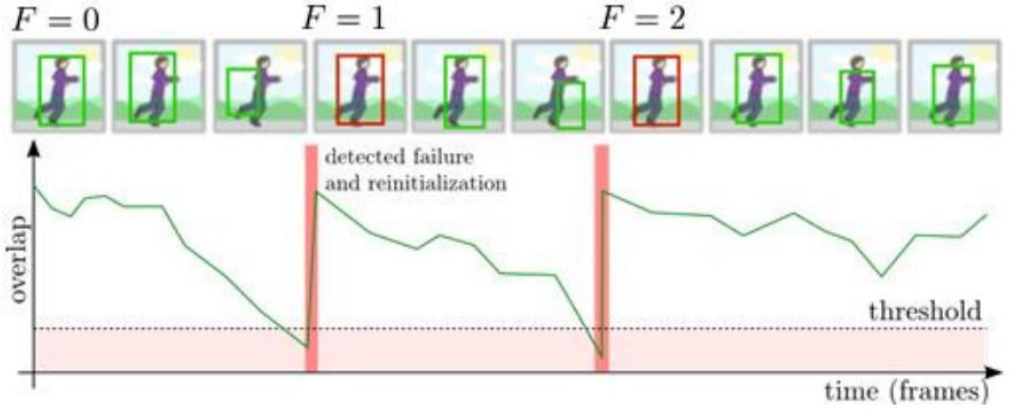
Φ_i : average of per-frame overlaps

$$\Phi_{N_s} = \frac{1}{N_s} \sum_{i=1:N_s} \Phi_i$$

$$\hat{\Phi}_{N_s} = \langle \Phi_{N_s} \rangle$$

$$\hat{\Phi} = \frac{1}{N_{hi} - N_{lo}} \sum_{N_s=N_{lo}:N_{hi}} \hat{\Phi}_{N_s}$$

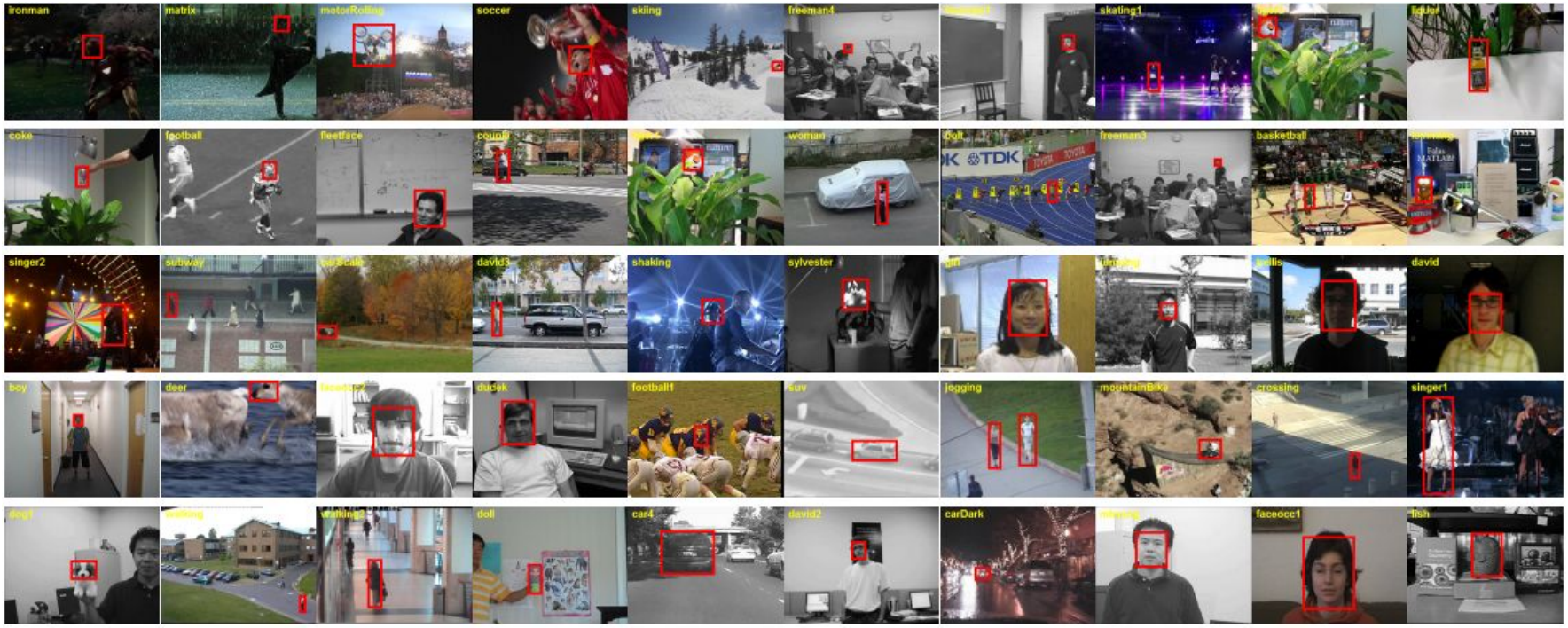




Čehovin, Luka, Aleš Leonardis, and Matej Kristan. "Visual object tracking performance measures revisited." *IEEETIPI* 25.3 (2016): 1261-1274.
Kristan, Matej, et al. "A novel performance evaluation methodology for single-target trackers." *IEEE TPAMI* 38.11 (2016): 2137-2155.

Benchmark: OTB

- OTB:
  - OTB2013
  - TB-100, OTB100, OTB2015
    - TB-50, OTB50: 50 difficult sequences among TB-100
  - http://cvlab.hanyang.ac.kr/tracker_benchmark/index.html

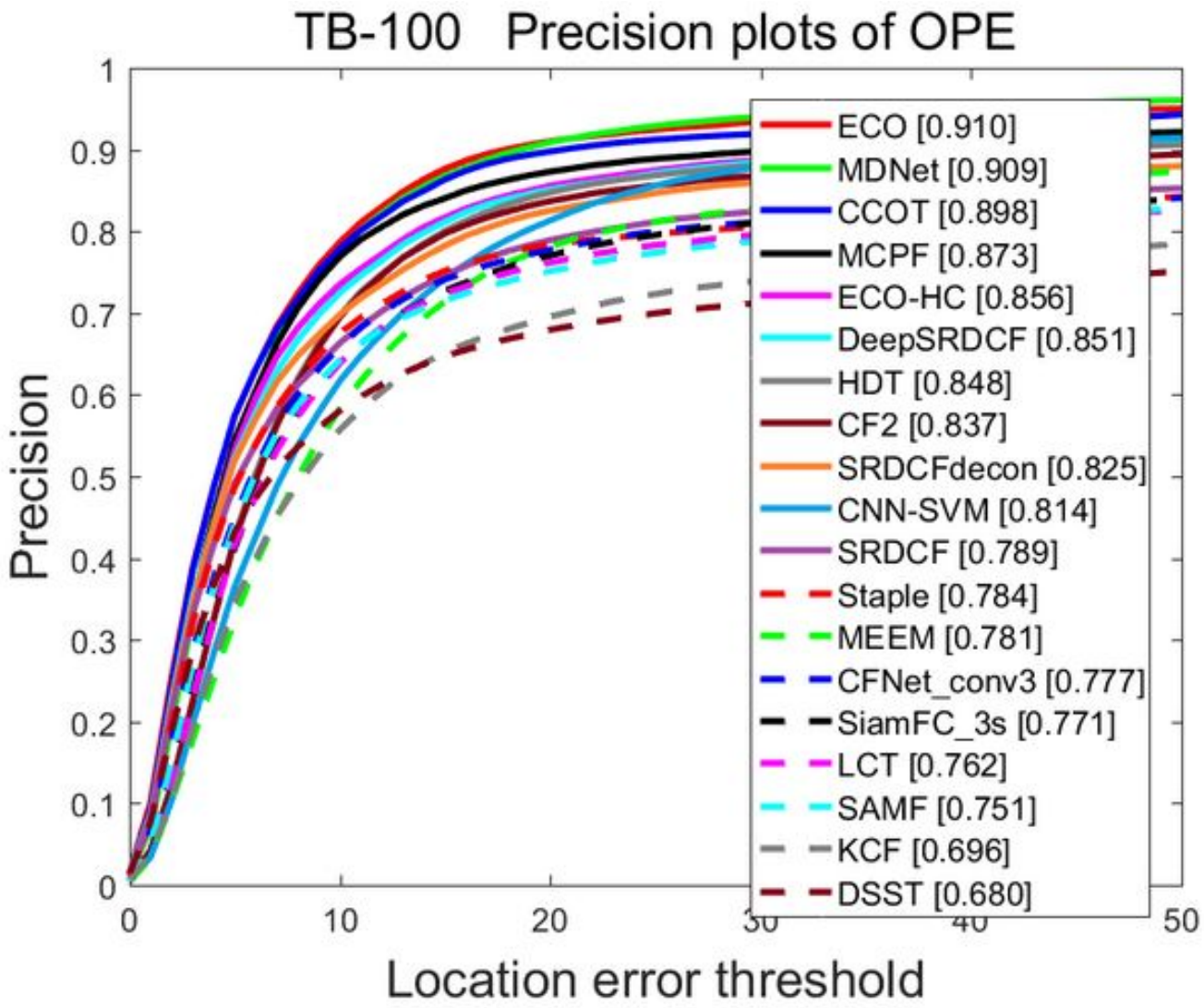


Evaluation Metrics: OTB

- One Pass Evaluation (OPE):
  - Run tracker throughout a test sequence initialized by ground truth bounding box in the first frame and return the average precision.
- Spatial Robustness Evaluation(SRE):
  - Run tracker throughout a test sequence with initialization from 12 **different** bounding boxes by shifting or scaling ground truth in the first frame and return the average precision.

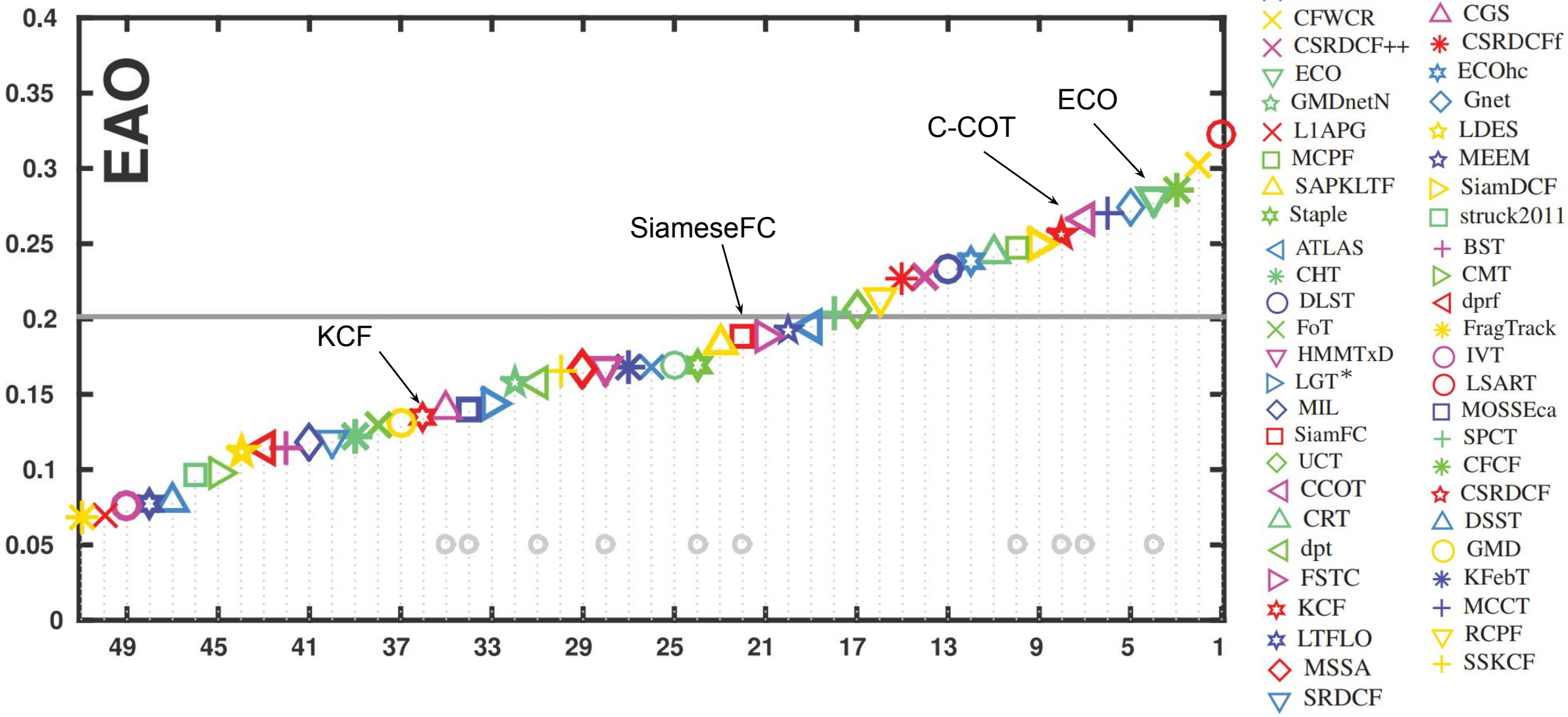
Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark." *CVPR*. 2013.

Results of TB-100



https://github.com/foolwood/benchmark_results

Results of VOT2017



http://openaccess.thecvf.com/content_ICCV_2017_workshops/papers/w28/Kristan_The_Visual_Object_ICCV_2017_paper.pdf

Outline

1. Motion Estimation / Optical Flow
2. Single Object Tracking
3. **Multiple Object Tracking**
4. Other

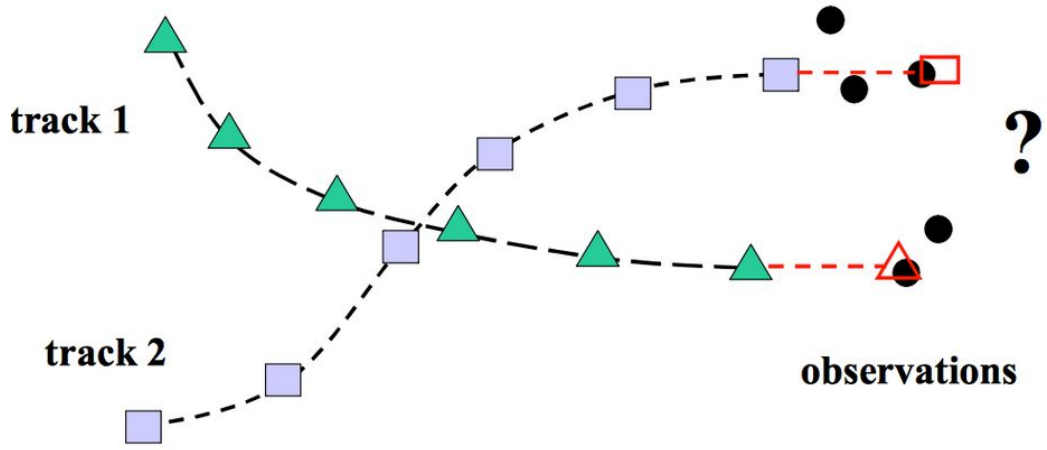
Multiple Object Tracking

- For each frame in a video, localize and identify all objects of interest, so that the identities are consistent throughout the video.
- Compared to single object tracking:
  - Target is not given in the first frame.
  - Classes of targets are known and models are always trained offline.
  - Long term: detection can be done whenever necessary.
  - Online and offline tracking are both available.
  - The number of objects is unknown.
  - The number of objects may change.
- Example:

  tracking all the persons in the video

Tracking by Detection

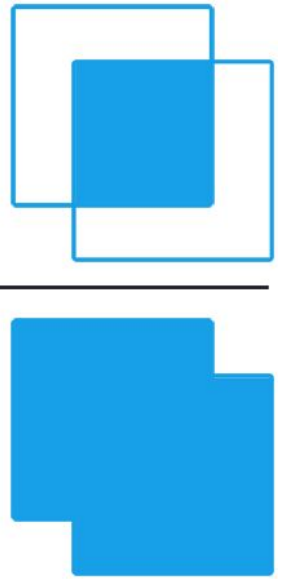
- For each frame, first localize all objects using an object detector
- Associate detected objects between frames
- Make multiple object tracking to be a association problem more than a tracking problem.
- Association based on location, motion, appearance and so on.



Location

- Intersection over union (IOU) :

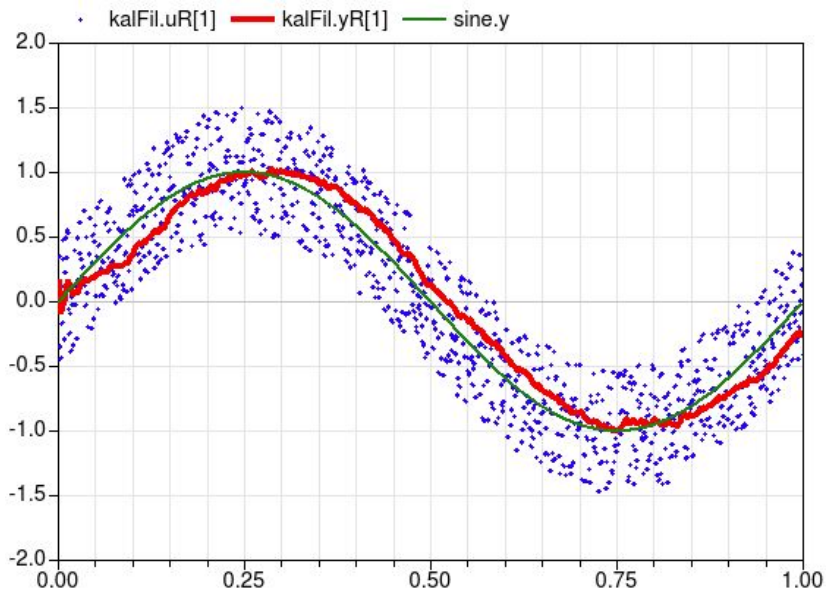
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



  - Problem: lack of discriminability if iou == 0
  - Sometimes we use intersection over minimum (IOM)
- L1/L2 distance
  - Problem: related to object's shape and camera's parameters.
  - Better to convert into world coordinate if possible.

Motion

- Modeling the movement of objects.
- Kalman filter:
  - Using Kalman filter is a way of optimally estimating the state of a linear dynamical system.
  - A possible state space: center position (*x, y*), aspect ratio *a*, height *h* and their respective velocities of the bounding box.
  - Use detection result as observation.



Appearance

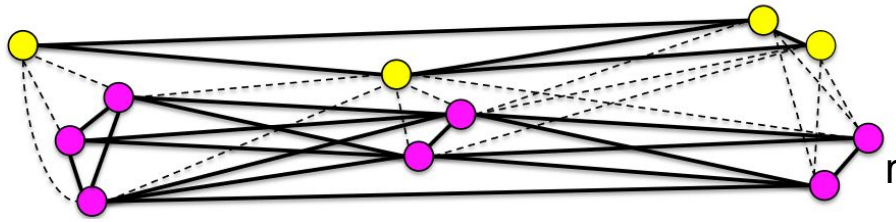
- Techniques in single object tracking like cross correlation and SiameseFC can be used here
- Hand-crafted features like histograms and color names
- CNN features
  - For pedestrian tracking, we can use reid feature

Association

- Location, motion and appearance features need to be combined.
  - Different weights in different applications
- Three kinds of assignments
  - Detection – Detection
  - Trajectory – Detection
  - Trajectory – Trajectory
- Do not trust the detector!
  - FP and FN of the detector make the association even more difficult.
- Tune your tracker according to your detector.
  - It is good for association to understand the mistakes your detector often makes.

Association as Optimization

- Local method:
  - Hungarian algorithm (Kuhn-Munkres algorithm)
- Global methods:
  - Clustering
  - Network flow
  - Minimum cost multi-cut problem
  - ……

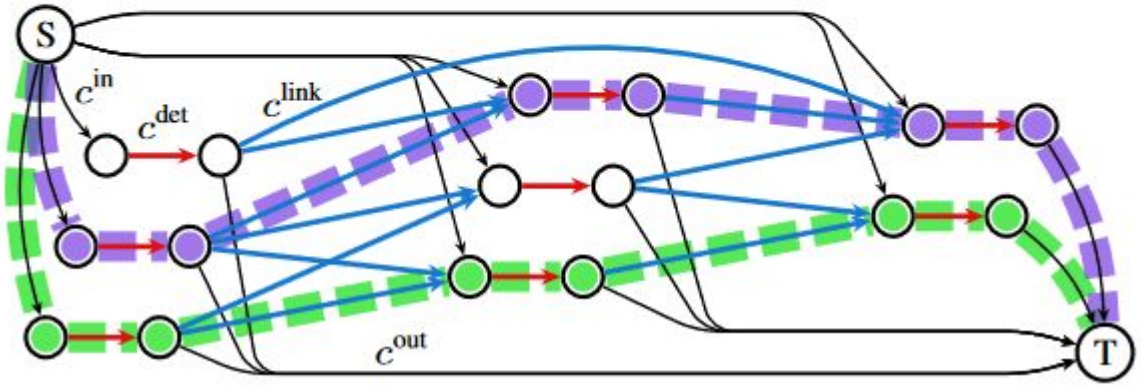


multi-cut

- Global optimization for a whole video is impractical if there are too many objects.
  - Merging nearby bounding boxes together to get reliable tracklets.
- To trade off speed against accuracy, we can do optimization in a window.

Network Flow

- Each detection d_i is represented with two nodes connected by an edge (red)
- Add a source node and a sink node represent appearing and disappearing
- Every edge has a cost c
- Every edge has a binary flow variable x means if they belong to the same trajectory
- Relax the binary constraint on x to solve the problem



$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}$$

$$\text{s.t. } \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{C}\mathbf{x} = \mathbf{0},$$

$$\mathbf{A} = [\mathbf{I}, -\mathbf{I}]^\top \in \mathbb{R}^{2M \times M} \text{ and } \mathbf{b} = [\mathbf{1}, \mathbf{0}]^\top \in \mathbb{R}^{2M}$$

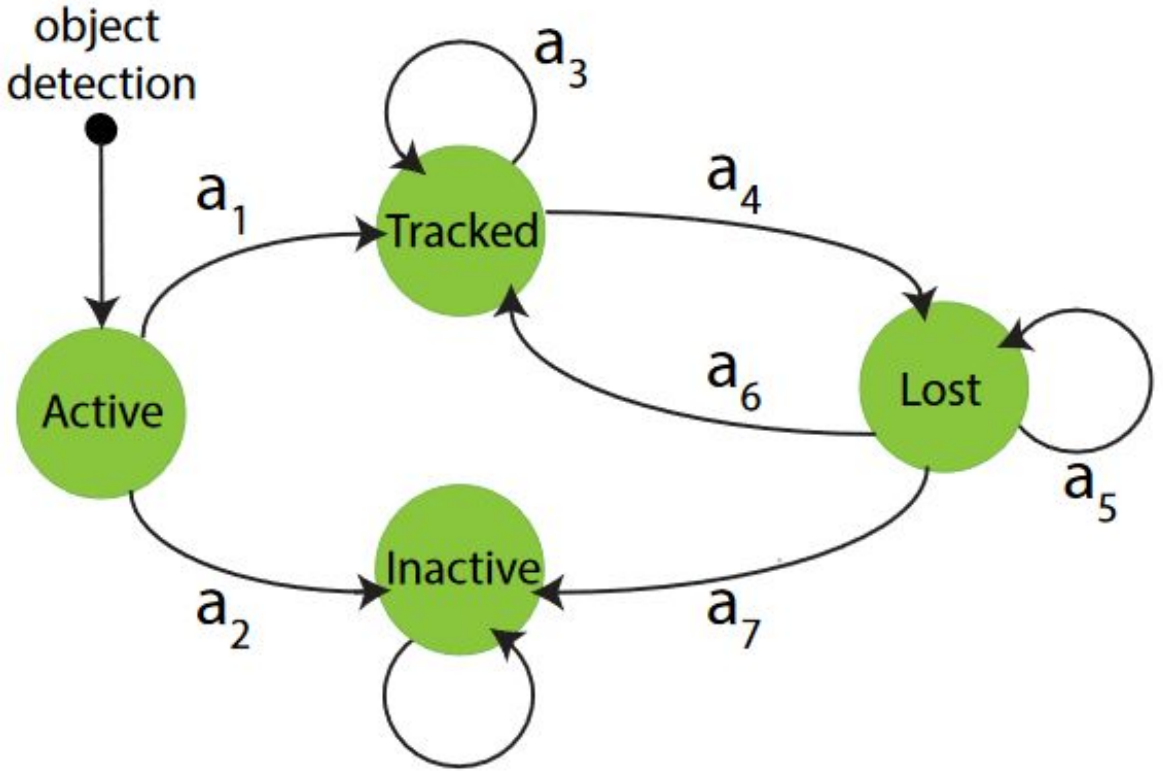
$$x_i^{\text{in}} + \sum_j x_{ji}^{\text{link}} = x_i^{\text{det}}$$

$$x_i^{\text{out}} + \sum_j x_{ij}^{\text{link}} = x_i^{\text{det}} \ \forall i$$

Zhang, Li, Yuan Li, and Ramakant Nevatia. "Global data association for multi-object tracking using network flows." *CVPR*, 2008.

State Transition

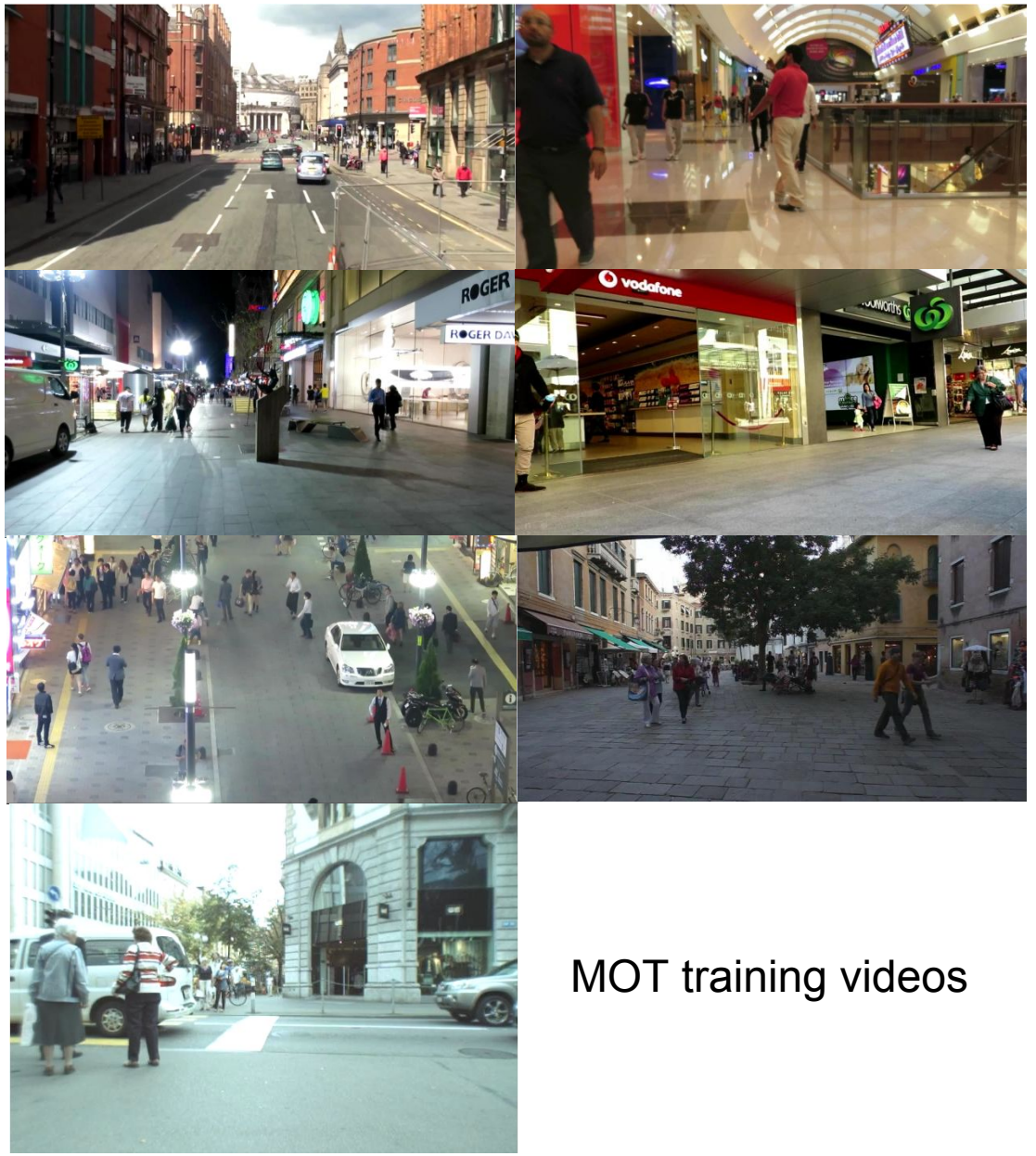
- a6: long term tracking, do interpolation if necessary.
- a7: objects lost for more than specified frames will no longer be considered.



Xiang, Yu, Alexandre Alahi, and Silvio Savarese. "Learning to track: Online multi-object tracking by decision making." *ICCV*. 2015.

Benchmark

- MOT
  - https://motchallenge.net
  - Pedestrian tracking
  - 7 training videos and 7 test videos
- KITTI
  - http://www.cvlibs.net/datasets/kitti/eval_tracking.php
  - Car and pedestrian tracking
- ImageNet VID
  - http://image-net.org/challenges/LSVRC/2017/
  - 30 classes



MOT training videos

Evaluation Metrics:

- Multiple object tracking accuracy (MOTA):
  - It is a combination of three errors:

$$MOTA = 1 - \frac{\sum_t (fn_t + fp_t + id\_sw_t)}{\sum_t g_t}$$

where id_sw_t means number of identity switches in frame t

Milan, Anton, et al. "MOT16: A benchmark for multi-object tracking." *arXiv preprint arXiv:1603.00831* (2016).

Summary

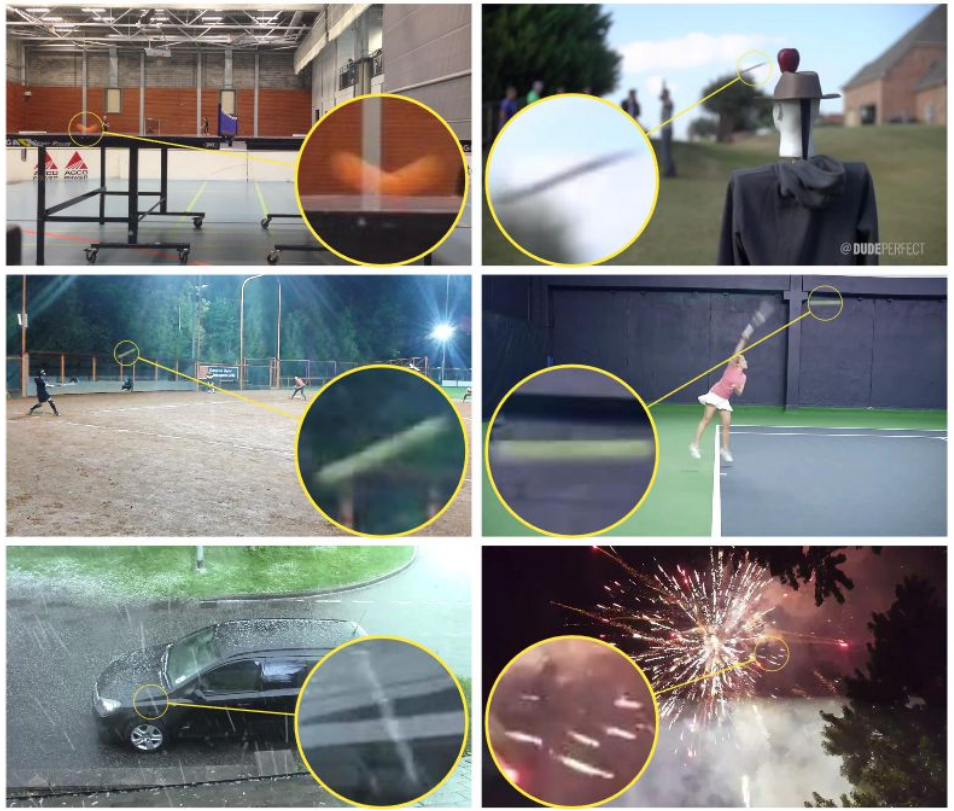
- “Visual object tracking” is not a single problem, but a series of problems.
- The area is just starting to be affected by CNNs.
- Key components of trackers:
  - Representation of object’s appearance, location and motion
  - Integration with detection
- Speed is very important for real applications.

Outline

1. Motion Estimation / Optical Flow
2. Single Object Tracking
3. Multiple Object Tracking
4. **Other**

World of Fast Moving

- Fast Moving Object (FMO)
  - An object that moves over a distance exceeding its size within the exposure time



Rozumnyi D, Kotera J, Sroubek F, et al. “The World of Fast Moving Objects”. *CVPR*. 2017.

Multiple Camera Tracking

- Tracking between cameras
  - Cameras may have overlap
  - Time of cameras need to be synchronized
  - Calibration of cameras

Tracking with Multiple Cues

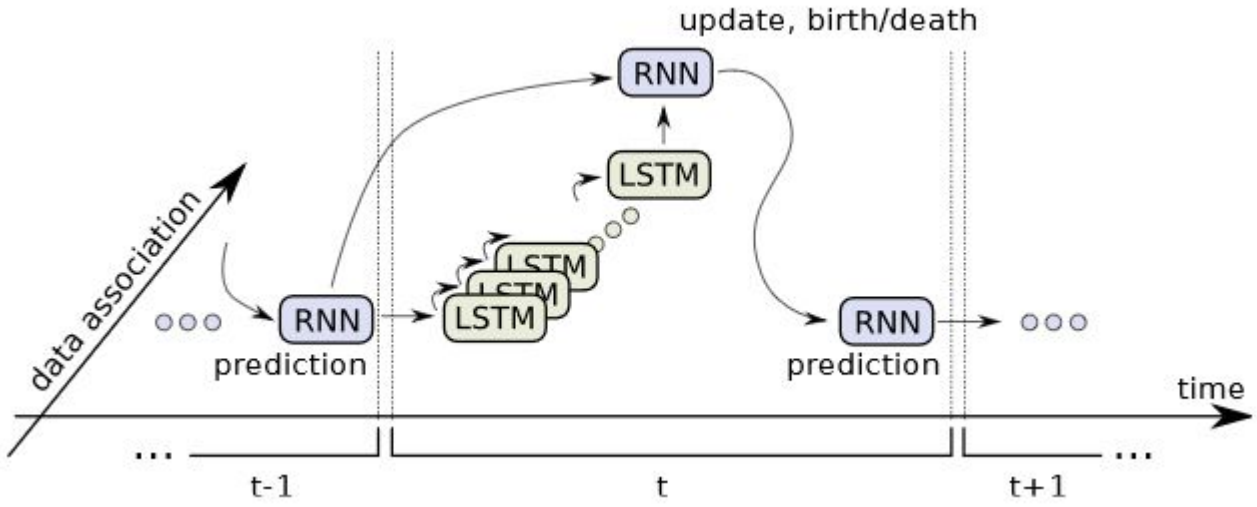
- With multiple detectors:
  - Head + pedestrian detector for pedestrian tracking
- With key points:
  - Skeleton for pedestrian tracking
  - Landmark for face tracking
- With semantic segmentation
  - Semantic optical flow
- With RGBD camera

Crowds Tracking

Saad Ali and Mubarak Shah. “Floor Fields for Tracking in High Density Crowd Scenes”. *ECCV*. 2008

Multiple Object Tracking with NN

- Milan, Anton, et al. "Online Multi-Target Tracking Using Recurrent Neural Networks“. *AAAI*. 2017.



Multiple Object Tracking with NN

- Son, Jeany, et al. "Multi-Object Tracking with Quadruplet Convolutional Neural Networks." *CVPR*. 2017.

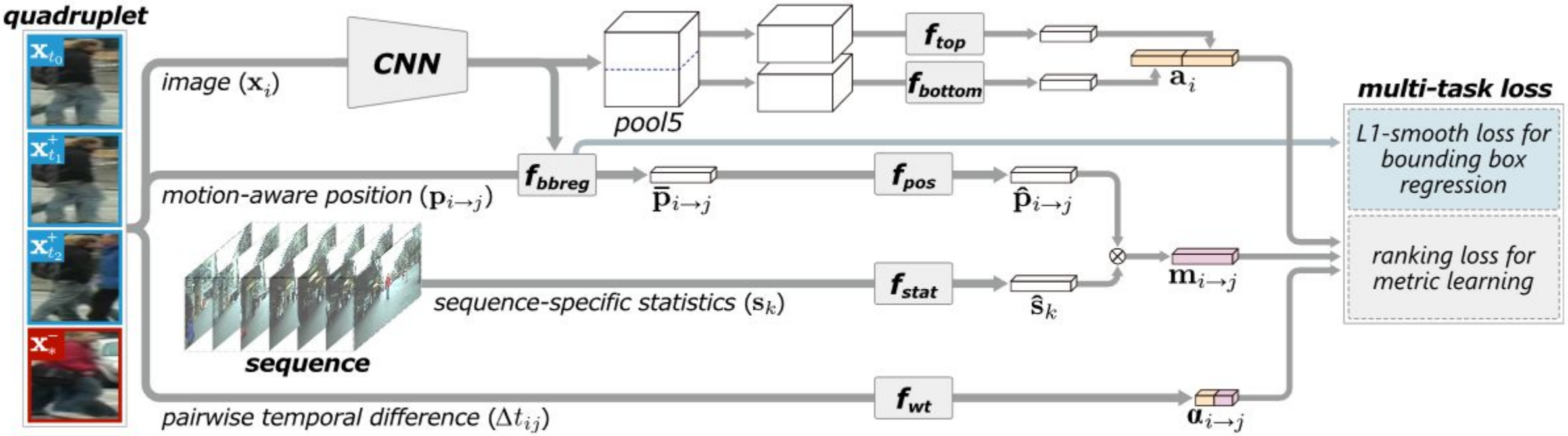


Figure 2: The architecture of the proposed Quad-CNN for multi target tracking using temporal coherency.

Thank You !
Q&A

"… Although tracking itself is by and large a solved problem..."

-- Jianbo Shi & Carlo Tomasi, CVPR 1994