NYU TANDON SCHOOL OF ENGINEERING

Robot Perception

Optical Flow & Tracking

Dr. Chen Feng

cfeng@nyu.edu

ROB-GY 6203, Fall 2023

Overview

- Optical flow & Tracking
  - + Basics definitions
  - ++ KLT
  - ++ Mean-shift
  - + Correlation filter
- CNNs for Tracking
  - + Supervised
  - + Self-supervised

- *: know how to code
- ++: know how to derive
- +: know the concept

References

- Sz: Ch 7.1.5, 9.4

- Baker, Simon, and Iain Matthews. "Lucas-kanade 20 years on: A unifying framework." *International journal of computer vision*56.3 (2004): 221-255.
- Comaniciu, Dorin, and Peter Meer. "Mean shift: A robust approach toward feature space analysis." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (2002): 603-619.
- Feichtenhofer, Christoph, Axel Pinz, and Andrew Zisserman. "Detect to track and track to detect." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3038-3046. 2017.
- Wang, Xiaolong, Allan Jabri, and Alexei A. Efros. "Learning correspondence from the cycle-consistency of time." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2566-2576. 2019.

Tracking an Object

Why Tracking?

Other than *classification* and *detection*, object *tracking* is one of the fundamental primitive tasks that is often required for more complex, real-world relevant downstream applications.

Can you think of any such downstream tasks?

Gif from https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

Tracking Applications

- monitoring, assistance, surveillance, control, defense
- robotics, autonomous car driving, rescue
- measurements: medicine, sport, biology, meteorology
- **human computer interaction**
- **augmented reality**
- **film production and postproduction: motion capture, editing**
- **management of video content: indexing, search**
- **action and activity recognition**
- image stabilization
- mobile applications

Slides from Jiri Matas 2016

Tracking Applications

Slides from Jiri Matas 2016

Tracking Definition

- [Forsyth and Ponce, Computer Vision: A modern approach, 2003]
  - “Tracking is the problem of generating an inference about the motion of an object given a sequence of images.”
- Another definition
  - Establishing point-to-point correspondences in consecutive frames of an image sequence
- Yet another definition
  - Given an initial estimate of its position, locate X in a sequence of images
  - X can be a region, an interest point, or an object

Slides from Jiri Matas 2016

Tracking Definition

Given an initial estimate of the pose and state of X :

In all images in a sequence, (in a causal manner)

1. *estimate the pose and state of X*
2. *(optionally) update the model of X*

- Pose: any geometric parameter (position, scale, …)
- State: appearance, shape/segmentation, visibility, articulations
- Model update: essentially a semi-supervised learning problem
  - a priori information (appearance, shape, dynamics, …)
  - labeled data (“track this”) + unlabeled data = the sequences
- Causal: for estimation at T, use information from time $t \leq T$

Slides from Jiri Matas 2016

Types of Tracking – Time Variations

Short-term Trackers:

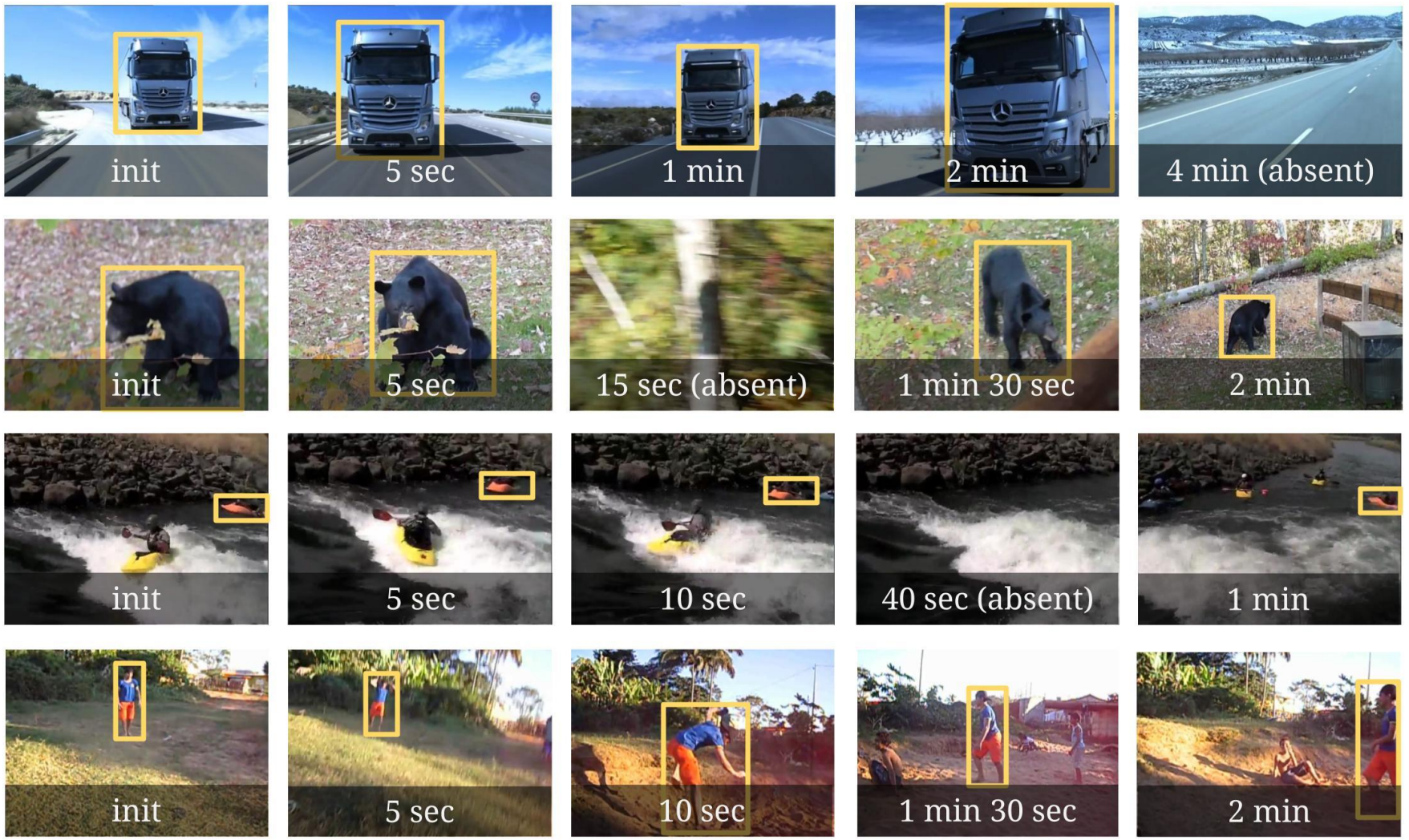
- primary objective: "where is X?" = precise estimation of pose
- secondary: be fast; don't lose track
- evaluation methodology: frame number where failure occurred
- examples: Lucas Kanade tracker, mean-shift tracker

Long-term Tracker-Detectors:

- primary objective: unsupervised learning of a detector, since *every (short-term) tracker fails, sooner or later* (disappearance from the field of view, full occlusion)
- avoid the "*first failure means lost forever*" problem
- close to online-learned detector, but assumes and exploits the fact that a sequence with temporal pose/state dependence is available
- evaluation methodology: precision/recall, false positive/negative rates (i.e. like detectors)
- note: the detector part may help even for short-term tracking problems, provides robustness to fast, unpredictable motions.

Slides from Jiri Matas 2016

Example of Long-Term Tracking Sequences



Long-term Tracking in the Wild: A Benchmark, Jack valmadre et al., 2018

Types of Tracking – Instance Variations

Note that the variations are *orthogonal* – *e.g.* can be both **MOT** and **Long-Term** in a given task!

Single Object Tracking (SOT)

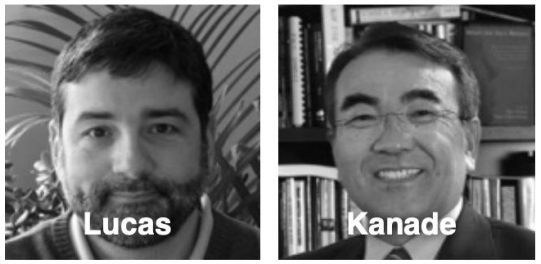
- Tracks a single subject (object) only.
- Usually the simpler task. We can do MOT by stacking SOT *if* we can tell the entities apart.

Multi Object Tracking (MOT)

- Tracks multiple subjects (objects) in a given seen.
- Typically, the harder task since we have to not only *track* a subject, but also often times we have to *disambiguate* the subjects from one another.

MotionTrack: Learning Robust Short-term and Long-term Motions for Multi-Object Tracking, Zheng Qin et al., 2023

KLT Tracker



An Iterative Image Registration Technique
with an Application to Stereo Vision.

1981

History of the

Kanade-Lucas-Tomasi (KLT) Tracker

Detection and Tracking of Feature Points.

1991

The original KLT algorithm

Good Features to Track.

1994

Slides from Kris Kitani

KLT Tracker

Kanade-Lucas-Tomasi

How should we track them from frame to frame?

Lucas-Kanade

Method for aligning (tracking) an image patch

How should we select features?

Tomasi-Kanade

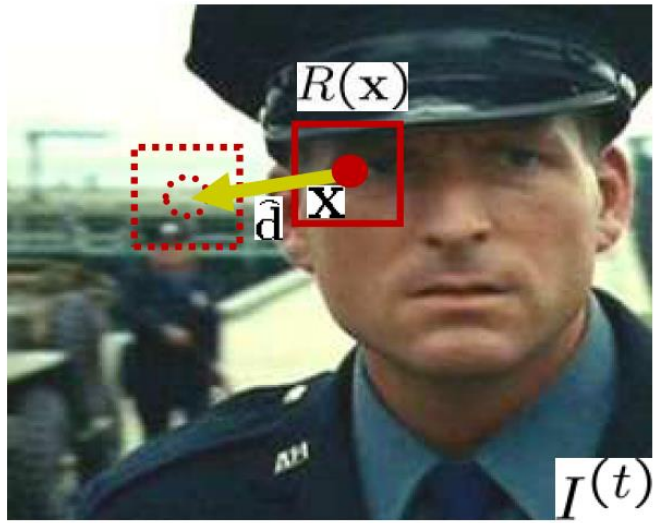
Method for choosing the best feature (image patch) for tracking

Slides from Kris Kitani

KLT Tracker

– Problem: tracking "key points" (SIFT, SURF, STAR, RIFF, FAST), or random image patches, as long as possible
  - Input: detected/chosen patches
  - Output: *tracklets* of various life-spans

slide credit: Patrick Perez



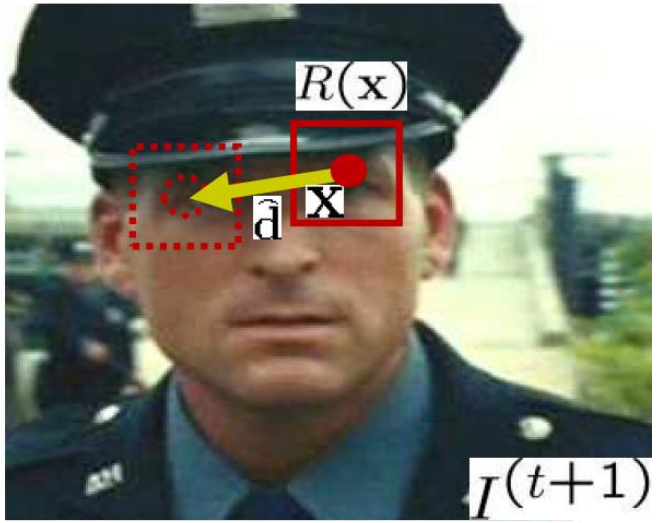
$$\hat{\mathbf{d}} = \arg\min_{\mathbf{d}} \underbrace{\sum_{\mathbf{p}\in R(\mathbf{x})} |I^{(t+1)}(\mathbf{p}+\mathbf{d}) - I^{(t)}(\mathbf{p})|^2}_{\text{SSD}}$$

Slides from Jiri Matas 2016

KLT Tracker

– Problem: tracking "key points" (SIFT, SURF, STAR, RIFF, FAST), or random image patches, as long as possible
  - Input: detected/chosen patches
  - Output: *tracklets* of various life-spans

slide credit: Patrick Perez



$$\hat{\mathbf{d}} = \arg\min_{\mathbf{d}} \underbrace{\sum_{\mathbf{p}\in R(\mathbf{x})} |I^{(t+1)}(\mathbf{p}+\mathbf{d}) - I^{(t)}(\mathbf{p})|^2}_{\text{SSD}}$$

Slides from Jiri Matas 2016

KLT Tracker

$$f(x) = f(a) + f'(a)(x-a) + R_2$$

– First assuming small displacement: 1st-order Taylor expansion inside SSD

$$\hat{d} = \underset{d}{\operatorname{argmin}} \sum_{p \in R(x)} \left| I^{(t+1)}(p) + \nabla I^{(t+1)}(p)^T d - I^{(t)}(p) \right|^2$$

$$\hat{d} = -\left(\sum_{p \in R(x)} \nabla I^{(t+1)}(p) \nabla I^{(t+1)}(p)^T \right)^{-1} \sum_{p \in R(x)} \nabla I^{(t+1)}(p) [I^{(t+1)}(p) - I^{(t)}(p)]$$

For good conditioning, patch must be textured/structured enough:

- Uniform patch: no information
- Contour element: aperture problem (one dimensional information)
- Corners, blobs and texture: best estimate




[Lucas and Kanda 1981][Tomasi and Shi, CVPR'94]

minimize β $\|Y - X\beta\|^2$

$$\hat{\beta} = (X^\mathsf{T} X)^{-1} X^\mathsf{T} Y$$

Slides from Jiri Matas 2016

KLT Tracker

– Translation is usually sufficient for small fragments, but:
  • Perspective transforms and occlusions cause drift and loss
– Two complementary options
  • Kill tracklets when minimum SSD too large
  • Compare as well with *initial patch under affine transform (warp)* assumption

slide credit:
Patrick Perez

$$\hat{\mathbf{d}} = \arg\min_{\mathbf{d}} \sum_{\mathbf{p}\in R_t} |I^{(t+1)}(\mathbf{p}+\mathbf{d}) - I^{(t)}(\mathbf{p})|^2$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{\mathbf{p}\in R_0} |I^{(t+1)}(\mathbf{w}[\mathbf{p}]) - I^{(0)}(\mathbf{p})|^2$$

Slides from Jiri Matas 2016

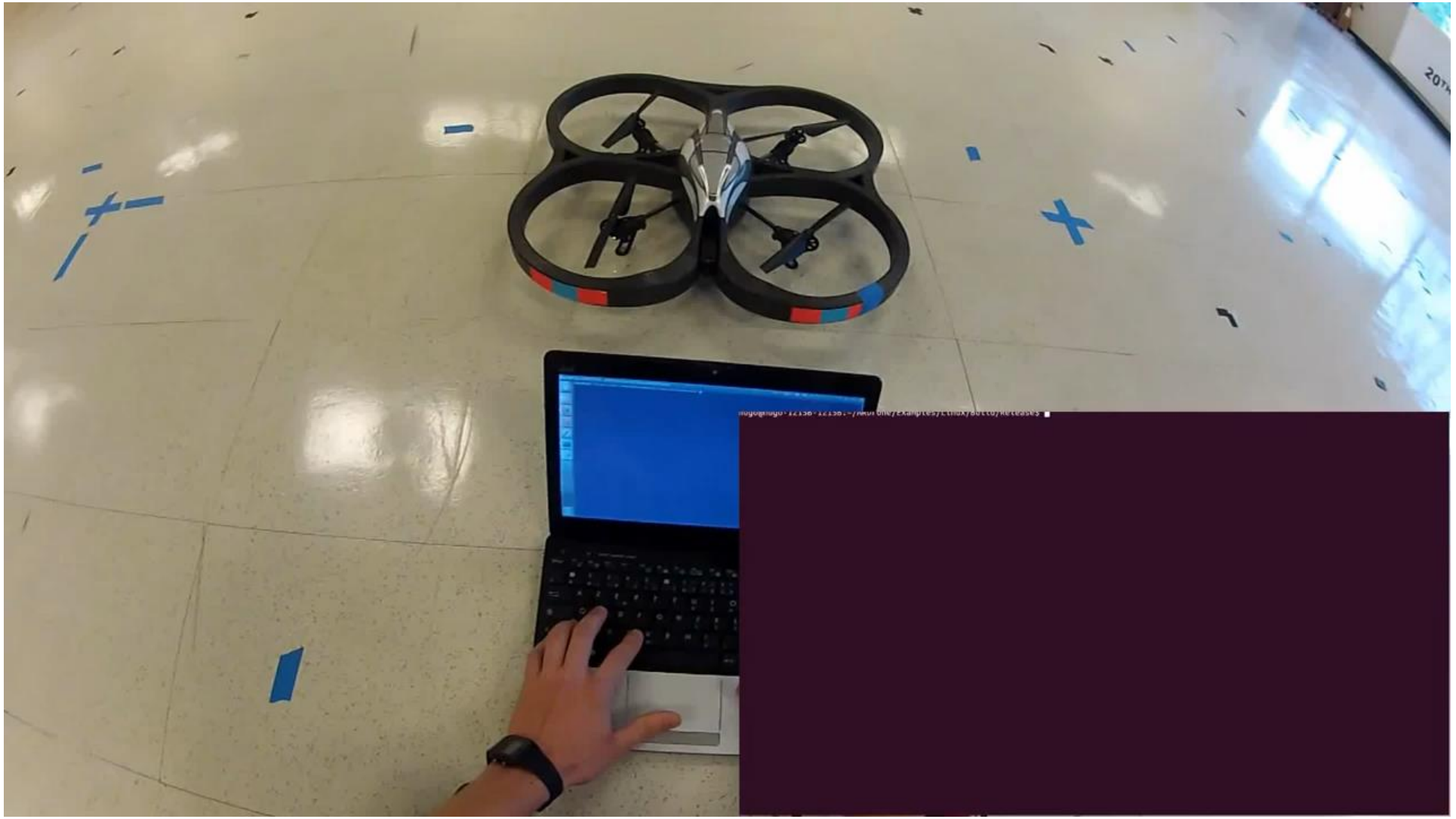
KLT Tracker

- cost function: sum of squared intensity differences between template and window
- optimization technique: gradient descent
- model learning: no update

- attractive properties:
  - fast
  - easily extended to image-to-image transformations with multiple parameters

Slides from Jiri Matas 2016

Optical Flow in Robotics



https://youtu.be/C95bngCOv9Q

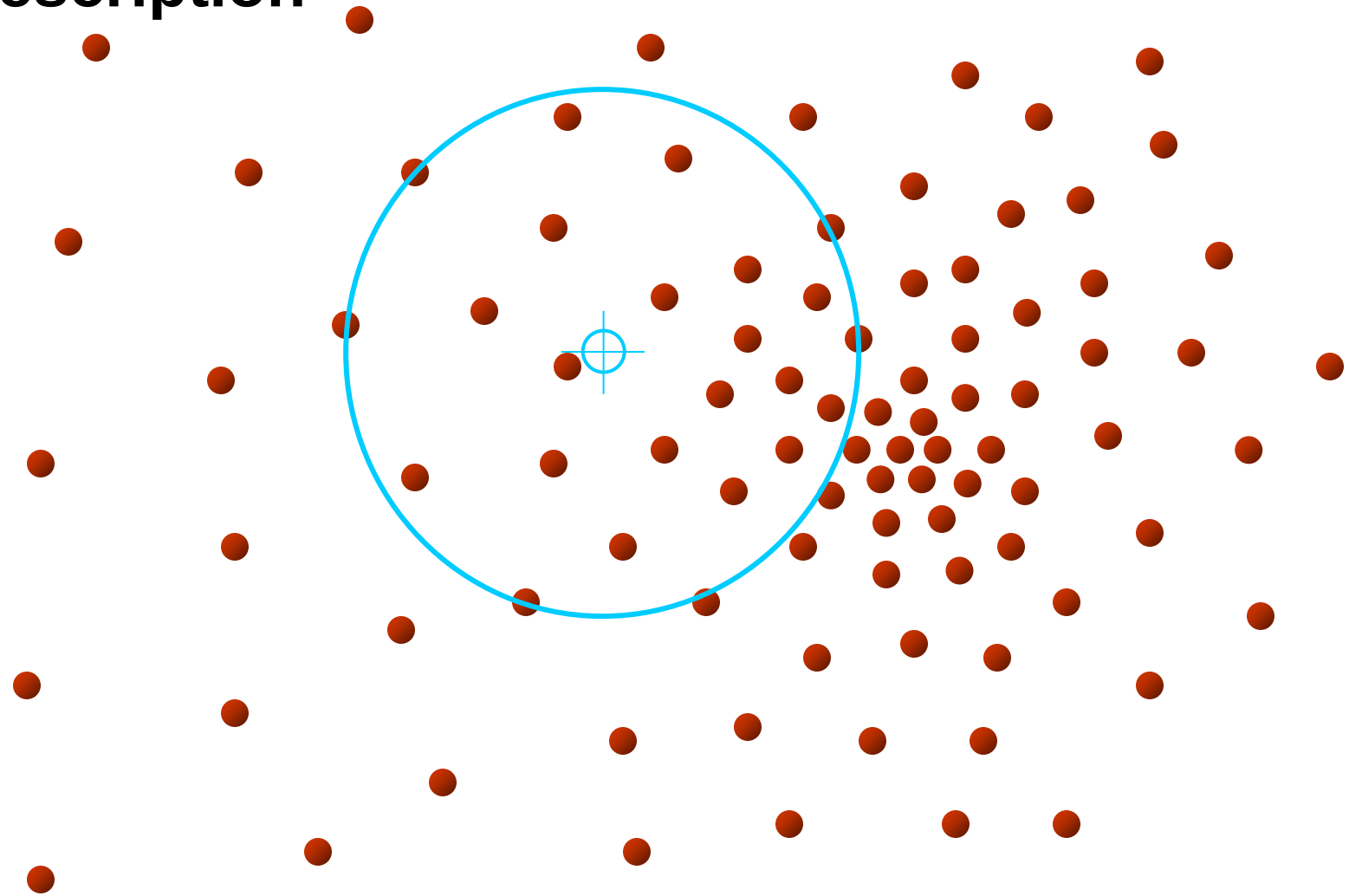
Mean Shift Theory

Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer.
"Real-Time Tracking of Non-Rigid Objects Using Mean Shift."

Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000. Vol. 2. IEEE, 2000.

Won the Best Paper Award CVPR 2000

Intuitive Description

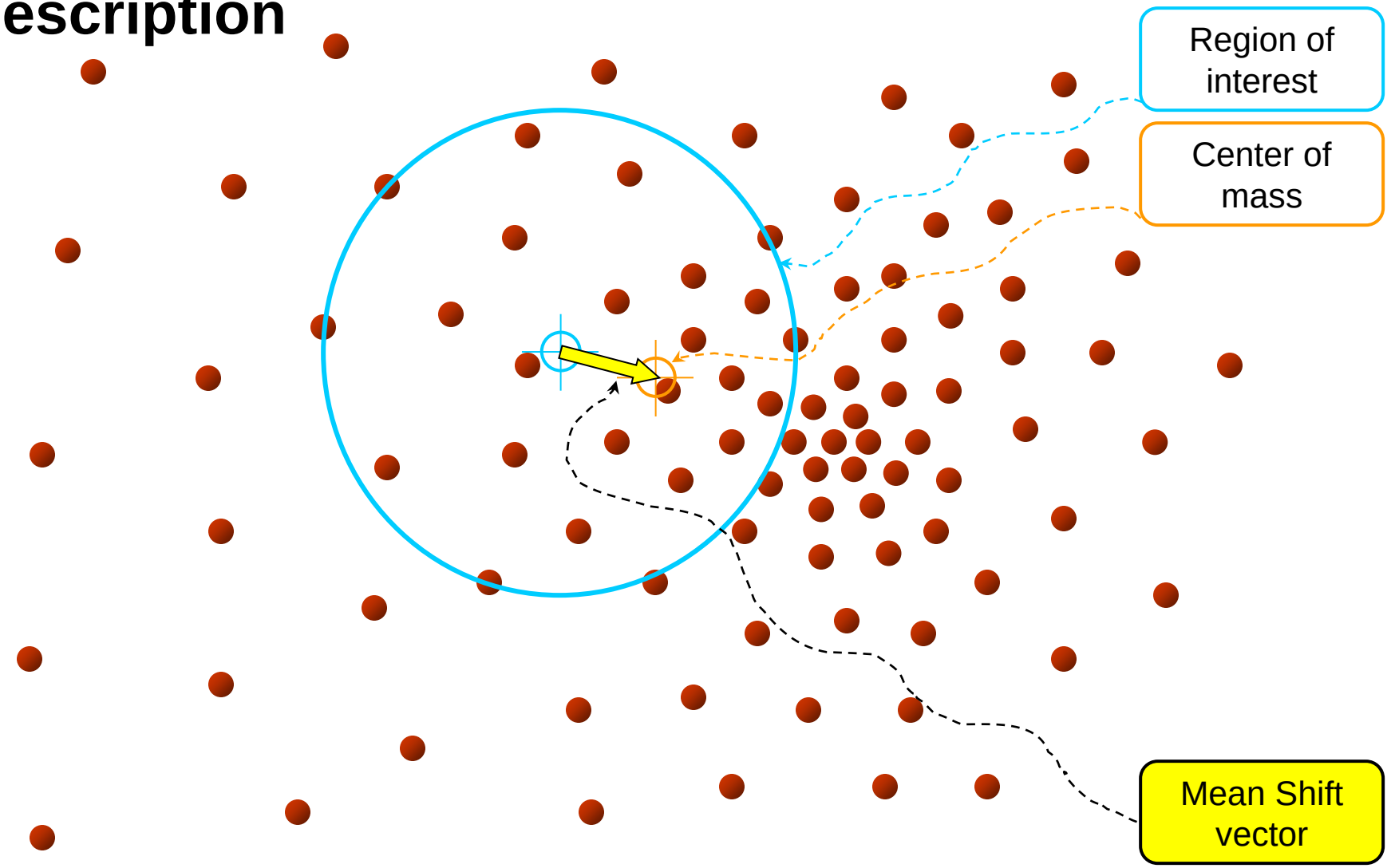


Objective : Find the densest region

Distribution of identical billiard balls

Slides from Ukrainitz & Sarel

Intuitive Description

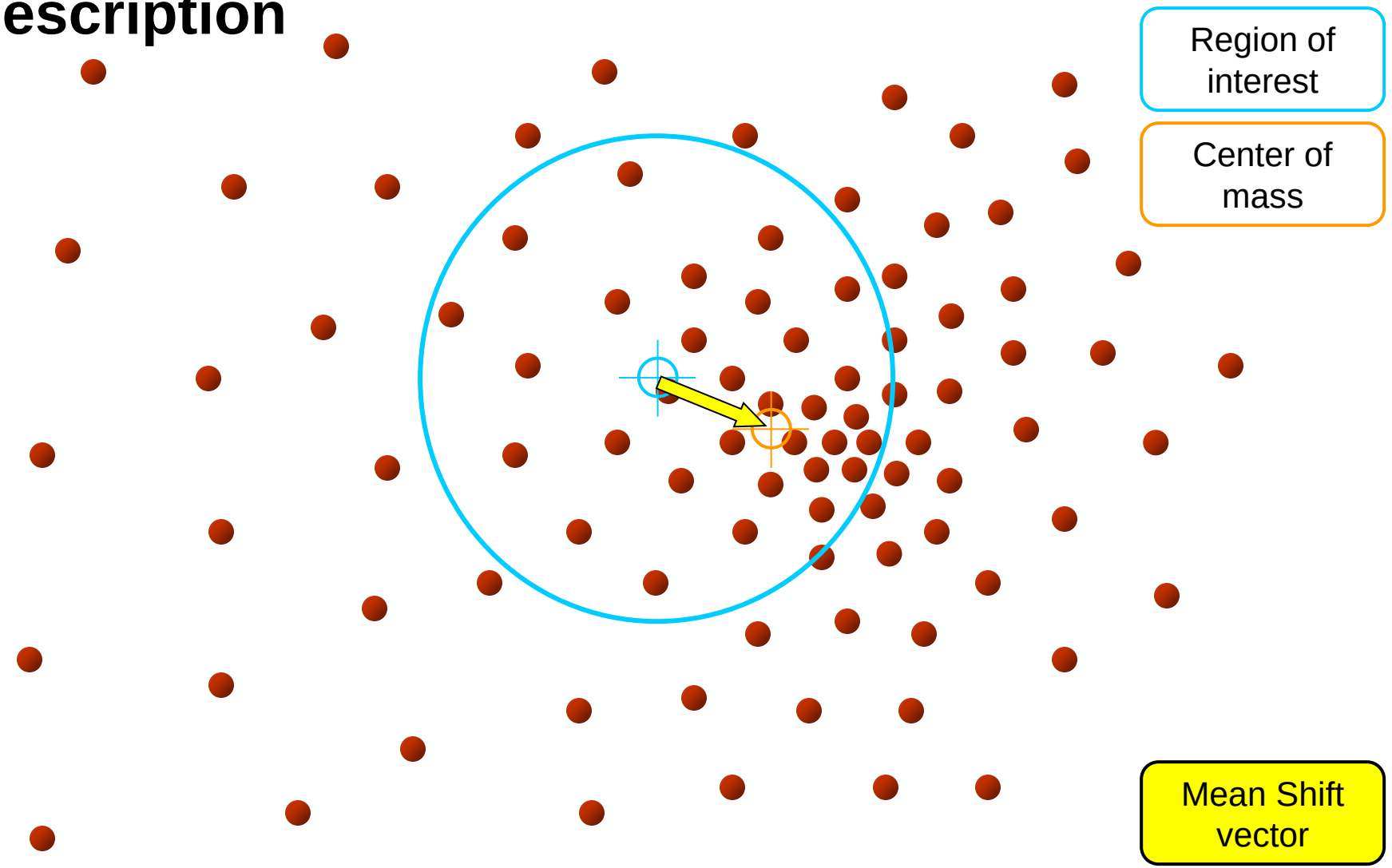


Objective : Find the densest region

Distribution of identical billiard balls

Slides from Ukrainitz & Sarel

Intuitive Description

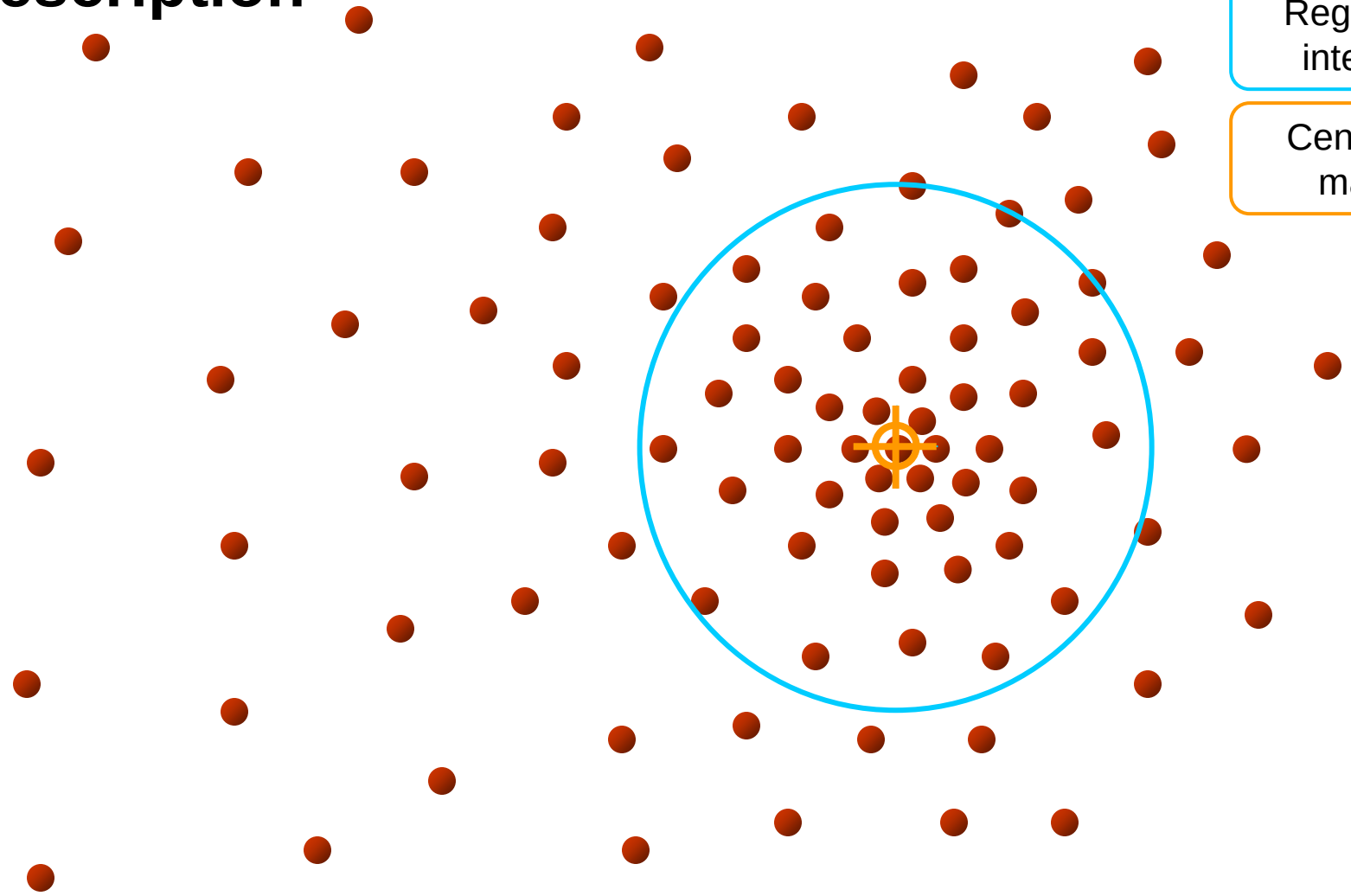


Objective : Find the densest region

Distribution of identical billiard balls

Slides from Ukrainitz & Sarel

Intuitive Description



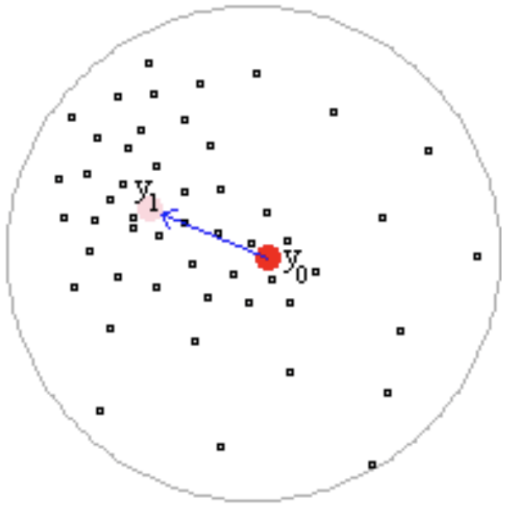
Objective : Find the densest region

Distribution of identical billiard balls

Slides from Ukrainitz & Sarel

What is Mean Shift?

- Given:
  - Data points and approximate location of the mean of this data
- Task:
  - Estimate the exact location of the mean of the data by determining the shift vector from the initial mean.
- Mean shift vector always points towards the direction of the maximum increase in the density



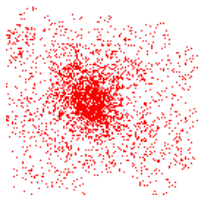
$$M_h(\mathbf{y}) = \left[\frac{1}{n_x} \sum_{i=1}^{n_x} \mathbf{x}_i \right] - \mathbf{y}_0$$

Slides from Ukrainitz & Sarel

What is Mean Shift?

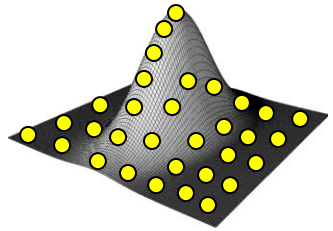
<u>A tool for</u>:
Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N

<u>PDF in feature space</u>
- **Color space**
- **Scale space**
- **Actually, any feature space you can conceive**



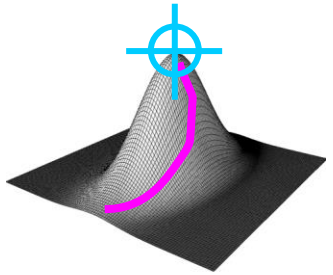
Data

Non-parametric
Density Estimation



Discrete PDF Representation

Non-parametric
Density **GRADIENT** Estimation
(Mean Shift)

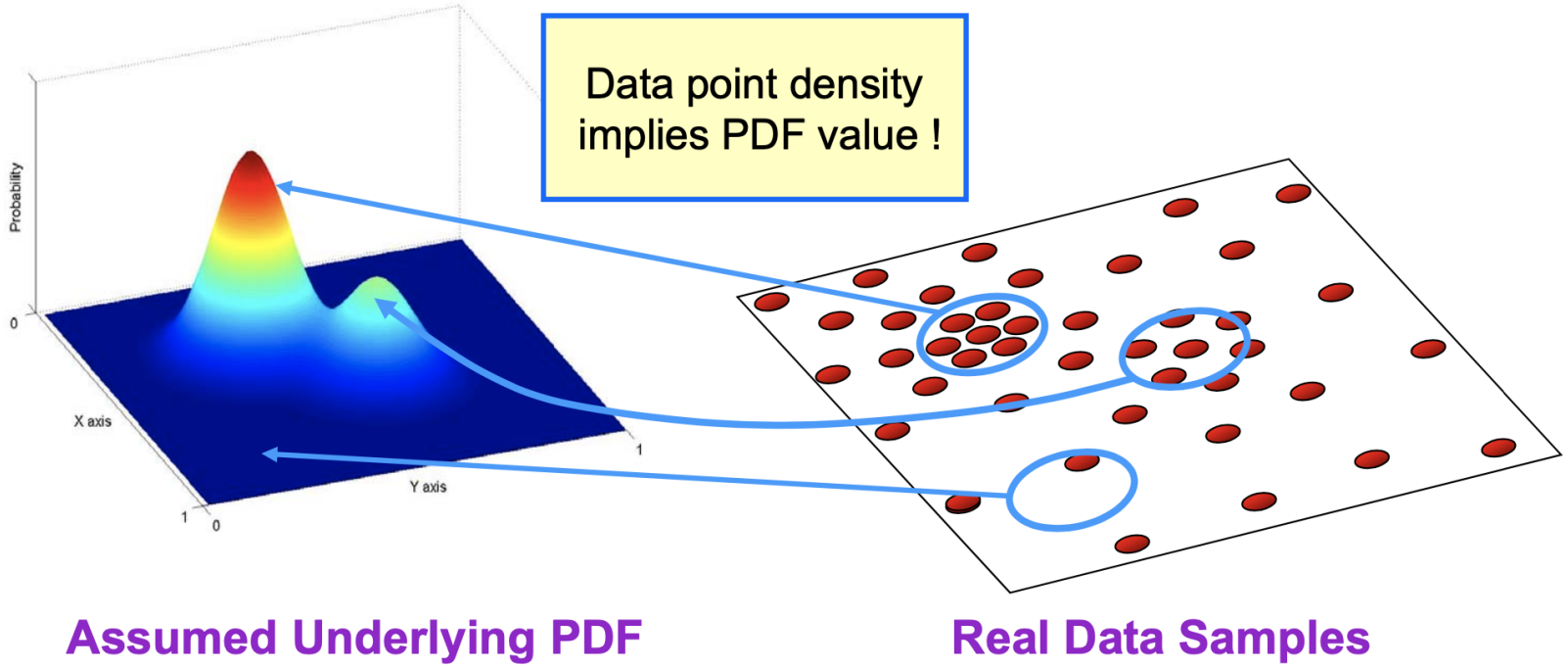


PDF Analysis

Slides from Ukrainitz & Sarel

Non-Parametric Density Estimation

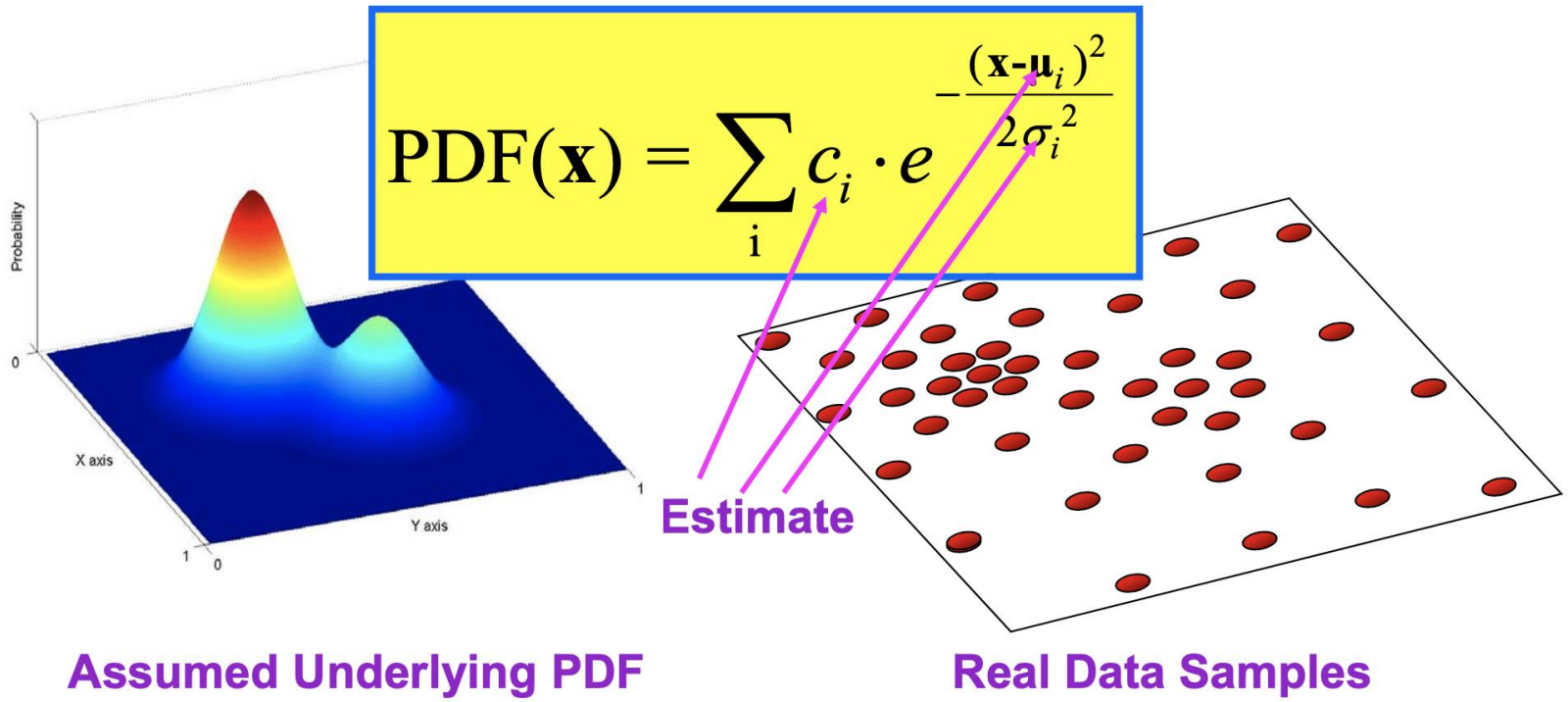
- Assumption: The data points are samples from an underlying PDF



Slides from Ukrainitz & Sarel

Parametric Density Estimation

- Assumption: The data points are samples from an underlying PDF



Slides from Ukrainitz & Sarel

Kernel Density Estimation - Function Forms

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$

Data

In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(\mathbf{x}_i) \quad \text{or} \quad K(\mathbf{x}) = ck(\|\mathbf{x}\|)$$

Same function on each dimension

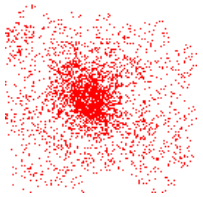
Function of vector length only

Slides from Ukrainitz & Sarel

Kernel Density Estimation - Various Kernels

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

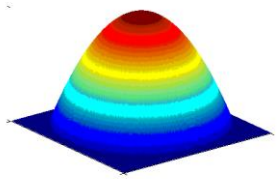
A function of some finite number of data points $x_1 \ldots x_n$



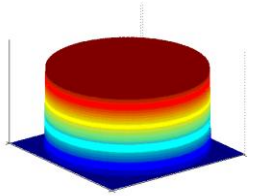
Data

Examples:

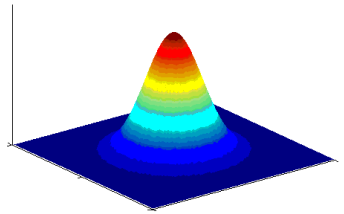
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Uniform Kernel $K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- Normal Kernel $K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$

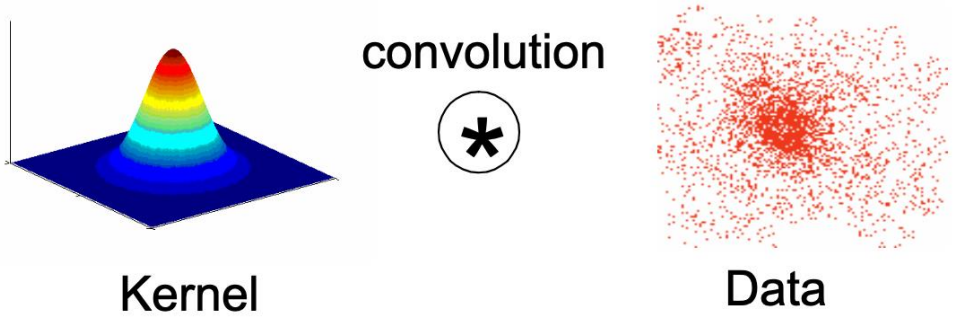


Slides from Ukrainitz & Sarel

Kernel Density Estimation - Key Idea

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

Superposition of kernels, centered at each data point is equivalent to convolving the data points with the kernel.



Slides from Ukrainitz & Sarel

Kernel Density *Gradient* Estimation

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x} - \mathbf{x}_i)$$

Give up estimating the PDF !
Estimate **ONLY** the gradient

Using the Kernel form:

$$K(\mathbf{x} - \mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$$

Size of window

We get :

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$$\mathrm{g}(\mathbf{x}) = -k'(\mathbf{x})$$

Slides from Ukrainitz & Sarel

Computing The Mean Shift

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

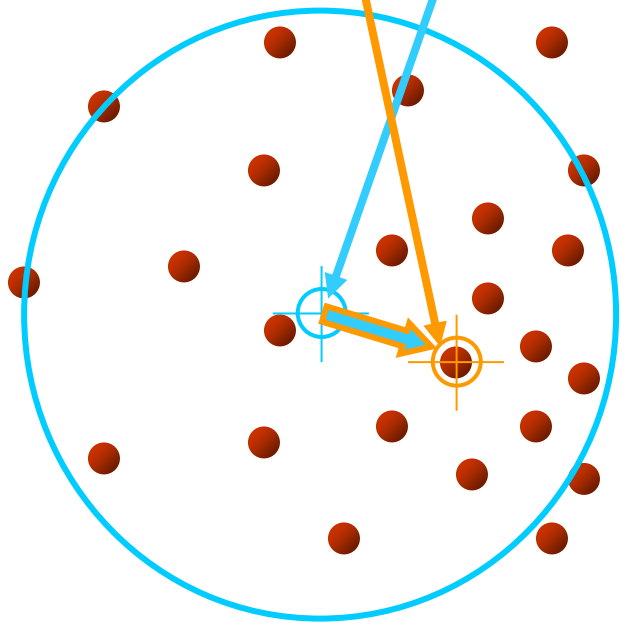
Yet another Kernel density estimation !

Mean Shift

Simple Mean Shift procedure:

- Compute mean shift vector

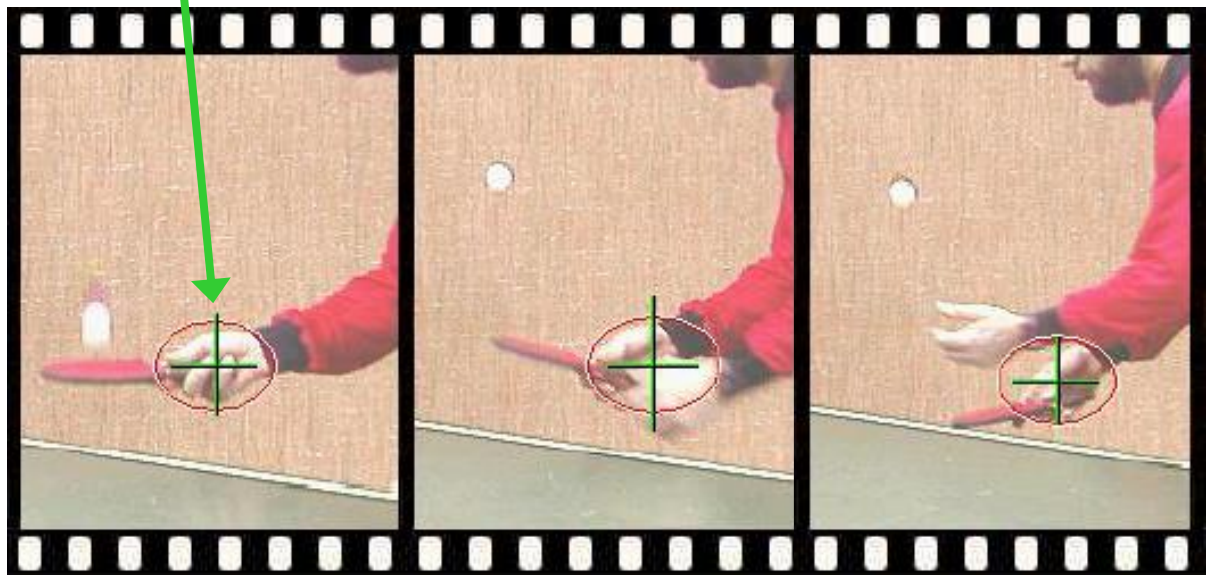
$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)} - \mathbf{x}\right]$$



- Translate the Kernel window by **m(x)**

Mean-Shift Object Tracking - General Framework: Target Representation

Choose a reference model in the current frame → Choose a feature space → Represent the model in the chosen feature space



… Current frame → Next frame

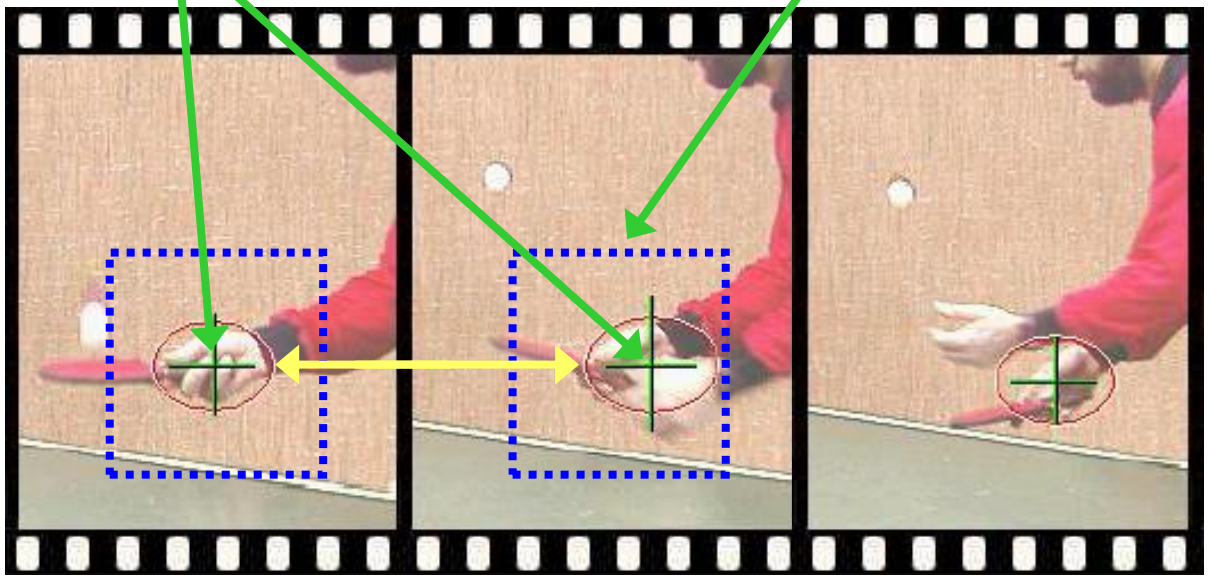
Slides from Ukrainitz & Sarel

Mean-Shift Object Tracking - General Framework: Target Localization

Start from the position of the model in the current frame

Search in the model's neighborhood in next frame

Find best candidate by maximizing a similarity func.



Model

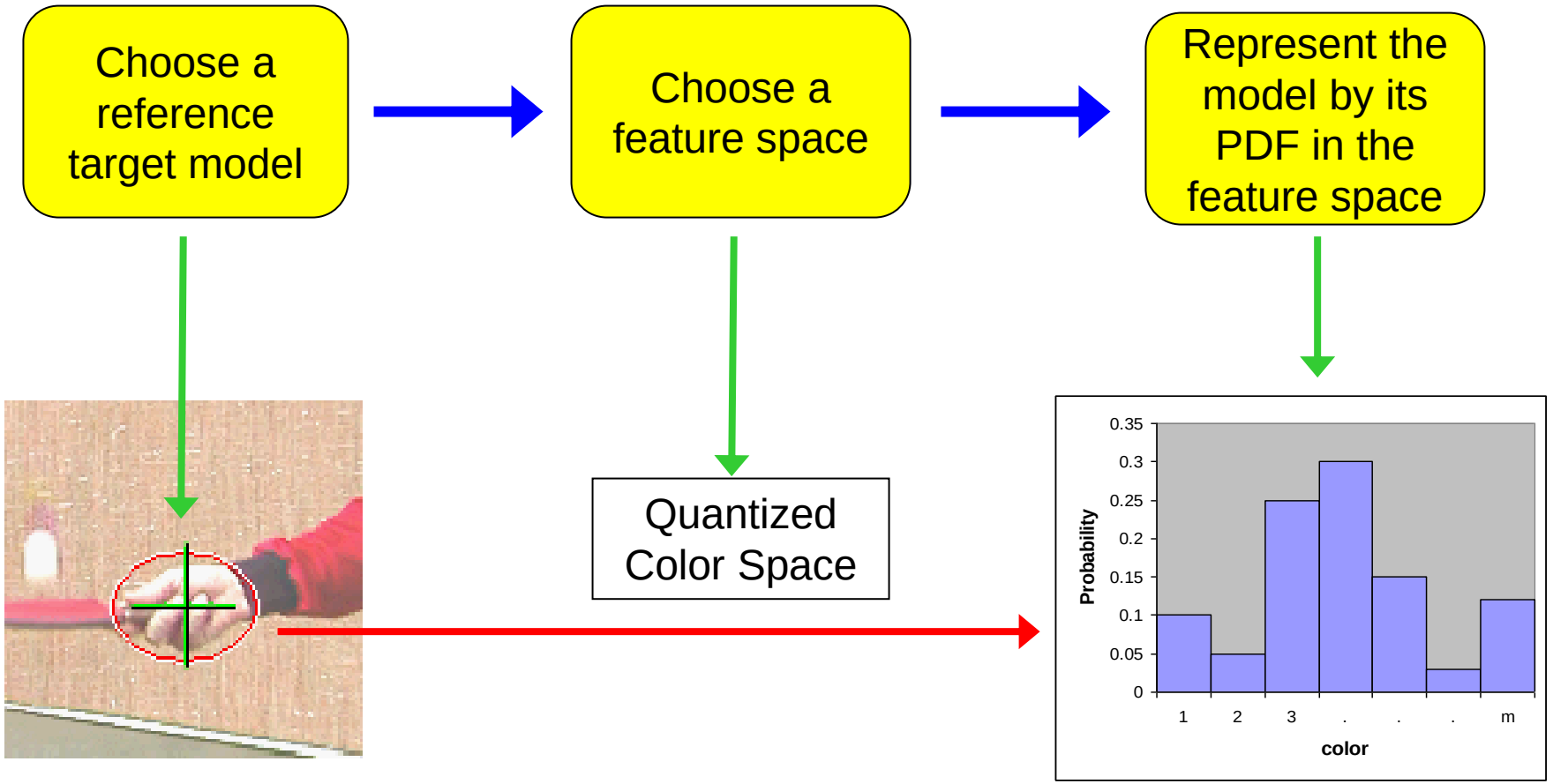
Candidate

… Current frame → Next frame

Repeat the same process in the next pair of frames

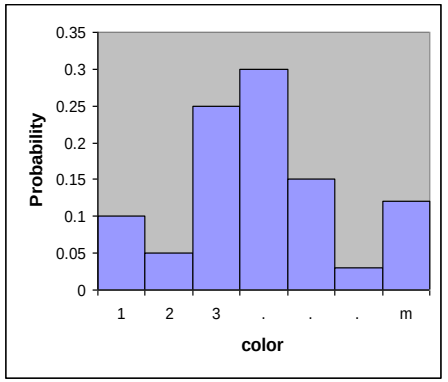
Slides from Ukrainitz & Sarel

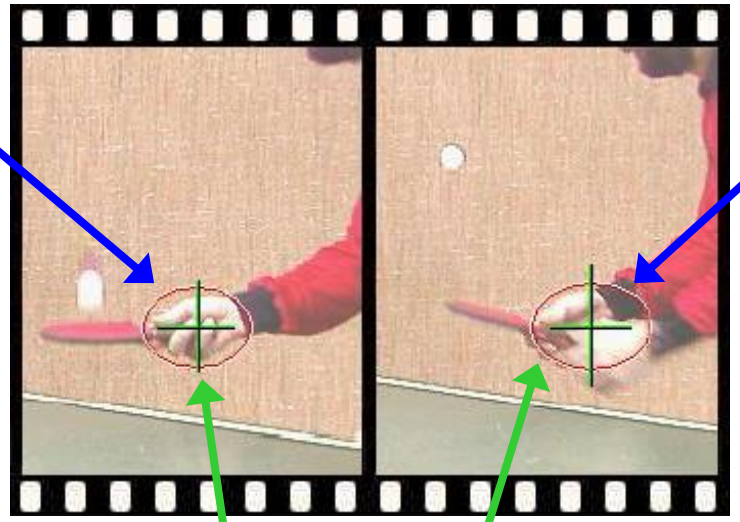
Mean-Shift Object Tracking - Target Representation



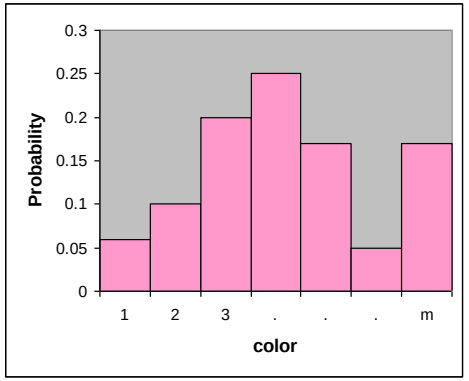
Mean-Shift Object Tracking - PDF Representation

Target Model
(centered at 0)





Target Candidate
(centered at y)



$$\vec{q} = \{q_u\}_{u=1..m} \qquad \sum_{u=1}^{m} q_u = 1$$

$$\vec{p}(y) = \{p_u(y)\}_{u=1..m} \qquad \sum_{u=1}^{m} p_u = 1$$

Similarity Function: $f(y) = f[\vec{q}, \vec{p}(y)]$

Mean-Shift Object Tracking - Similarity Function

Target model: $\vec{q} = (q_1, \dots, q_m)$

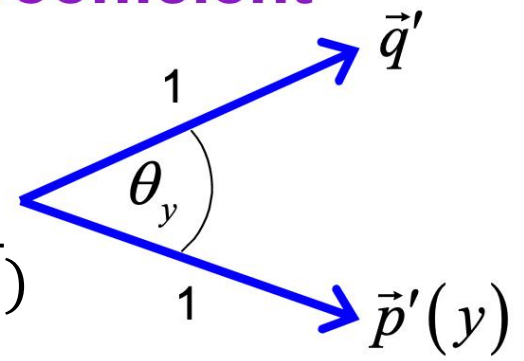
Target candidate: $\vec{p} = (p_1(y), \dots, p_m(y))$

Similarity function: $f(y) = f[\vec{p}(y), \vec{q}] = ?$

The Bhattacharyya Coefficient

$\vec{q}' = (\sqrt{q_1}, \dots, \sqrt{q_m})$

$p' = (\sqrt{p_1(y)}, \dots, \sqrt{p_m(y)})$



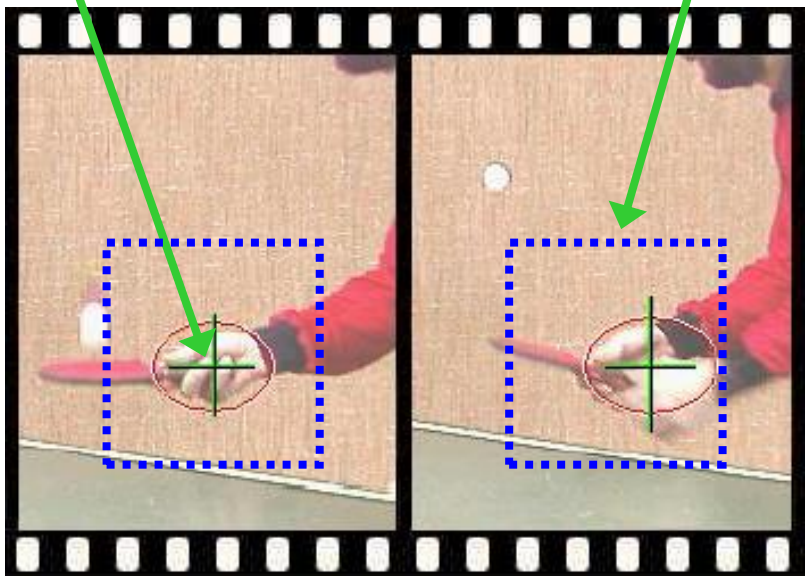
$$f(y) = \cos\theta_y = \frac{p'(y)^T q'}{\|p'(y)\|\|q'\|} = \sum_{u=1}^{m} \sqrt{p_u(y) q_u}$$

Mean-Shift Object Tracking - Target Localization Algorithm

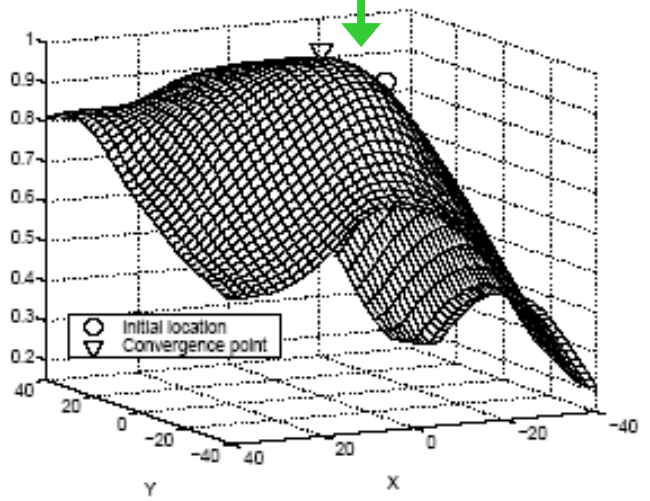
Start from the position of the model in the current frame

Search in the model's neighborhood in next frame

Find best candidate by maximizing a similarity func.



$\vec{q}$ $\vec{p}(y)$



$f\left[\vec{p}(y), \vec{q}\right]$

Mean-Shift Object Tracking - Approximating the Similarity Function

Model Location: y_i Candidate Location: y

$$f(y) = \sum_{u=1}^{m} \sqrt{p_u(y) q_u}$$

Linear approximation around y_0

$$f(y) \approx \frac{1}{2} \sum_{u=1}^{m} \sqrt{p_u(y_0) q_u} + \frac{1}{2} \sum_{u=1}^{m} p_u(y) \sqrt{\frac{q_u}{p_u(y_0)}}$$

Independent of y

, where $p_u(y) = C_h \sum_{b(x_i)=u} k\left[\left\|\frac{y - x_i}{h}\right\|^2\right]$

Sub $p_u(y)$

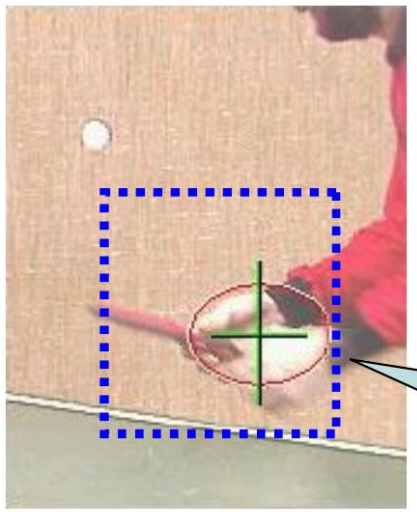
$$\frac{C_h}{2} \sum_{i=1}^{n} w_i k\left[\left\|\frac{y - x_i}{h}\right\|^2\right]$$

← Density estimate as a function of y

Mean-Shift Object Tracking - Maximizing the Similarity Function

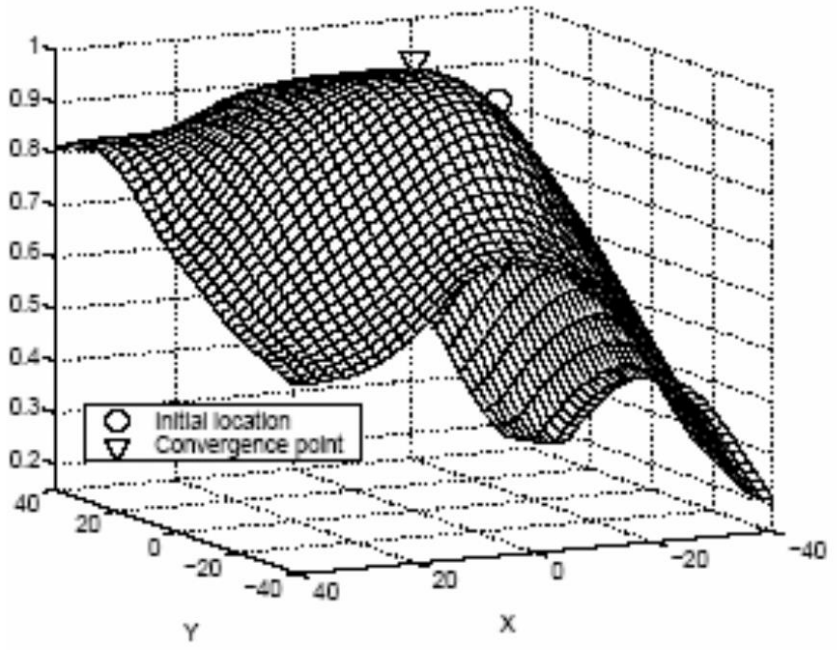
The mode of $\frac{C_h}{2}\sum_{i=1}^{n} w_i k\left[\left\|\frac{y-x_i}{h}\right\|^2\right]$ = sought maximum

Important Assumption:



The target representation provides sufficient discrimination

One mode in the searched neighborhood



Mean-Shift Object Tracking - Applying Mean-Shift

The mode of $\frac{C_h}{2}\sum_{i=1}^{n} w_i k\left[\left\|\frac{y-x_i}{h}\right\|^2\right]$ = sought maximum

Original Mean-Shift:

Find mode of $c\sum_{i=1}^{n} k\left[\left\|\frac{y-x_i}{h}\right\|^2\right]$ using $y_1 = \frac{\sum_{i=1}^{n} x_i g\left[\left\|\frac{y_0-x_i}{h}\right\|^2\right]}{\sum_{i=1}^{n} g\left[\left\|\frac{y_0-x_i}{h}\right\|^2\right]}$

Extended Mean-Shift:

Find mode of $c\sum_{i=1}^{n} \boxed{w_i} k\left[\left\|\frac{y-x_i}{h}\right\|^2\right]$ using $y_1 = \frac{\sum_{i=1}^{n} x_i \boxed{w_i} g\left[\left\|\frac{y_0-x_i}{h}\right\|^2\right]}{\sum_{i=1}^{n} \boxed{w_i} g\left[\left\|\frac{y_0-x_i}{h}\right\|^2\right]}$

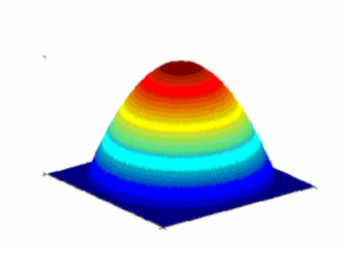
Slides from Ukrainitz & Sarel

Mean-Shift Object Tracking - Choosing the Kernels

A special class of radially symmetric kernels:

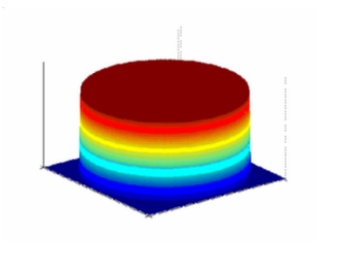
$$K(x) = ck\left(\|x\|^2\right)$$

Epanechnikov kernel:



$$k(x) = \begin{cases} 1 - x, & \|x\| \le 1 \\ 0, & else \end{cases}$$

Uniform kernel:



$$g(x) = -k(x) \begin{cases} 1, & \|x\| \le 1 \\ 0, & else \end{cases}$$

$$y_1 = \frac{\sum_{i=1}^{n} x_i w_i g\left[\left\|\frac{y_0 - x_i}{h}\right\|^2\right]}{\sum_{i=1}^{n} w_i g\left[\left\|\frac{y_0 - x_i}{h}\right\|^2\right]} \rightarrow y_1 = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$$

Likelihood Maximization Using Mean Shift Vector

Maximization of the likelihood of target and candidate depends on the weights:

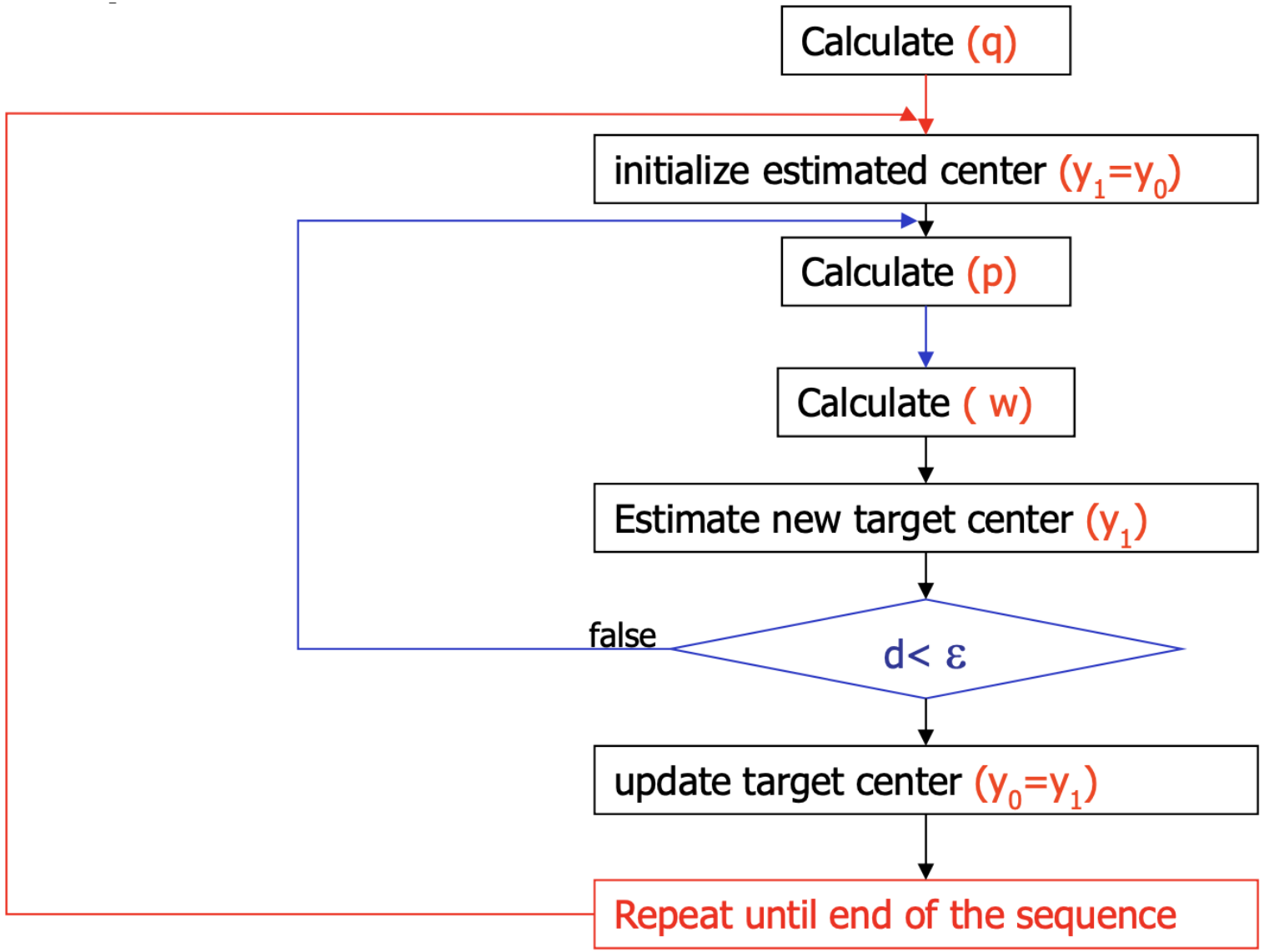
$$w_i(\mathbf{y}_o) = \sum_{u=1}^{m} \delta[S(\mathbf{x}_i) - u] \sqrt{\frac{q_u}{\hat{p}_u(\mathbf{y}_o)}}$$

Again, the mean shift vector M is:

$$M_h(\mathbf{y}_0) = \frac{\sum_{i=1}^{n_x} w_i(\mathbf{y}_0)\mathbf{x}_i}{\sum_{i=1}^{n_x} w_i(\mathbf{y}_0)} - \mathbf{y}_0$$

Thus, new target center is $\hat{\mathbf{y}} = \mathbf{y}_0 + M_h(\mathbf{y}_0)$

Algorithm


Calculate (q)
initialize estimated center (y_1=y_0)
Calculate (p)
Calculate (w)
Estimate new target center (y_1)
false
d< ε
update target center (y_0=y_1)
Repeat until end of the sequence

Mean-Shift Pros/Cons

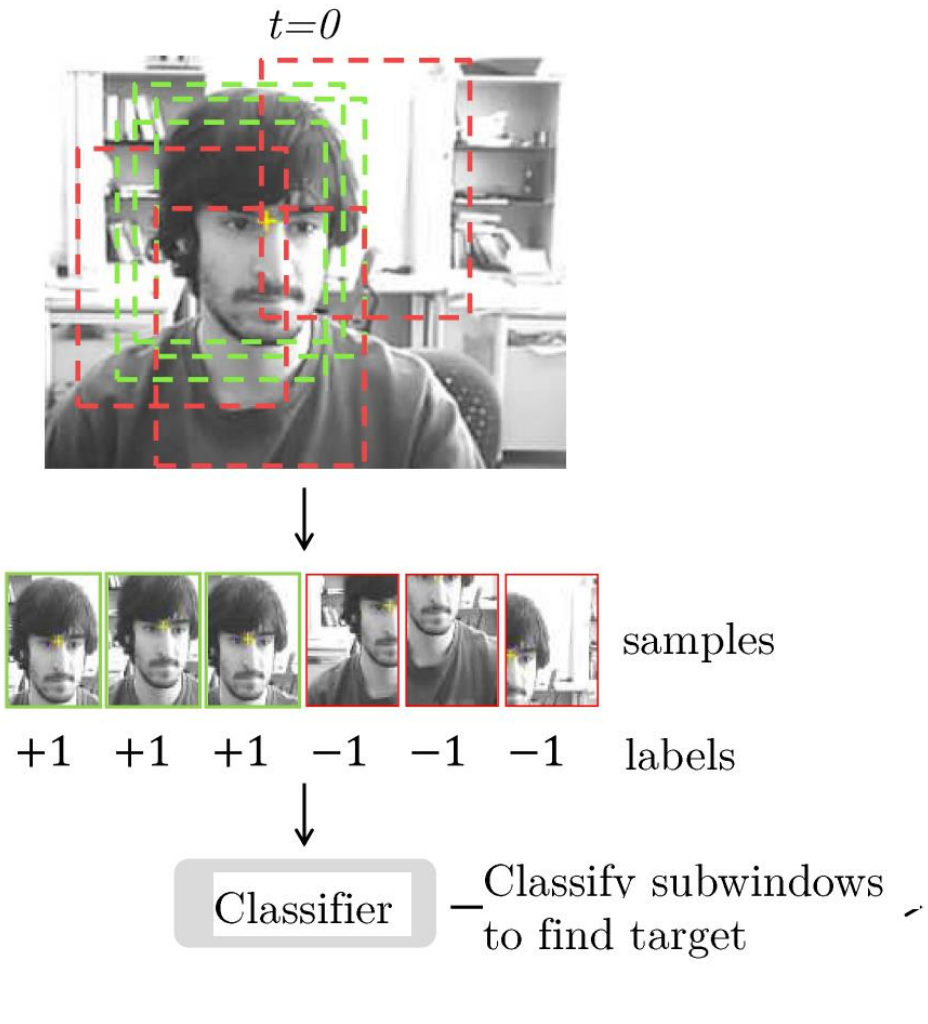
- Low computational cost (easily real-time)
- Surprisingly robust
  - Invariant to pose and viewpoint
  - Often no need to update reference color model
- Invariance comes at a price
  - Position estimate prone to fluctuation
  - Scale and orientation not well captured
  - Sensitive to color clutter (e.g., teamates in team sports)
- Local search by gradient descent
- Problems:
  - abrupt moves
  - occlusions

Slides from Jiri Matas 2016

Mean-Shift Example

https://youtu.be/RG5uV_h50b0

Discriminative Tracking: Tracking by Detection



Slides from Jiri Matas 2016

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

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

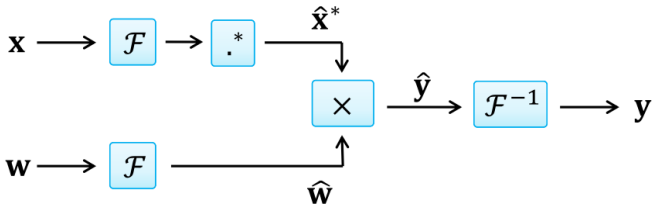
- Note: Convolution is related; it is the same as cross-correlation, but with the flipped image of $\mathbf{w}$ (P → d).

The Convolution Theorem

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

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

- In practice:

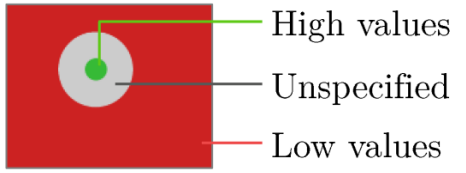


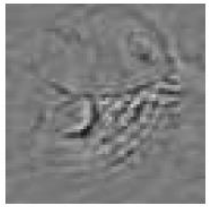
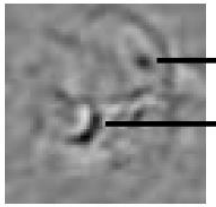
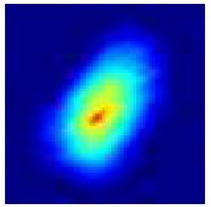
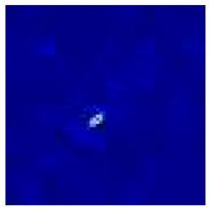
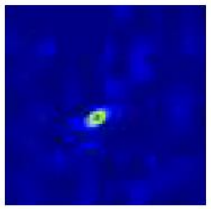
Slides from Jiri Matas 2016

How to Decide Classifier Weights, w?

Training image: $\mathbf{x} =$ 

Objective

 $\circledast\ \mathbf{w} =$ 

	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.**

Slides from Jiri Matas 2016

Minimum Output Sum of Squared Error (MOSSE) based Correlation Filter

Slides from Jiri Matas 2016

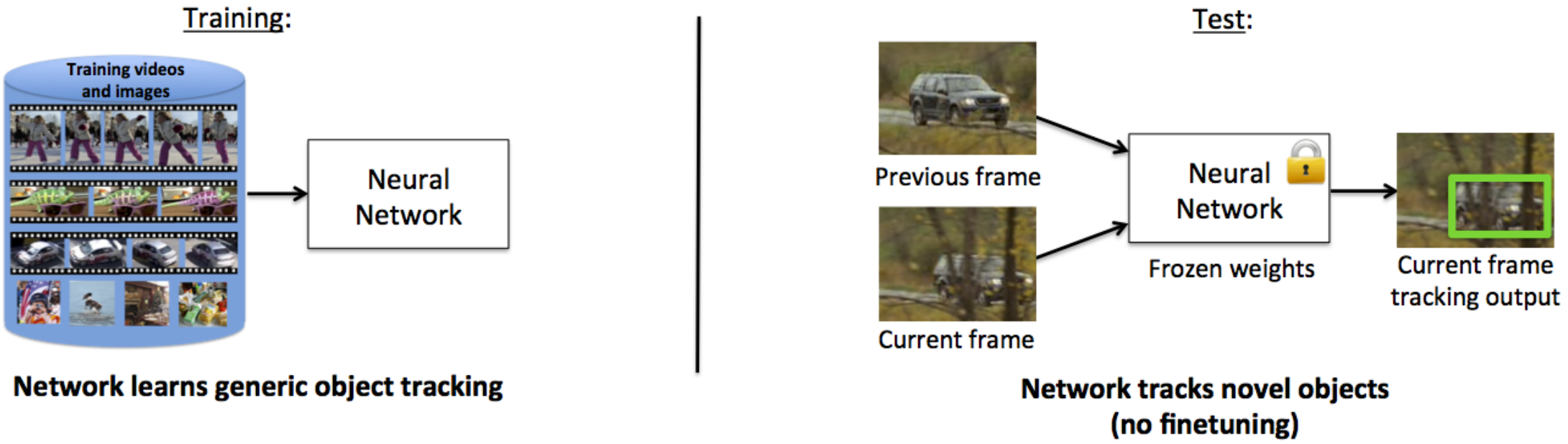
A Nice Summary for Tracking API in OpenCV

- https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

Deep Learning based Tracking

The Chronological Big Picture

1. The defining early work for tracking with Deep Learning began with GOTURN which proposes a Siamese CNN architecture and track via direct regression to predict the object bounding box.



Learning to Track at 100 FPS with Deep Regression Networks, David Held et al., 2016

The Chronological Big Picture

2. Following that, we had an explosion of deep trackers that employ a similar framework of Siamese CNN in differing ways. But the key idea stayed.
    1. E.g. Correlation Filter + Siamese CNN tracker
    2. E.g. Fully Convolutional Siamese tracker
    3. E.g. Triplet-loss Like Siamese CNN tracker ("distraction-aware")

3. There are also a noteworthy distinct class of approach that combines CNN + classical Kalman Filters around the same time (2017) with SORT, DeepSORT; still studied to some extent as of today (e.g. StrongSORT).

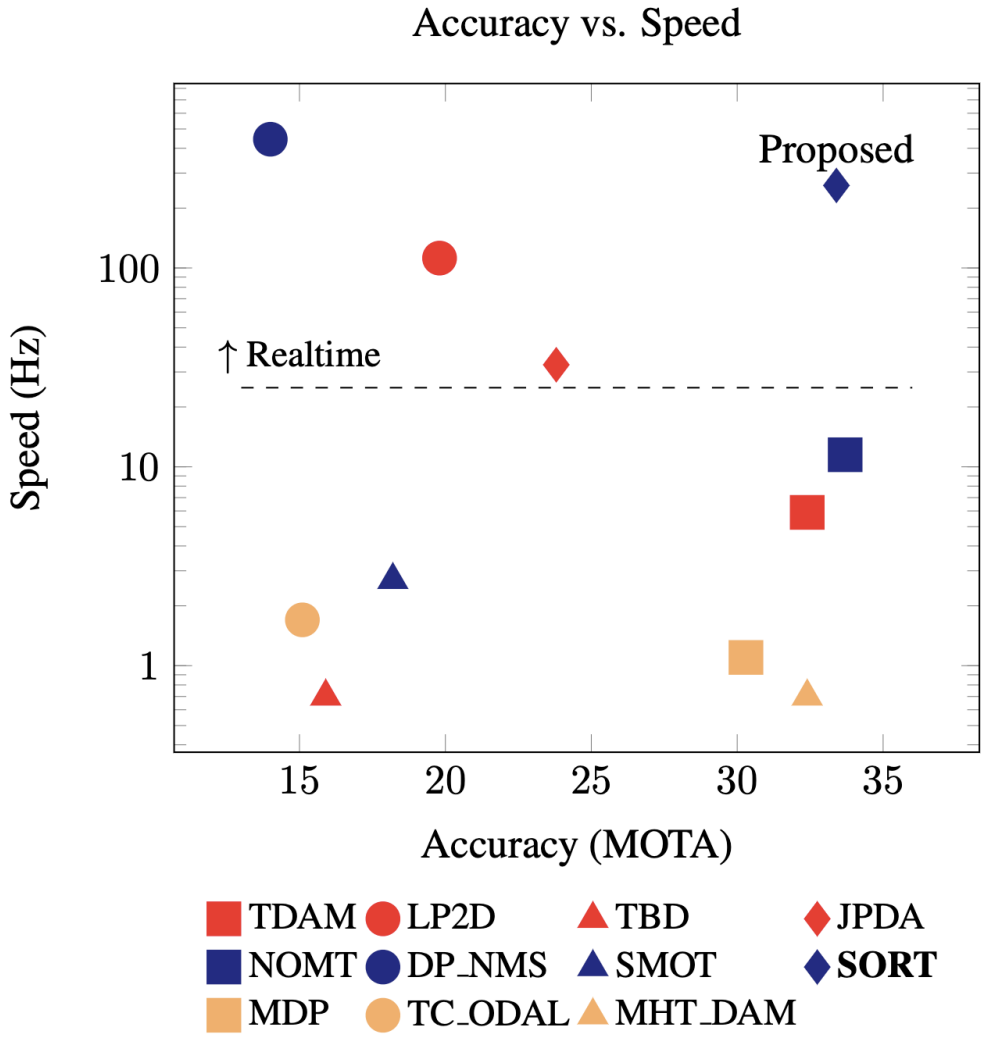
4. Then came attention and Transformers 🤖 (GNN is similar too), which kickstarted another "class" of approach until *today* (e.g. SUSHI, MotionTrack).

Overview of Methods

Let’s go over some key methods

1. SORT & StrongSORT
2. Detect to Track and Track to Detect
3. Self-Supervised Methods for Tracking

Simple Online & Realtime Tracking (SORT)



Key Ideas

1. Object tracking does not need to be complicated.

2. A track-by-detection framework – detect objects first, then assign.

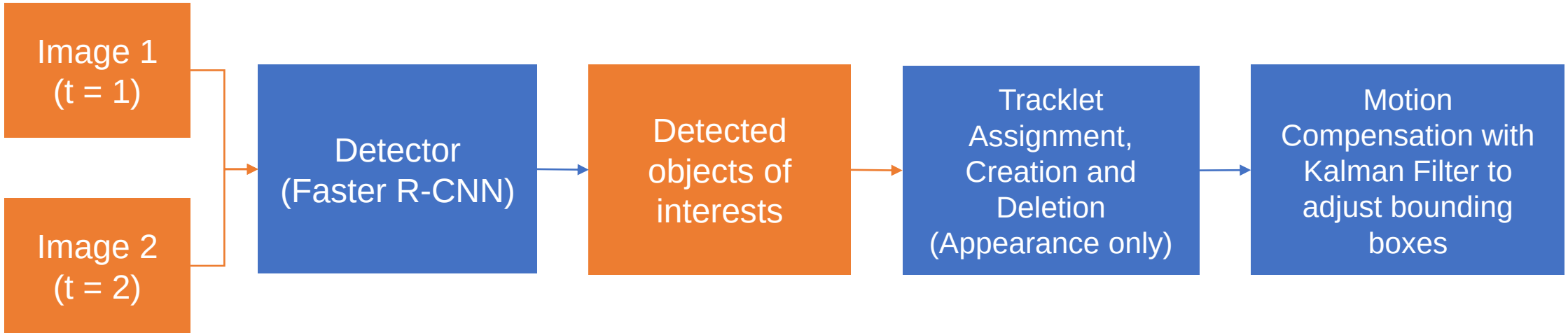
3. Key advantage of SORT is *speed!* Runs at ~260Hz / ~20x faster than baselines.

Simple Online & Realtime Tracking, Alex Bewley et al., 2016

Methodology of SORT

Key Ideas

1. Utilizes a pretrained Faster R-CNN instead of handcrafted descriptor.
2. For motion estimation, uses Kalman Filter for motion compensation (assumes fixed velocity; independent of camera motion)
3. For assignment, uses a simple appearance-matching schema of detect → match via IoU with Hungarian algorithm.
4. Finally, top it off with simple heuristics to create new tracklets and delete out-of-scene tracklets!



Simple Online & Realtime Tracking, Alex Bewley et al., 2016

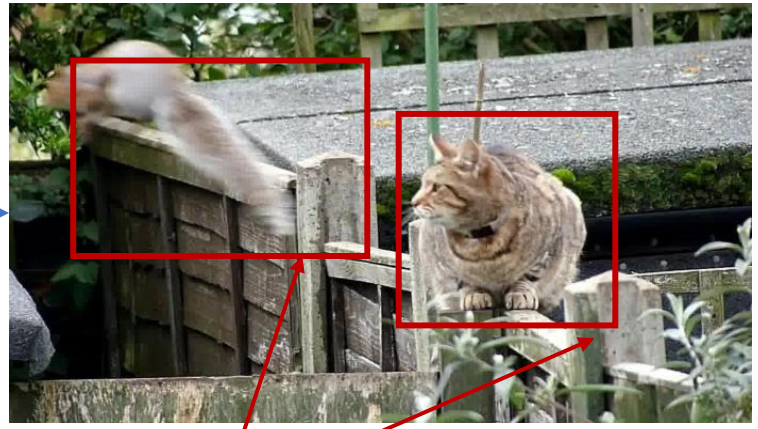
Methodology of SORT - Detection

Image t = 0

Image t = 1

Detector

Unassigned detections



Not that good either!

Simple Online & Realtime Tracking, Alex Bewley et al., 2016

Methodology of SORT – Assignment

Note: Tracklet creation and deletion not shown here

Image t = 0

Unassigned detections

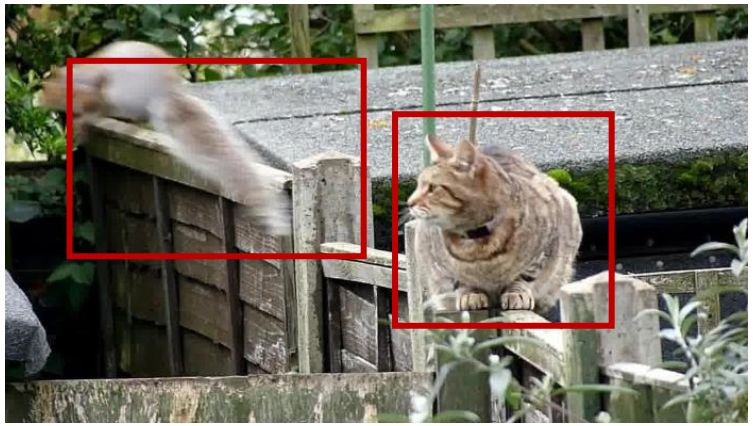


Image t = 1

Extract

Assigned detections

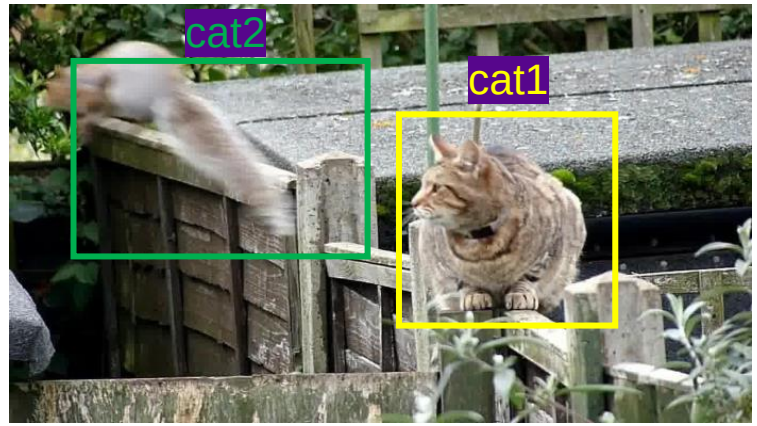


Image t = 1

Compute IoU Cost Matrix & assign via Hungarian Algorithm

Simple Online & Realtime Tracking, Alex Bewley et al., 2016

Methodology of SORT – Motion Compensation

Image t = 0

Bounding Box state

Assigned detections

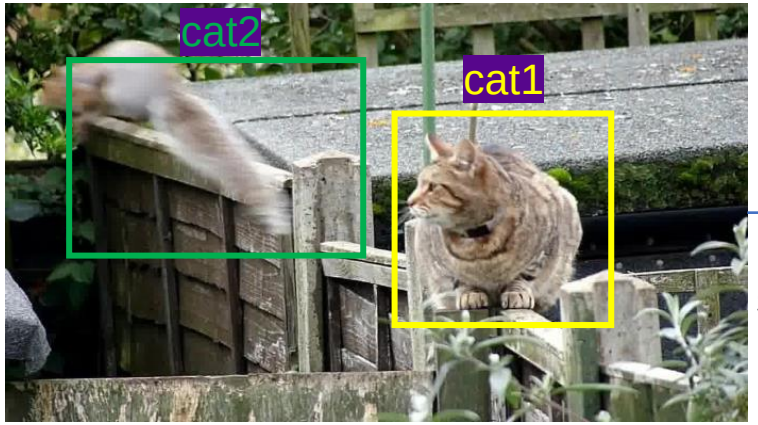


Image t = 1

Bounding Box state

Kalman Filter

Adjusted detections

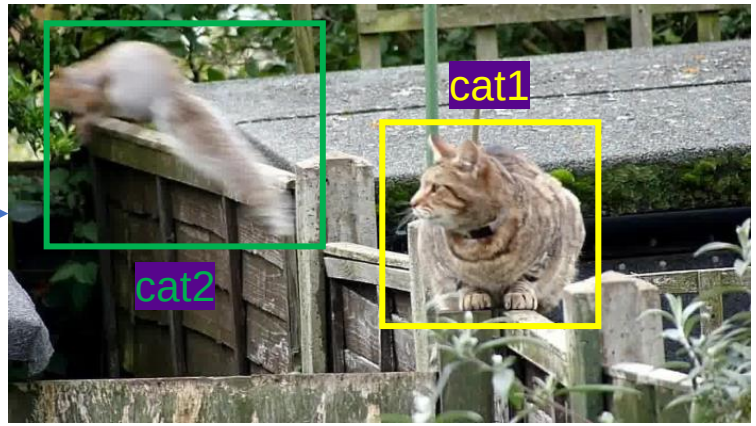


Image t = 1

Simple Online & Realtime Tracking, Alex Bewley et al., 2016

A Demo

Works pretty well!

The formulation naturally handles **short term occlusion** as well thanks to the motion compensation & appearance-based assignment algorithm.

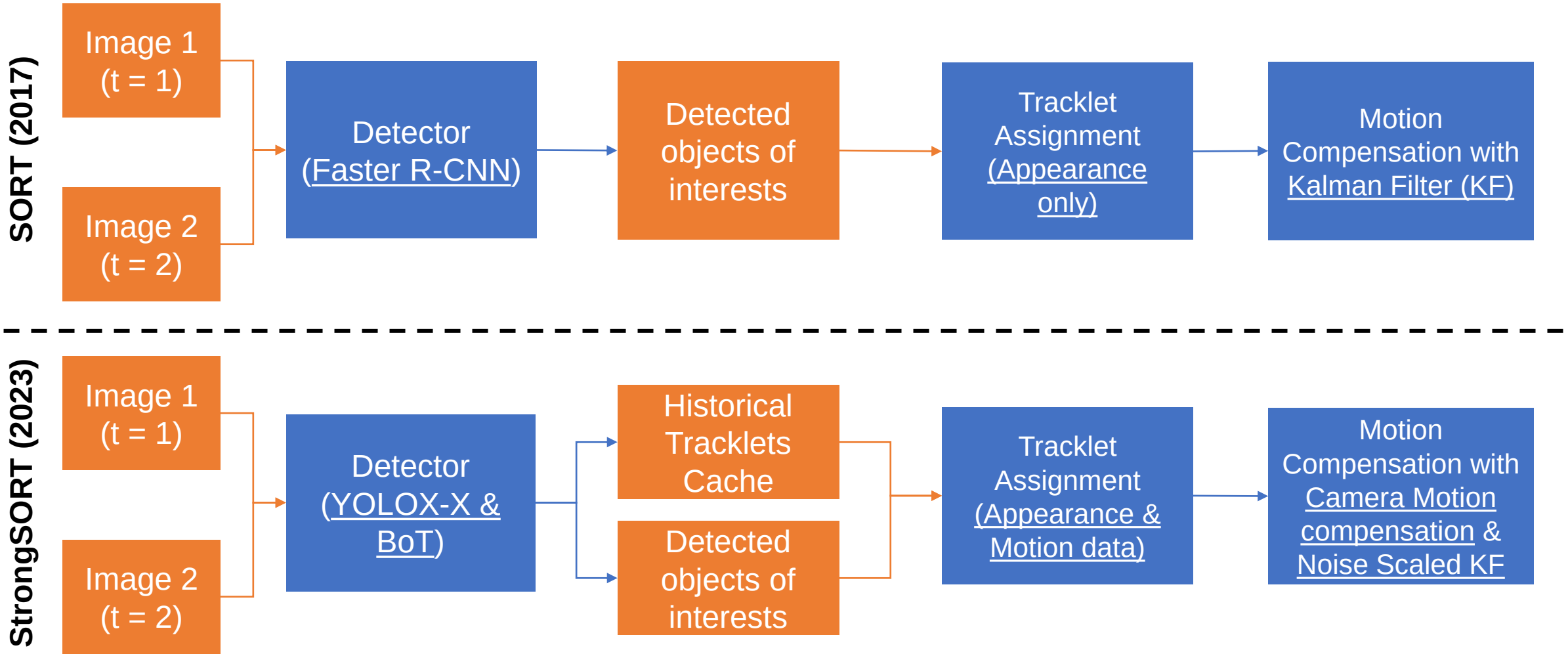
But! It does not handle long term occlusion / long term tracking well! ☹

Can we do better?

Simple Online & Realtime Tracking, Alex Bewley et al., 2016

StrongSORT vs SORT

An improved SORT algorithm!



StrongSORT: Make DeepSORT Great Again, Yunhao Du et al., 2023

StrongSORT Overview

1. Summary of improvements
    1. Faster R-CNN → YOLOX-X and BoT
    2. Simple feature bank → Feature bank augmented via Exponential Moving Average (EMA)
    3. Vanilla Kalman Filter → Noise Scaled Adaptive (NSA) Kalman Filter
    4. Feature matching loss: Photometric only → Photometric + Motion
    5. Added camera motion compensation via Correlation Coefficient Maximization

2. Also introduced 2 new modules to handle (StrongSORT++)
    1. *Missing associations* via simple DL classifier – dubbed AFLink
    2. *Trajectory gaps* via Gaussian modelling – dubbed GSI

Performance of StrongSORT

TABLE IV: Comparison with state-of-the-art MOT methods on the MOT17 test set. "*" represents our reproduced version. "(w/o LI)" means abandoning the offline linear interpolation procedure. The two best results for each metric are bolded and highlighted in red and blue.

mode	Method	Ref.	HOTA(↑)	IDF1(↑)	MOTA(↑)	AssA(↑)	DetA(↑)	IDs(↓)	FPS(↑)
online	SORT [3]	ICIP2016	34.0	39.8	43.1	31.8	37.0	4,852	**143.3**
	MTDF [15]	TMM2019	37.7	45.2	49.6	34.5	42.0	5,567	1.2
	DeepMOT [57]	CVPR2020	42.4	53.8	53.7	42.7	42.5	1,947	4.9
	ISEHDADH [8]	TMM2019	-	-	54.5	-	-	3,010	3.6
	Tracktor++ [1]	ICCV2019	44.8	55.1	56.3	45.1	44.9	1,987	1.5
	TubeTK [33]	CVPR2020	48.0	58.6	63.0	45.1	51.4	4,137	3.0
	CRF-MOT [17]	TMM2022	-	60.4	58.9	-	-	2,544	-
	CenterTrack [66]	ECCV2020	52.2	64.7	67.8	51.0	53.8	3,039	3.8
	TransTrack [45]	arxiv2020	54.1	63.5	75.2	47.9	61.6	3,603	59.2
	PermaTrack [46]	ICCV2021	55.5	68.9	73.8	53.1	58.5	3,699	11.9
	CSTrack [28]	TIP2022	59.3	72.6	74.9	57.9	61.1	3,567	15.8
	FairMOT [64]	IJCV2021	59.3	72.3	73.7	58.0	60.9	3,303	25.9
	CrowdTrack [42]	AVSS2021	60.3	73.6	75.6	59.3	61.5	2,544	**140.8**
	CorrTracker [51]	CVPR2021	60.7	73.6	76.5	58.9	62.9	3,369	15.6
	RelationTrack [59]	TMM2022	61.0	74.7	73.8	61.5	60.6	**1,374**	8.5
	OC-SORT* (w/o LI) [7]	arxiv2022	61.7	76.2	76.0	62.0	61.6	2,199	29.0
	ByteTrack* (w/o LI) [63]	ECCV2022	**62.8**	**77.2**	**78.9**	**62.2**	**63.8**	2,310	29.6
	DeepSORT* [55]	ICIP2017	61.2	74.5	78.0	59.7	63.1	1,821	13.8
	StrongSORT	ours	**63.5**	**78.5**	**78.3**	**63.7**	**63.6**	**1,446**	7.5
offline	TPM [35]	PR2020	41.5	52.6	54.2	40.9	42.5	1,824	0.8
	MPNTrack [6]	CVPR2020	49.0	61.7	58.8	51.1	47.3	**1,185**	6.5
	TBooster [49]	TMM2022	50.5	63.3	61.5	52.0	49.2	2,478	6.9
	MAT [20]	NC2022	56.0	69.2	67.1	57.2	55.1	1,279	11.5
	ReMOT [58]	IVC2021	59.7	72.0	77.0	57.1	62.8	2,853	1.8
	MAATrack [43]	WACVw2022	62.0	75.9	79.4	60.2	64.2	1,452	**189.1**
	OC-SORT [7]	arxiv2022	63.2	77.5	78.0	63.4	63.2	1,950	29.0
	ByteTrack* [63]	ECCV2022	63.2	77.4	**79.7**	62.3	**64.4**	2,253	**29.6**
	StrongSORT+	ours	**63.7**	**79.0**	78.3	**64.1**	63.6	1,401	7.4
	StrongSORT++	ours	**64.4**	**79.5**	**79.6**	**64.4**	**64.6**	**1,194**	7.1

1. Result is pretty good as far as the object tracking literature is concerned.
2. Though because of its complexity, StrongSORT is not very fast! (Unlike SORT)

StrongSORT: Make DeepSORT Great Again, Yunhao Du et al., 2023

Detect to Track and Track to Detect

1. Simultaneous detection and tracking using a multi-task objective for frame-based object detection and across-frame track regression.

2. Introduces correlation features that represent object co-occurrences across time to aid the CNN during tracking.

3. Links the frame level detections based on our across-frame tracklets to produce high accuracy detections at the video level

4. Simple but effective idea! Nicely encodes the prior of *persistence across time*.

https://arxiv.org/pdf/1710.03958.pdf

Backbone network : R-FCN

- R-FCNs are convolutional networks used for object detection.
- It is faster than traditional object detection network such as R-CNN as its region-based feature maps are independent of region of interests.
- The network is lighter because it is fully convolutional and has no FC layers.
- The output of the network is a set of bounding coordinates and class labels for detected objects.

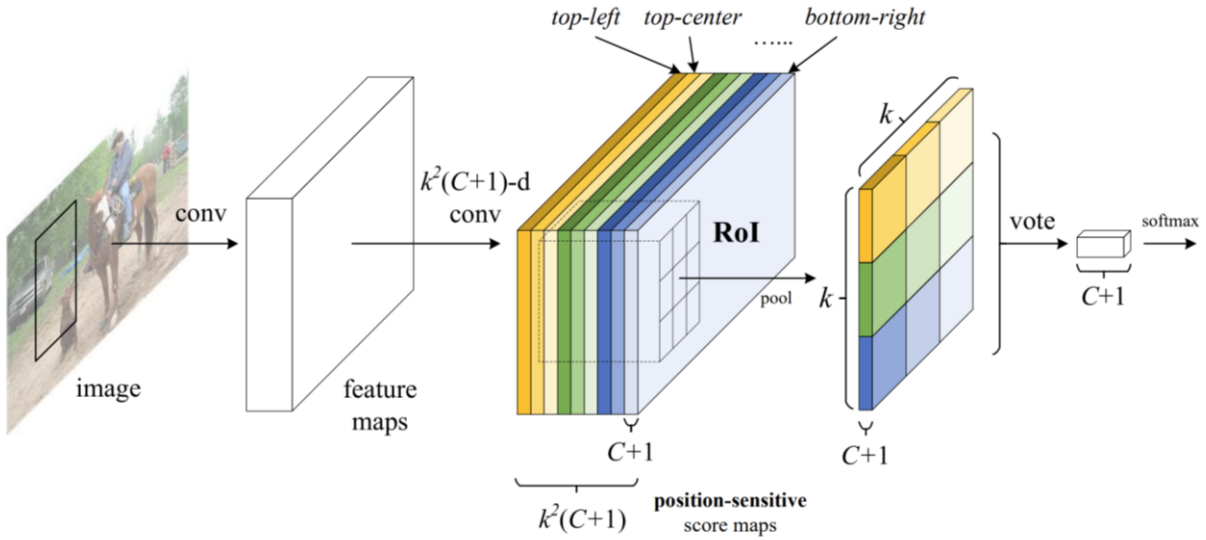
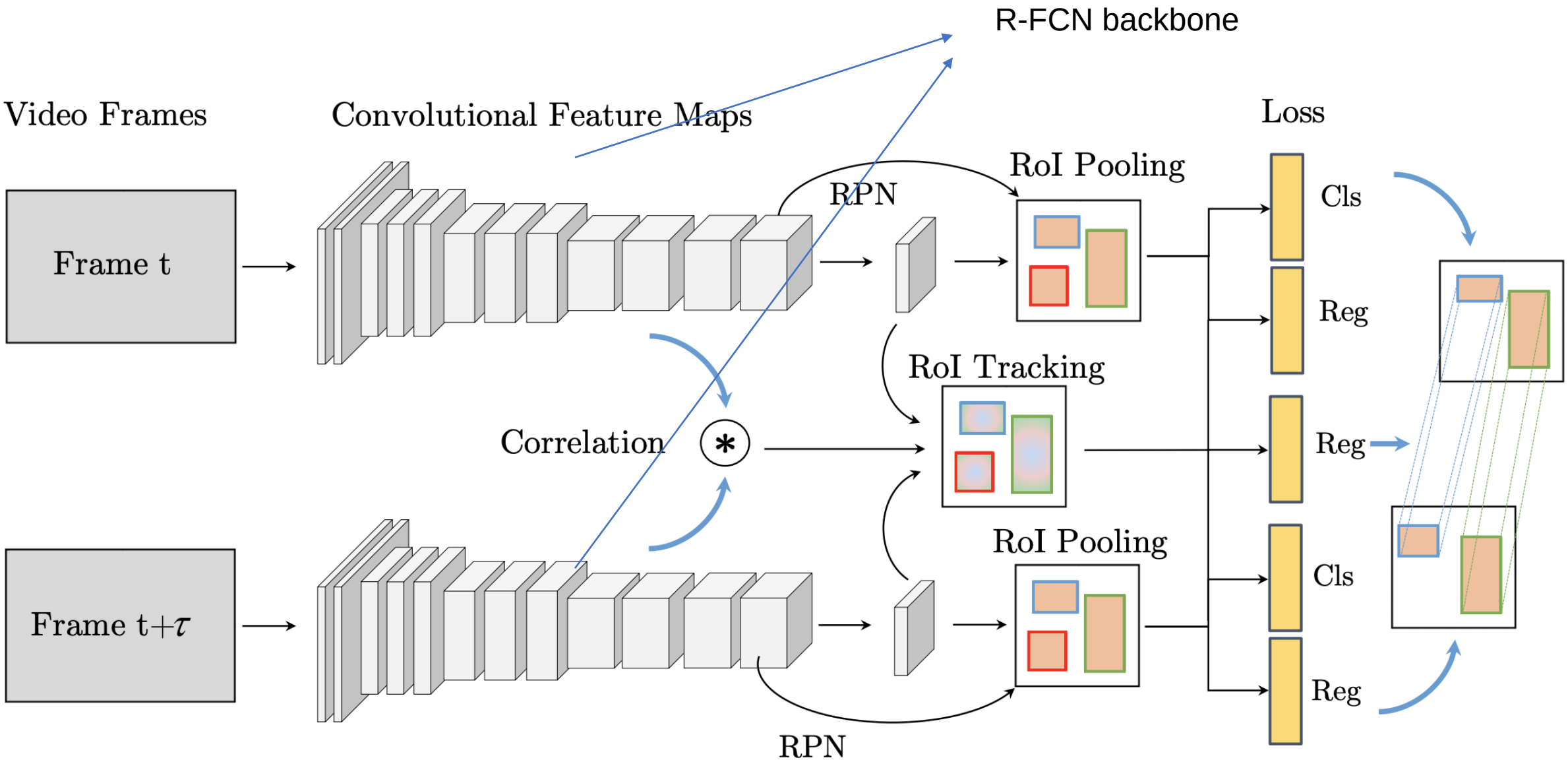


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

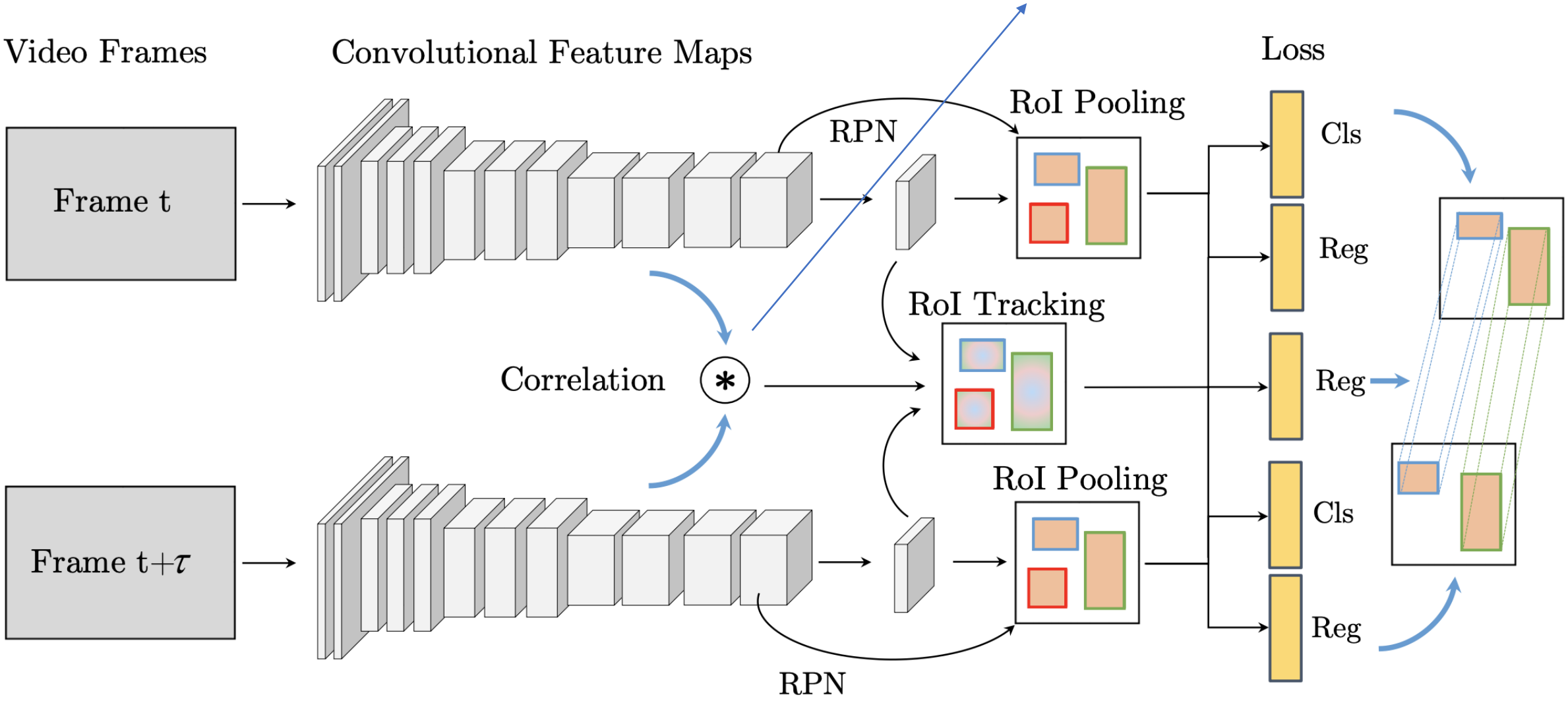
https://arxiv.org/pdf/1605.06409.pdf

Overall Architecture

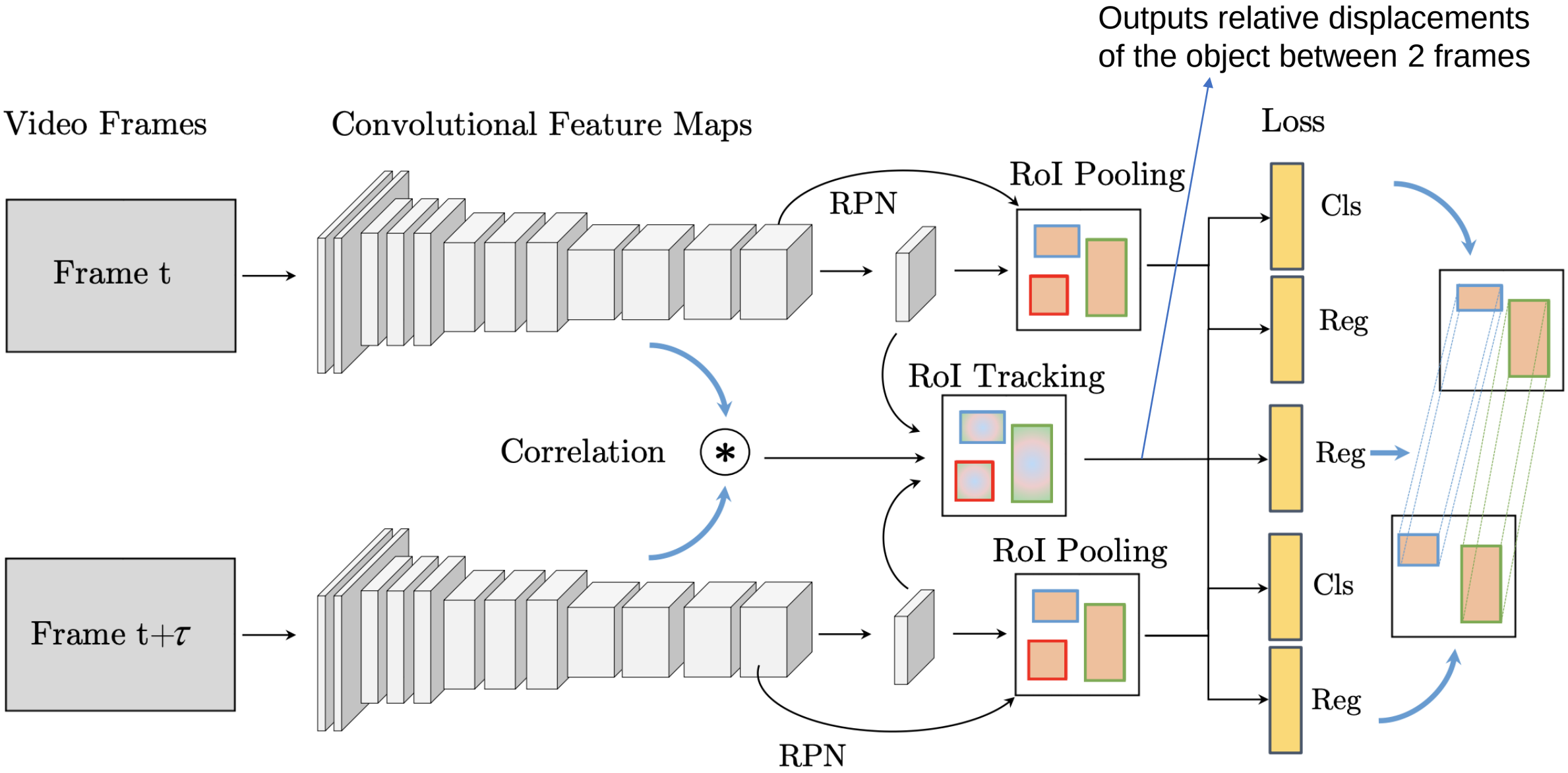


Overall Architecture

Correlation of features represent object co-occurrences across time



Overall Architecture



Objective function

$$L(\{p_i\},\{b_i\},\{\Delta_i\}) = \frac{1}{N}\sum_{i=1}^{N} L_{cls}(p_{i,c^*}) \longrightarrow$$

Classification loss (foreground/background)

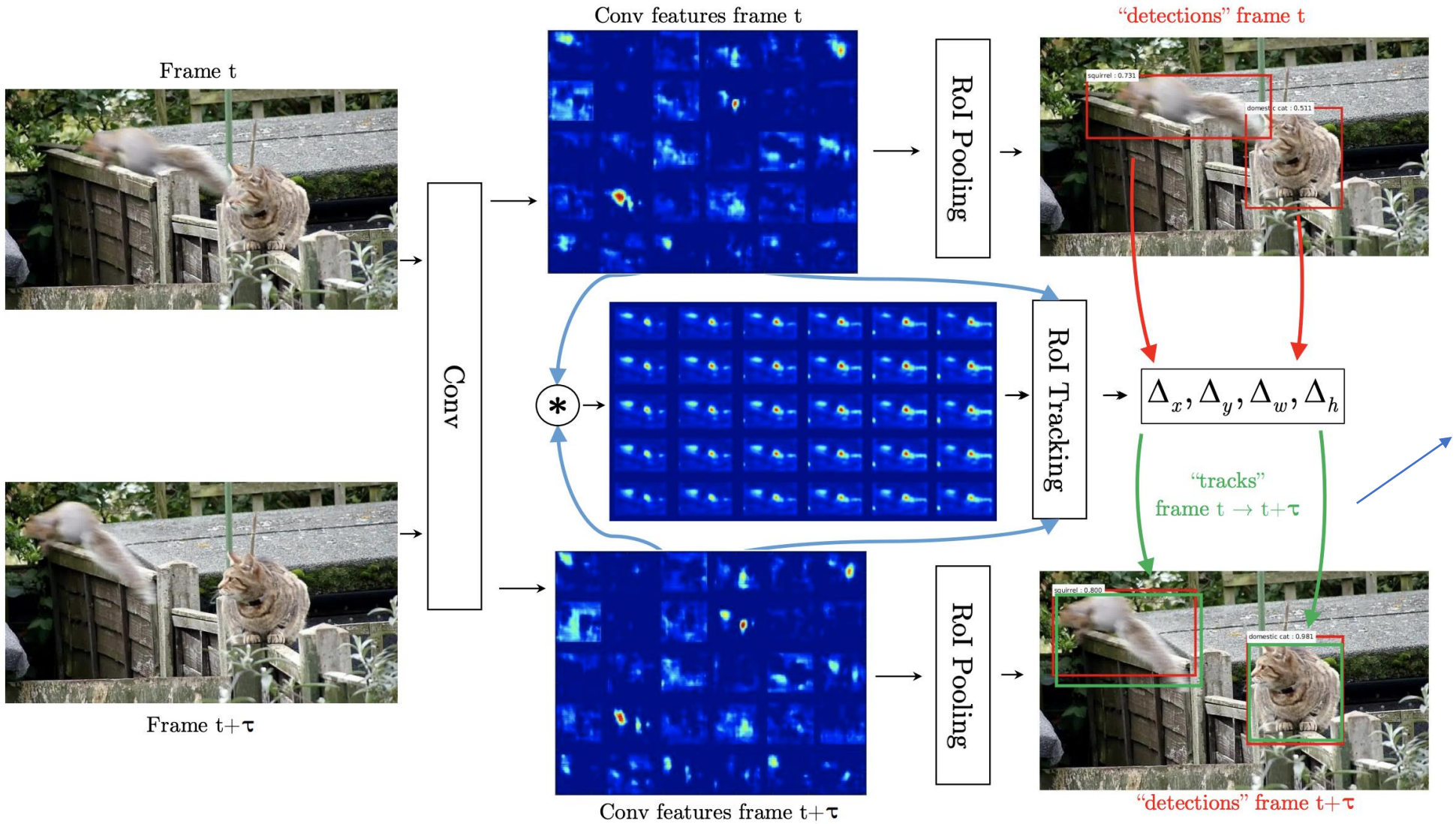
$$+\lambda\frac{1}{N_{fg}}\sum_{i=1}^{N}[c_i^* > 0]L_{reg}(b_i, b_i^*) \longrightarrow$$

Bounding box regression cost (for both frames)

$$+\lambda\frac{1}{N_{tra}}\sum_{i=1}^{N_{tra}} L_{tra}(\Delta_i^{t+\tau}, \Delta_i^{*,t+\tau}). \longrightarrow$$

Regression offsets

Inference



Bounding boxes for t+τ is computed using boxes for frame t and the predicted offsets

Self-supervised Tracking

- To learn a feature extractor without any supervision

- Self-supervised learning via cycle consistency

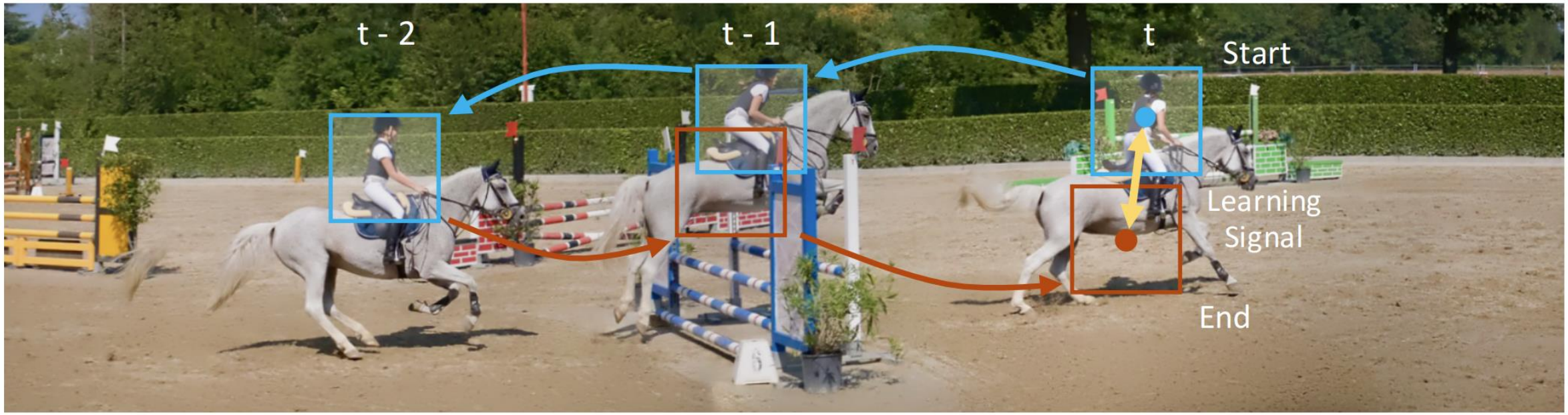


Figure 2: **A Cycle in Time.** Given a video, tracking along the sequence formed by a cycle in time can be self-supervised: the target is simply the beginning of the cycle. The yellow arrow between the start and end represents the differentiable learning signal.

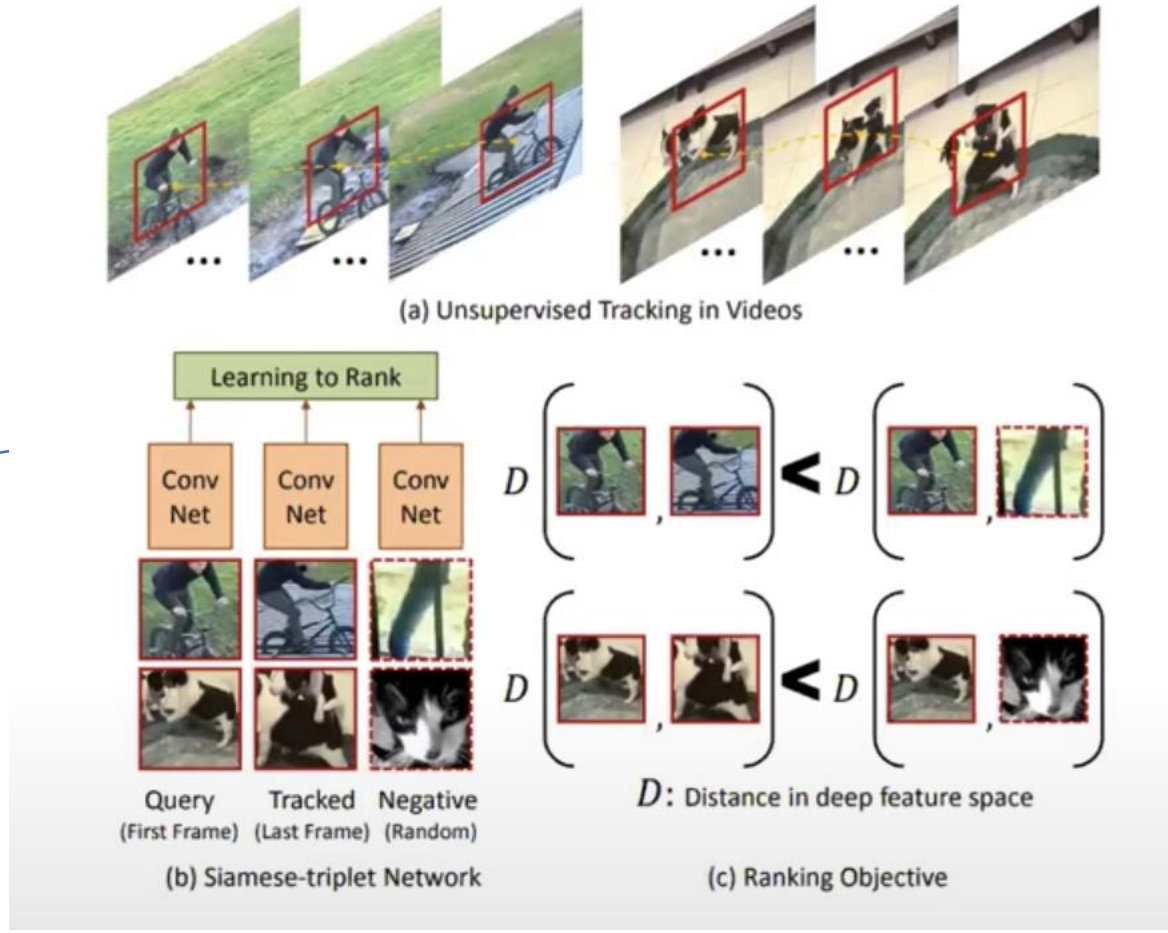
Previous Self-Supervised Correspondence Learning

- Tracking-based Similarity Learning

(a) Prepare Query (Positive & Negative)

(b) Obtain Positive via Hand-craft tracker

(c) Use Siamese-Triplet Network

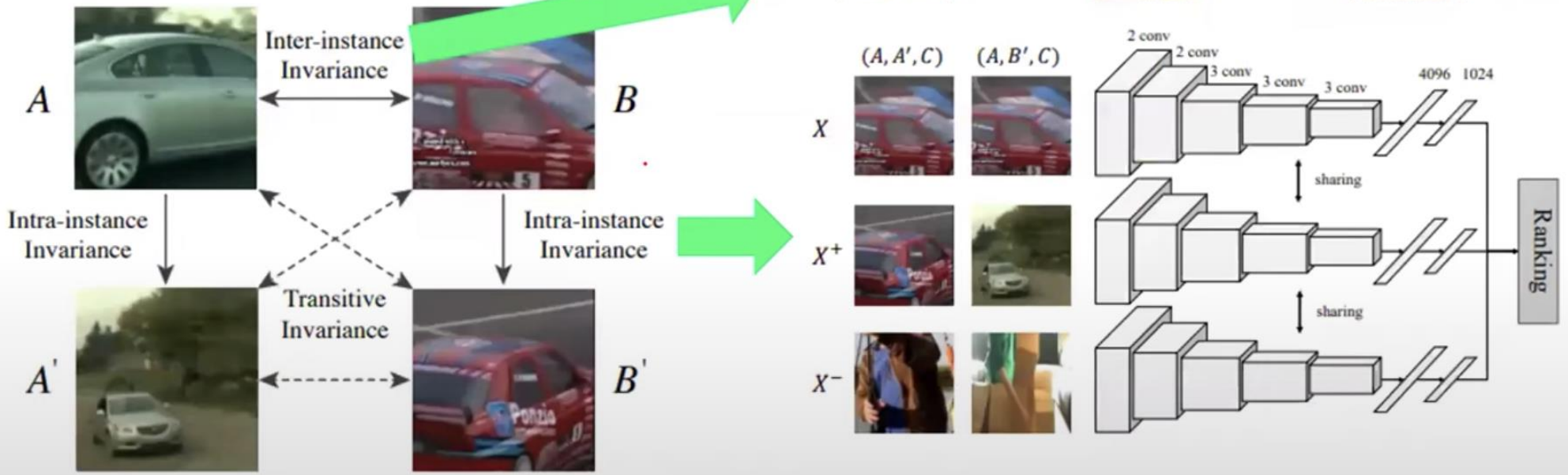


https://www.youtube.com/watch?v=5g8-8Hz5x08

https://arxiv.org/pdf/1903.07593.pdf

Previous Self-Supervised Correspondence Learning

- Tracking-based Similarity Learning
- Transitive Invariance Learning



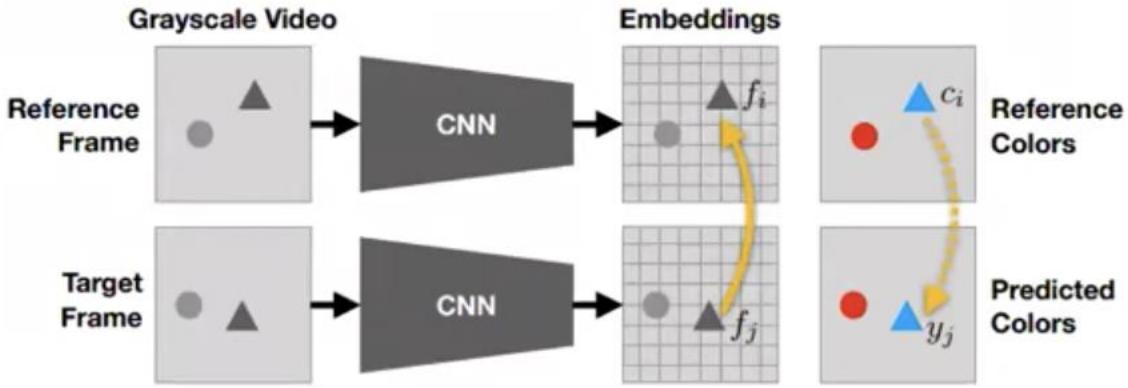
X. Wang, K. He, and A. Gupta. Transitive invariance for selfsupervised visual representation learning. In ICCV, 2017

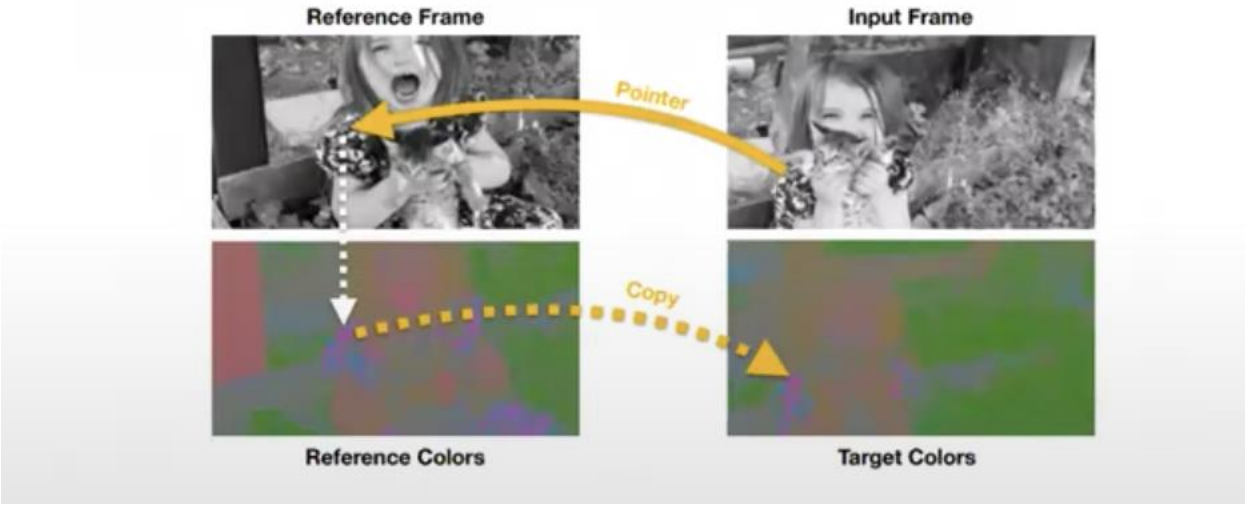
https://arxiv.org/pdf/1708.02901.pdf

Previous Self-Supervised Correspondence Learning

- Tracking-based Similarity Learning
- Transitive Invariance Learning
- Colorization

- (a) Find similar features
- (b) Copy colors from reference frame
- (c) Learn CNN using colorization loss

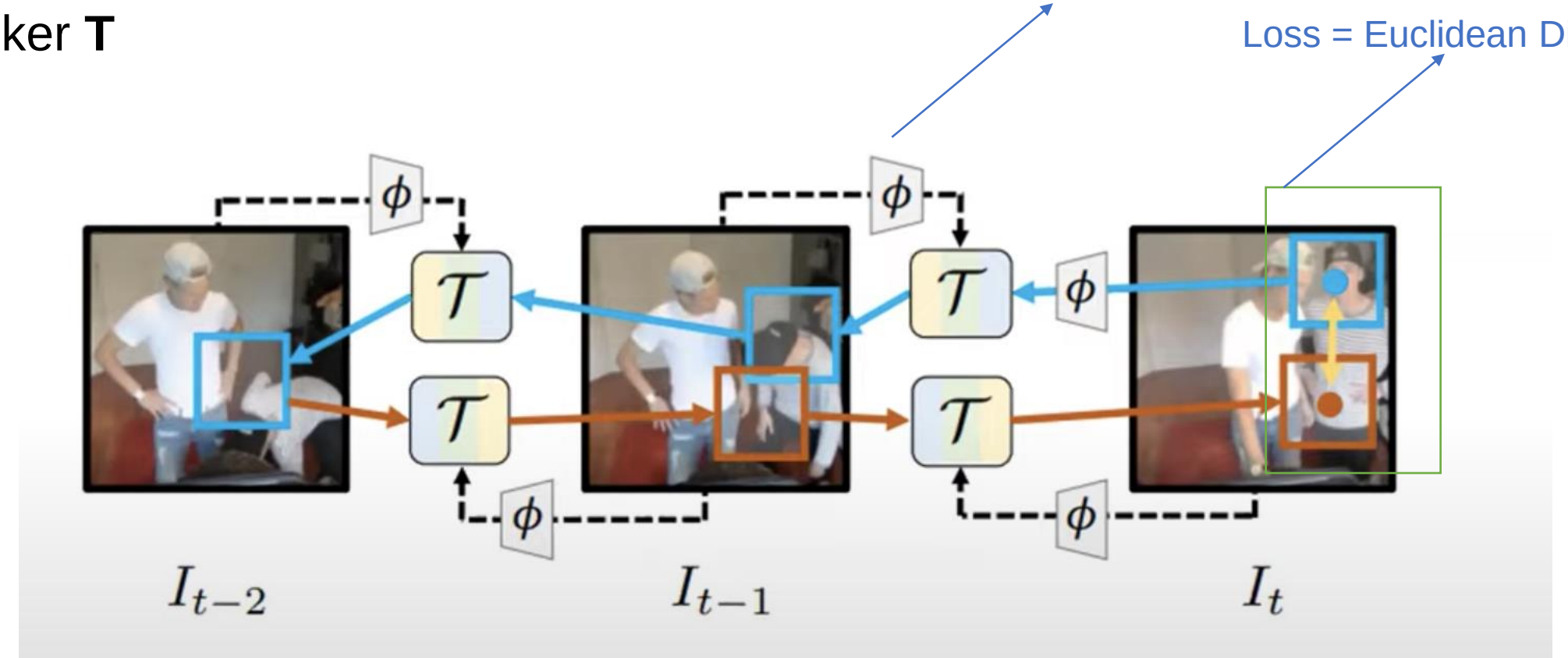




https://arxiv.org/pdf/1806.09594.pdf

Cycle-in-Time

- Feature extractor **Φ**
- Tracker **T**



https://arxiv.org/pdf/1903.07593.pdf

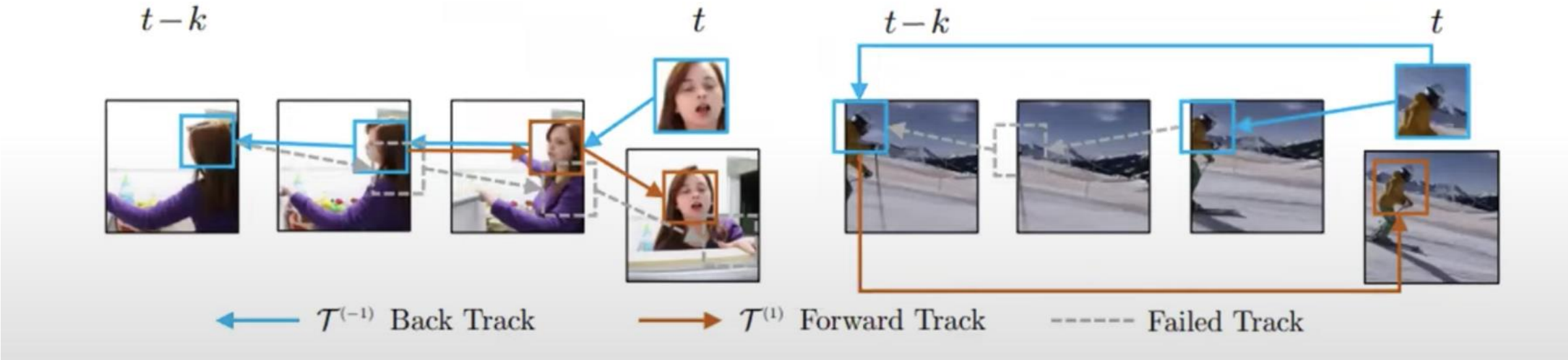
Multiple Cycles / Skip Cycle

- Multiple cycles for variation
- Skip cycle for occlusion

Mean region similarity

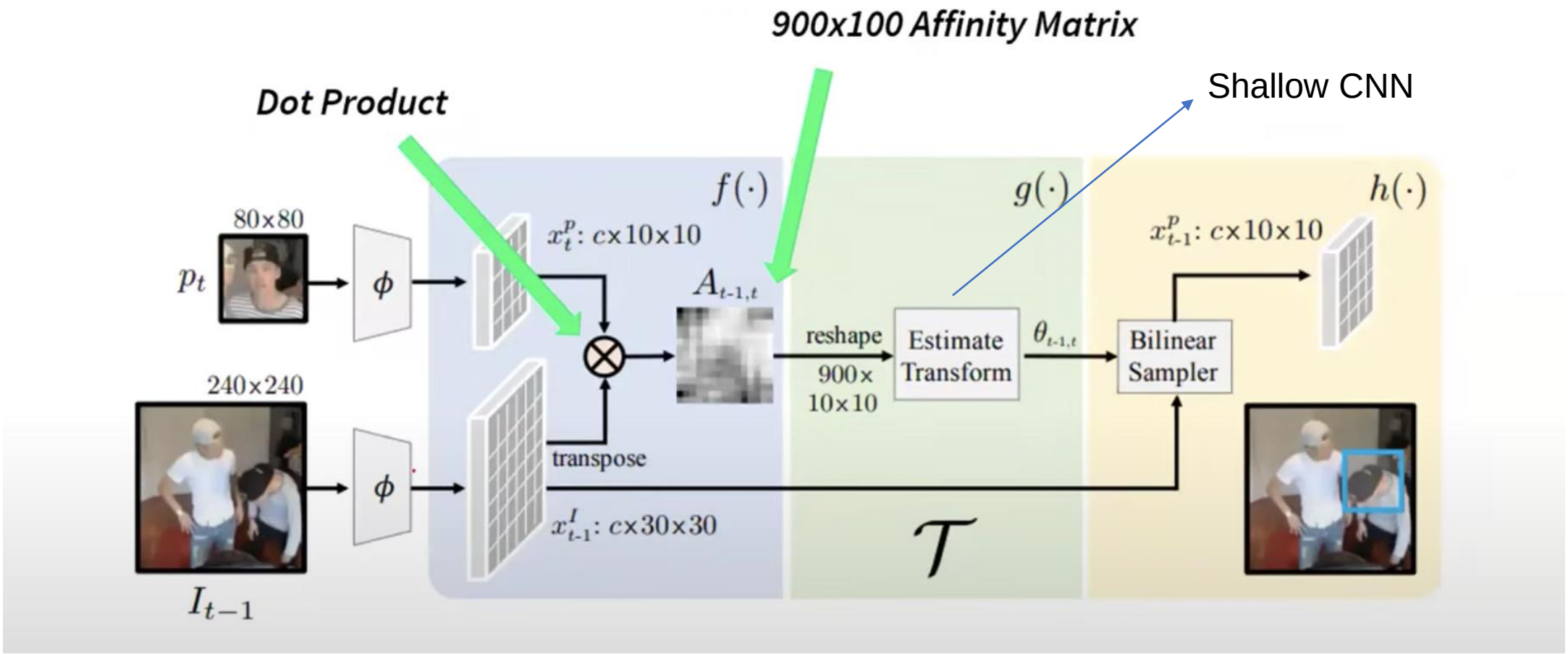
Contour based accuracy

Experiment	$\mathcal{J}$(Mean)	$\mathcal{F}$(Mean)
Ours	41.9	39.4
Ours without Skip Cycles	39.5	37.9



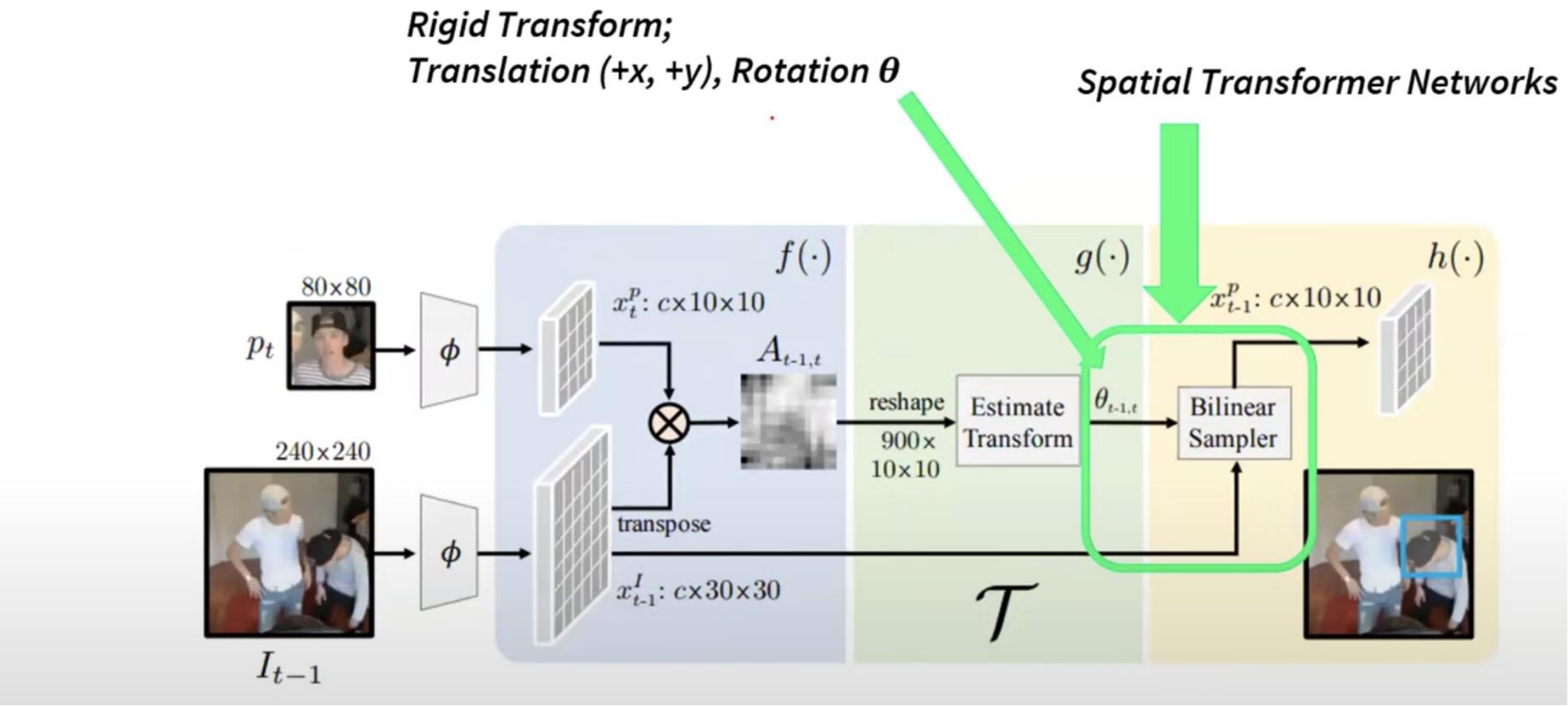
https://arxiv.org/pdf/1903.07593.pdf

Differentiable Tracker



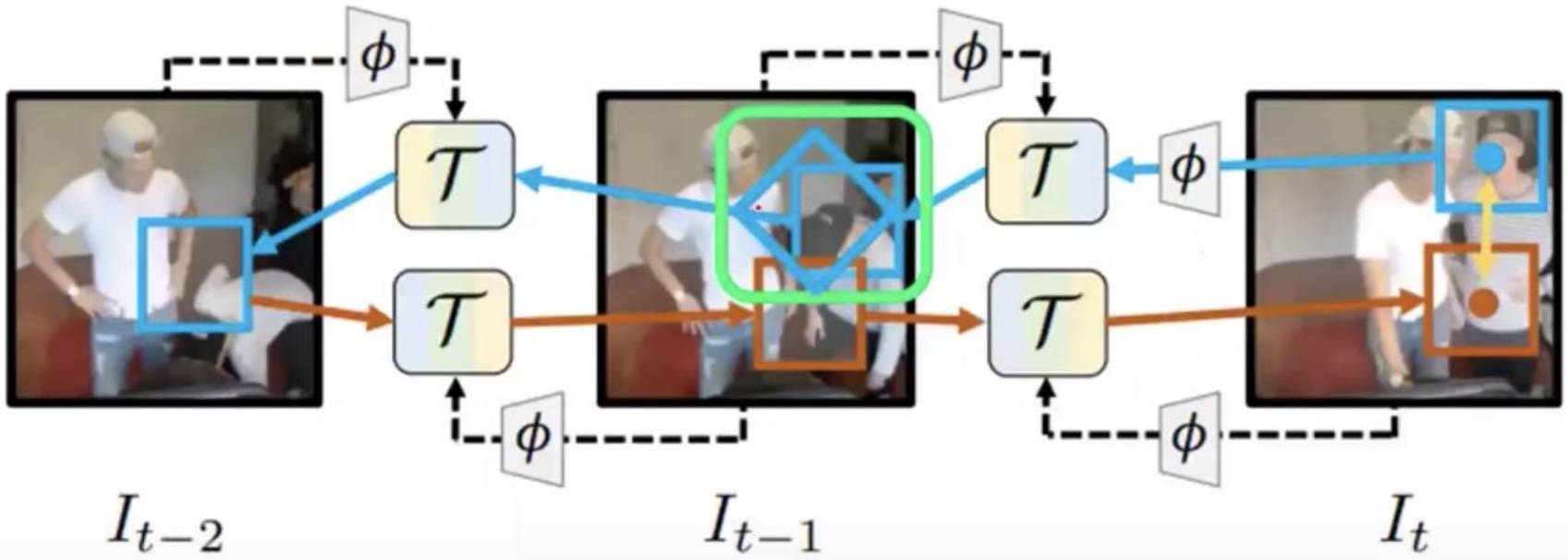
https://arxiv.org/pdf/1903.07593.pdf

Differentiable Tracker



https://arxiv.org/pdf/1903.07593.pdf

Training



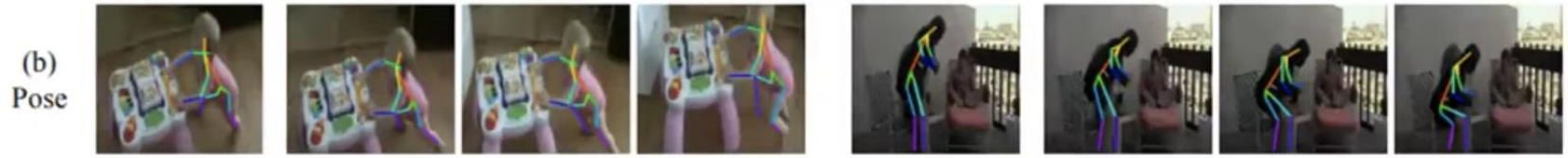
https://arxiv.org/pdf/1903.07593.pdf

Results : Video Object Segmentation

model	Supervised	$\mathcal{J}$(Mean)	$\mathcal{F}$(Mean)
Identity		22.1	23.6
Random Weights (ResNet-50)		12.4	12.5
Optical Flow (FlowNet2) [22]		26.7	25.2
SIFT Flow [39]		33.0	35.0
Transitive Inv. [74]		32.0	26.8
DeepCluster [8]		37.5	33.2
Video Colorization [69]		34.6	32.7
Ours (ResNet-18)		40.1	38.3
Ours (ResNet-50)		**41.9**	**39.4**
ImageNet (ResNet-50) [18]	✓	50.3	49.0
Fully Supervised [81, 7]	✓	55.1	62.1

https://arxiv.org/pdf/1903.07593.pdf

Results: Pose Propagation



model	Supervised	PCK@.1	PCK@.2
Identity		43.1	64.5
Optical Flow (FlowNet2) [22]		45.2	62.9
SIFT Flow [39]		49.0	68.6
Transitive Inv. [74]		43.9	67.0
DeepCluster [8]		43.2	66.9
Video Colorization [69]		45.2	69.6
Ours (ResNet-18)		57.3	78.1
Ours (ResNet-50)		**57.7**	**78.5**
ImageNet (ResNet-50) [18]	✓	58.4	78.4
Fully Supervised [59]	✓	68.7	92.1

Probability of Correct Keypoint

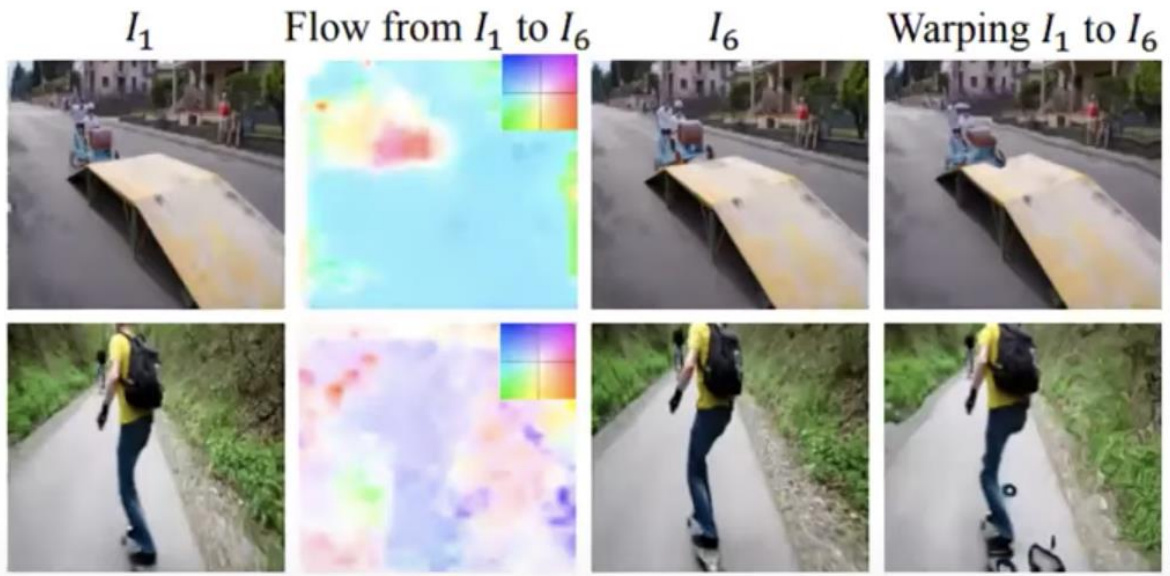
https://arxiv.org/pdf/1903.07593.pdf

Results: Human Part Labels Propagation

model	Supervised	mIoU	AP^r_{vol}
Identity		13.6	4.0
Optical Flow (FlowNet2) [22]		16.1	8.3
SIFT Flow [39]		21.3	10.5
Transitive Inv. [74]		19.4	5.0
DeepCluster [8]		21.8	8.1
Ours (ResNet-50)		**28.9**	**15.6**
ImageNet (ResNet-50) [18]	✓	34.7	16.1
Fully Supervised [85]	✓	37.9	24.1

https://arxiv.org/pdf/1903.07593.pdf

Results: Long-range Optical Flow



model	5-F	10-F
Identity	82.0	97.7
Optical Flow (FlowNet2) [22]	62.4	90.3
ImageNet (ResNet-50) [18]	64.0	79.2
Ours (ResNet-50)	**60.4**	**76.4**

https://arxiv.org/pdf/1903.07593.pdf

Next week

- Segmentation
  - + Graph-based segmentation
  - + Semantic segmentation
  - + Instance segmentation
  - + Panoptic segmentation
- 3D Deep Learning (Recognition and Segmentation)
  - + VoxNet
  - + PointNet
  - + MVCNN
  - + FoldingNet

- + : know the concept

References for next week

- Sz: Ch 6.4

- Maturana, Daniel, and Sebastian Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition." *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015.
- Su, Hang, et al. "Multi-view convolutional neural networks for 3d shape recognition." *Proceedings of the IEEE international conference on computer vision*. 2015.
- Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652-660).
- Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 206-215).