2D Visual Object Tracking

I. Karakostas, V. Mygdalis, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr

Version 3.8.4

Artificial Intelligence & Information Analysis Lab

2D Object Tracking

- **Introduction**
- Prediction in object tracking
- Feature Point based trackers
- Region similarity trackers
- Correlation trackers
- Object Detection Performance Metrics

2D Object Tracking

- Video tracking is the process of locating a moving object (or multiple objects) over time using a camera
- Variety of uses:
  - Human-computer interaction;
  - Security and surveillance;
  - Video communication and compression;
  - Traffic control;
  - Medical imaging.

Target/object examples

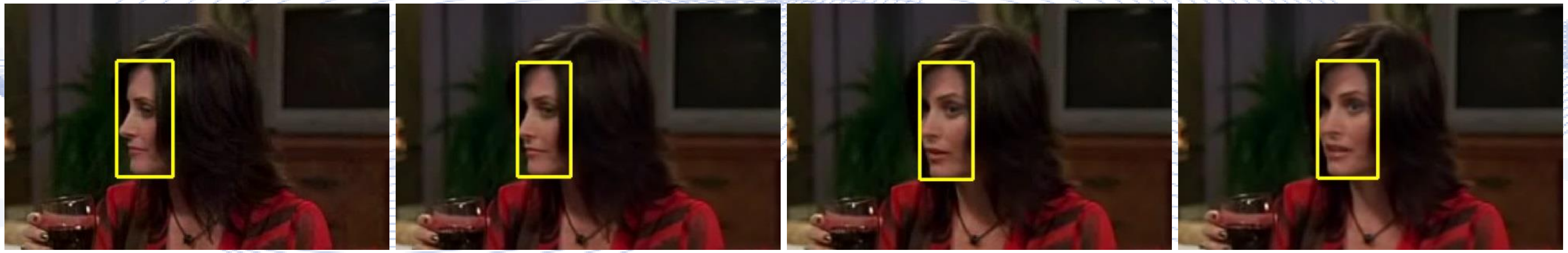
- Athletes, boats, bicycles.

Artificial Intelligence & Information Analysis Lab

2D Visual Object Tracking

- Problem statement:
  - To track a target/object (e.g. human face) image in each video frame and localize its ***Region-Of-Interest*** (***ROI***).
  - To track the detected object over the video frames.

1st frame 6th frame 11th frame 16th frame



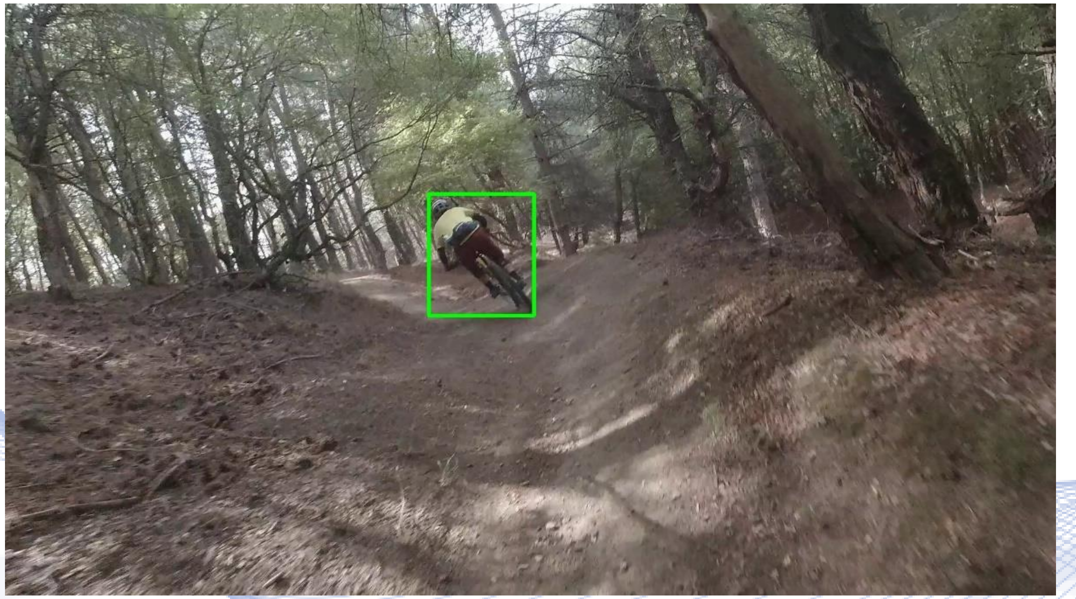
Artificial Intelligence & Information Analysis Lab

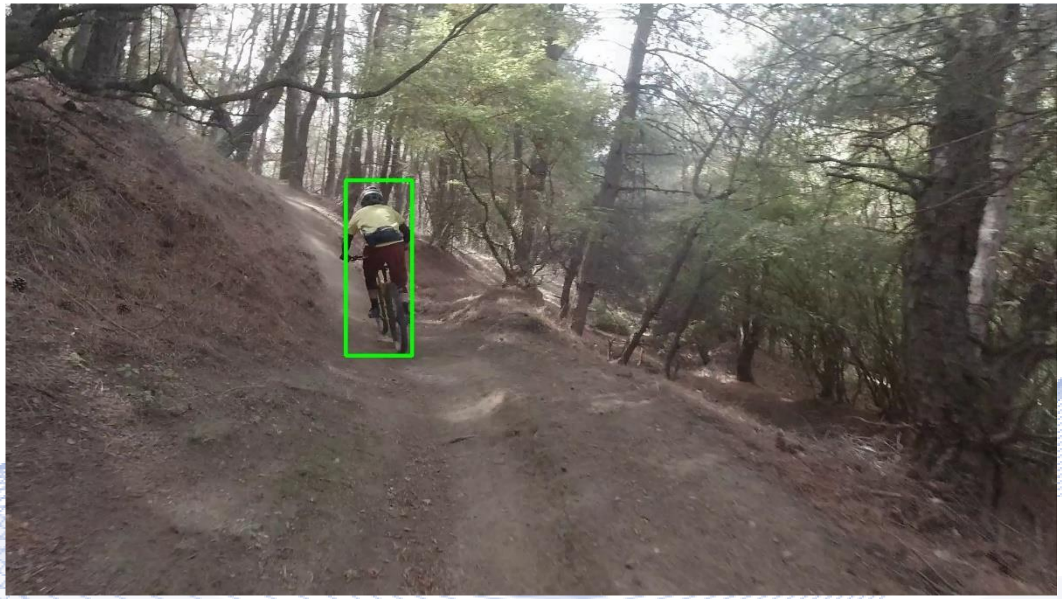
2D Object Tracking

- ROI is typically a bounding box at time t defined by:
  - its center/size parameter vector $[x_c, y_c, w, h]^T$ or
  - the upper left lower right rectangle coordinates $[x_l, y_l, x_r, y_r]$.
- ***Object ROI center***: $\mathbf{c} = [x_c, y_c]^T$.
- ***Object trajectory***: object ROI center coordinates over time.
- ***Moving region***: a series of tracked object ROIs over time.
- ***Object instance***: object region ROI plus other info.

Artificial Intelligence & Information Analysis Lab

2D Object Tracking





Object Region-Of-Interest (ROI).

2D Object Tracking

- ***2D visual object tracking*** is performed ***on the image plane***:
  - object ROI coordinates and trajectory are defined in (x, y, t) image plane coordinates (typically in pixels, sec).
- 2D visual object racking associates each detected object ROI in the current video frame with one in the next video frame.

2D Object Tracking

- ***3D object tracking*** is performed on a world coordinate system: (X, Y, Z, t) (in meters, sec).
- ***3D object following*** is a control problem, ensuring a vehicle follows a physical object moving in a world coordinate system: (X, Y, Z, t).

Artificial Intelligence & Information Analysis Lab

2D Object tracking requirements

In order to track a moving object, a tracker has to confront:

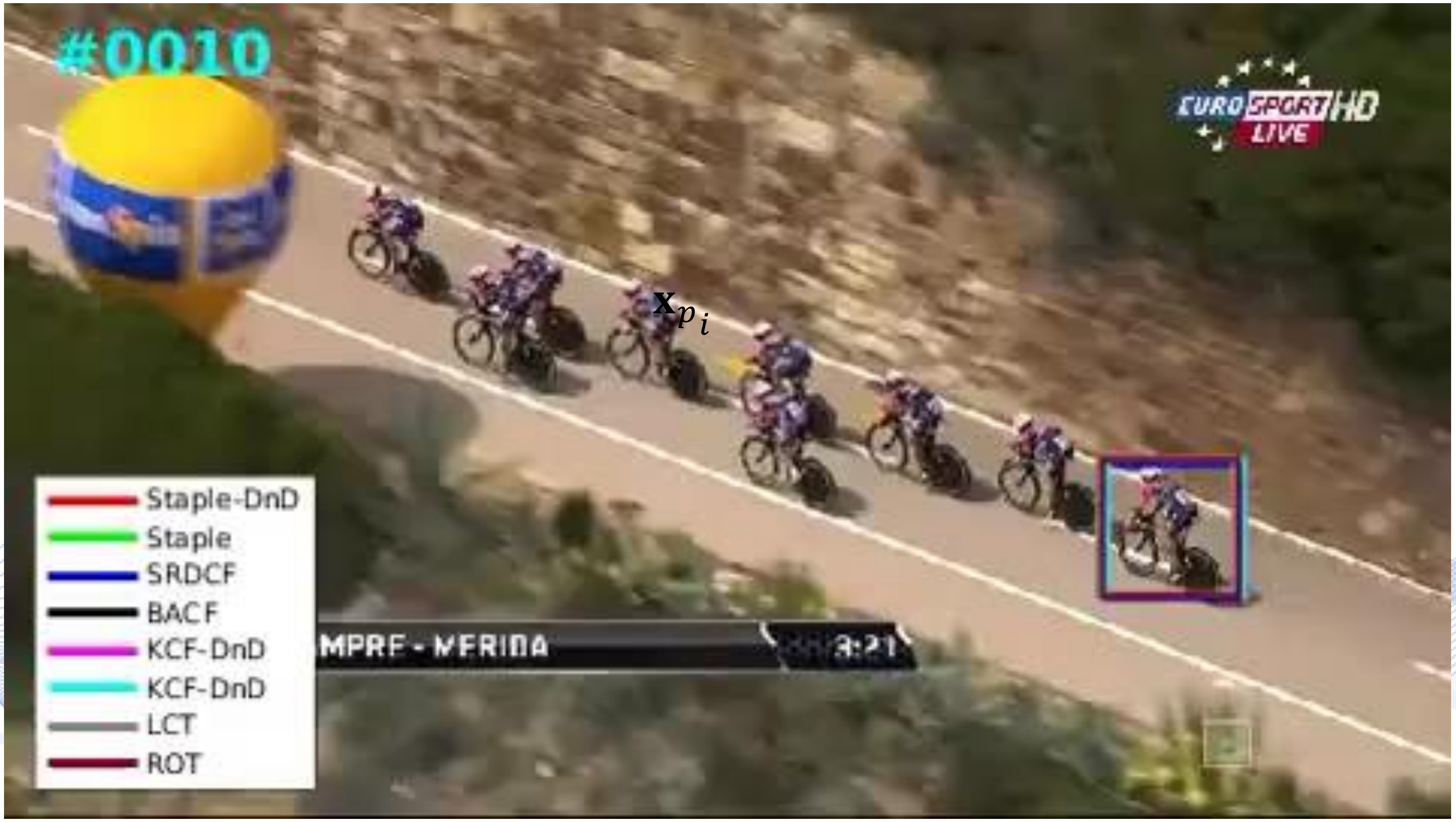
- 3D geometric solid object motion (3D translations, rotations) causing 2D object image transformations:
  - notably 2D translation, rotation, scaling or projective transformations object scaling.

- Effects of camera motion and/or parameter change:
  - zooming, global motion field;

2D Object tracking requirements

- In order to track a moving object, a tracker has to confront:
  - Partial occlusion,
  - Object image deformation,
  - Motion blur,
  - Fast object image motion,
  - Illumination variations,
  - Background clutter.

2D Object Tracking



2D Object Tracking

Detection-tracking loop:

- ***Object (re)detection***.
- ***Object position prediction*** in the next frame and search region initialization (optional).
- ***Object localization*** in the next video frame:
  - Feature/Similarity/Correlation matching.
  - Handling tracking failure (optional):
    - Occlusion detection and handling;
    - Object/background model update;
    - Background discrimination.

Object tracking

Given two video frames at time $t, t+1$ and a detected object at time t described by:

- a vector $\hat{\mathbf{y}}(t) = [\hat{\mathbf{y}}_1{}^T | \mathbf{x}^T | \boldsymbol{\mathcal{I}} | \boldsymbol{\mathcal{M}} | \hat{\mathbf{y}}_2{}^T]^T(t)$ consisting of:
- ROI parameter vector $\hat{\mathbf{y}}_1(t) = [x, y, w, h]^T$.
- ROI image content (feature) vector $\mathbf{x}^T(t)$.
- A unique object id $\boldsymbol{\mathcal{I}}$.
- Object model $\boldsymbol{\mathcal{M}}$ (optional). It can be learnt:
  - a) a set of representative images, b) an ML model.
- (optional) Object identification/recognition:
  - produce a class vector $\hat{\mathbf{y}}_2(t) \in [0,1]^m$.
  - At times, $\boldsymbol{\mathcal{I}}$ may coincide with the winner class label $\hat{\mathbf{y}}_{1w}$.

Object tracking

Track this object in video frame $t + 1$:

- (Optional) Predict object position $[x, y]^T(t + 1)$;
- Find ROI parameter vector $\hat{\mathbf{y}}_1(t + 1) = [x, y, w, h]^T$ within a ***search region*** on video frame $t + 1$;
- Retain object id $\mathcal{J}(t + 1) = \mathcal{J}(t)$;
- update model vector $\mathcal{M}$ (optional);
- Object identification/recognition (optional)
  - produce a class vector $\hat{\mathbf{y}}_2(t + 1) \in [0, 1]^m$.

2D Object Tracking

Tracking failure sources:

- Occlusion, drifting to the background.
- In such cases, ***object re-detection*** is employed.
- If any of the detected objects coincides with any of the objects already being tracked, tracking/detection information can be merged.

Periodic object re-detection:

- It can account for new object appearance (typically every 5-100 video frames).

- Forward and backward tracking:
  - when the entire video is available.

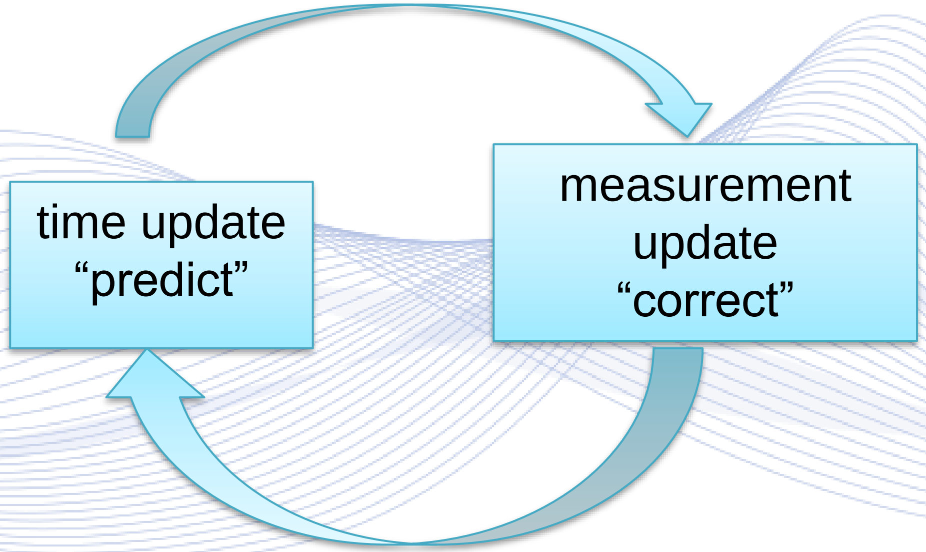
2D Object Tracking

- Introduction
- **Prediction in object tracking**
- Feature Point based trackers
- Region similarity trackers
- Correlation trackers
- Object Detection Performance Metrics

Artificial Intelligence & Information Analysis Lab

Object position prediction

Kalman filter for object position/velocity parameter prediction.



Object position prediction

Kalman filter for object position prediction:

- ***Object state vector***: $\mathbf{x}_t = \left[x, y, v_x, v_y\right]^T$ (2D image plane position, velocity).
- Motion state estimation model: $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{n}_t$
- Position prediction: $\hat{\mathbf{x}}_{t+1} = \mathbf{A}\hat{\mathbf{x}}_t,$

$$\widehat{\mathbf{P}}_{t+1} = \mathbf{A}\widehat{\mathbf{P}}_t\mathbf{A}^T + \mathbf{Q}_s$$

Object position prediction

- Measurement model:

$$\mathbf{z}_{t+1} = \mathbf{H}\mathbf{x}_{t+1} + \mathbf{v}_{t+1},$$

$$\mathbf{K}_{t+1} = \widehat{\mathbf{P}}_t \mathbf{H}^T \left(\mathbf{H}\widehat{\mathbf{P}}_t \mathbf{H}^T + \mathbf{Q}_m\right)^{-1}.$$

- Adjustment of $\mathbf{P}_{t+1}$:

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\hat{\mathbf{x}}_{t+1}),$$

$$\mathbf{P}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\mathbf{P}_{t+1}.$$

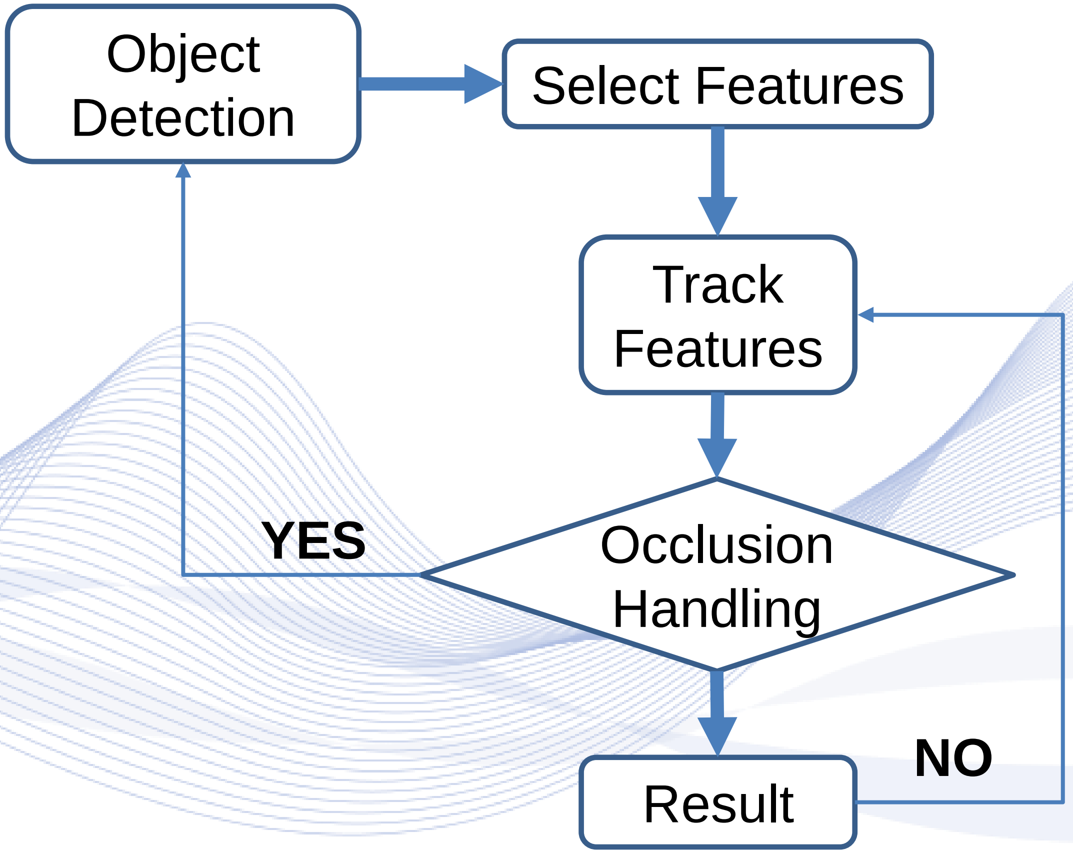
2D Object Tracking

- Introduction
- Prediction in object tracking
- **Feature Point based trackers**
- Region similarity trackers
- Correlation trackers
- Object Detection Performance Metrics

Feature Point based Tracking

Feature point based face tracking:

- Sparse image feature generation.
- Various image features can be used.
- Features may have semantic meaning (e.g., mouth corner).
- Feature points (landmarks) are individually tracked.



Feature point based Tracking

- **Feature point loss**, primarily due to occlusion:
  - The number of features in each tracked region is checked in each frame against a specified threshold.
  - If the number falls below the threshold, features are regenerated.
  - Feature regeneration also takes place at regular intervals, in an effort to further enhance the tracking process.

2D Object Tracking

- Introduction
- Prediction in object tracking
- Feature Point based trackers
- **Region similarity trackers**
- Correlation trackers
- Object Detection Performance Metrics

Region Similarity Tracking

- An object ROI is detected.
- For this ROI, feature vectors are calculated:
    - image color similarity features, e.g., color histogram;
    - structure similarity features (e.g., LSK).
  - They form the object image model.
- A search region is defined around the predicted object position.
    - The search region is divided in overlapping patches.

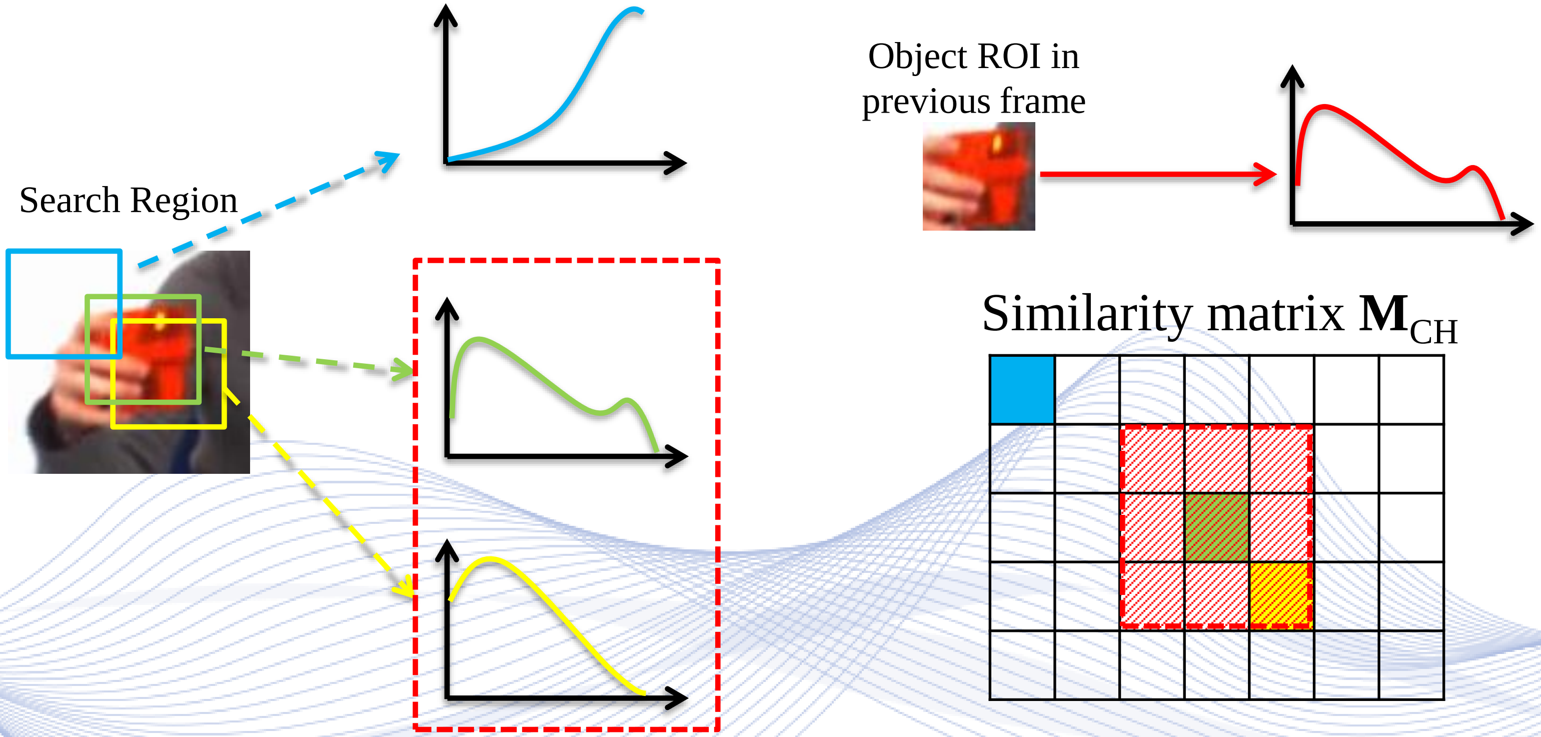
Region Similarity Tracking

- Feature vector similarity between:
  - the ones of image patches
  - the ones of the image model.
- If feature vector similarity is big:
  - tracking is successful.

- When an object appearance change is detected, the object model is updated.

Region Similarity Tracking

- Color-histogram (CH) similarity

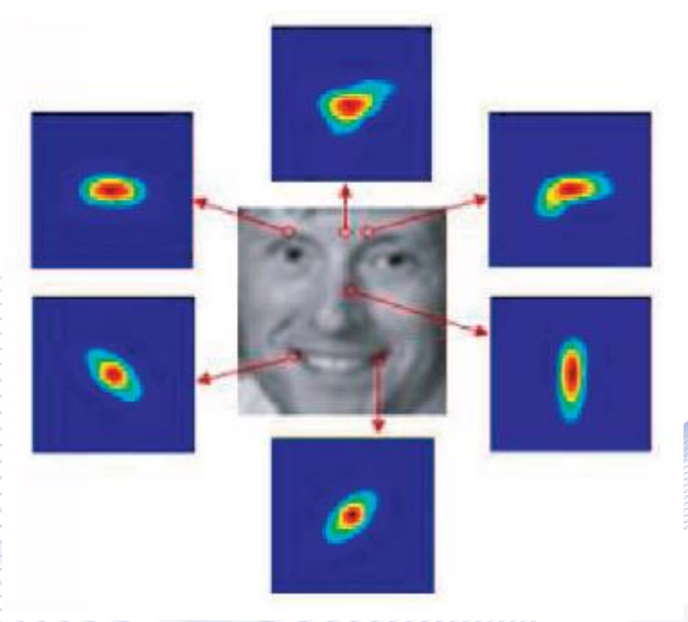

Object ROI in previous frame
Search Region
Similarity matrix $\mathbf{M}_{CH}$

- Only ROIs with the most similar color histograms are retained.

Region Similarity Tracking

Local Steering Kernels (***LSKs***)

- They are a non-linear combination of weighted distances between a pixel and its surrounding pixels.
- They exploit both spatial and edge detection information.
- One LSK vector per pixel is derived.
- LSKs are invariant to brightness & contrast variations and noise.


LSKs [SEO2010].

Region Similarity Tracking

- Local Steering Kernels:

$$K(\mathbf{x}_l - \mathbf{x}) = \frac{\sqrt{\det(\mathbf{C}_l)}}{h^2} \exp\left\{ -\frac{(\mathbf{x}_l - \mathbf{x})^T \mathbf{C}_l^{-1} (\mathbf{x}_l - \mathbf{x})}{2h^2} \right\}.$$

- $\mathbf{C}_l$: Covariance matrix of $k \times k$ neighboring pixel gradient matrix.
- It rotates, elongates, and scales the Gaussian kernel along the local edge.

Region Similarity Tracking

- ***Cosine similarity*** between histogram vectors $\mathbf{h}_1$ and $\mathbf{h}_2$:

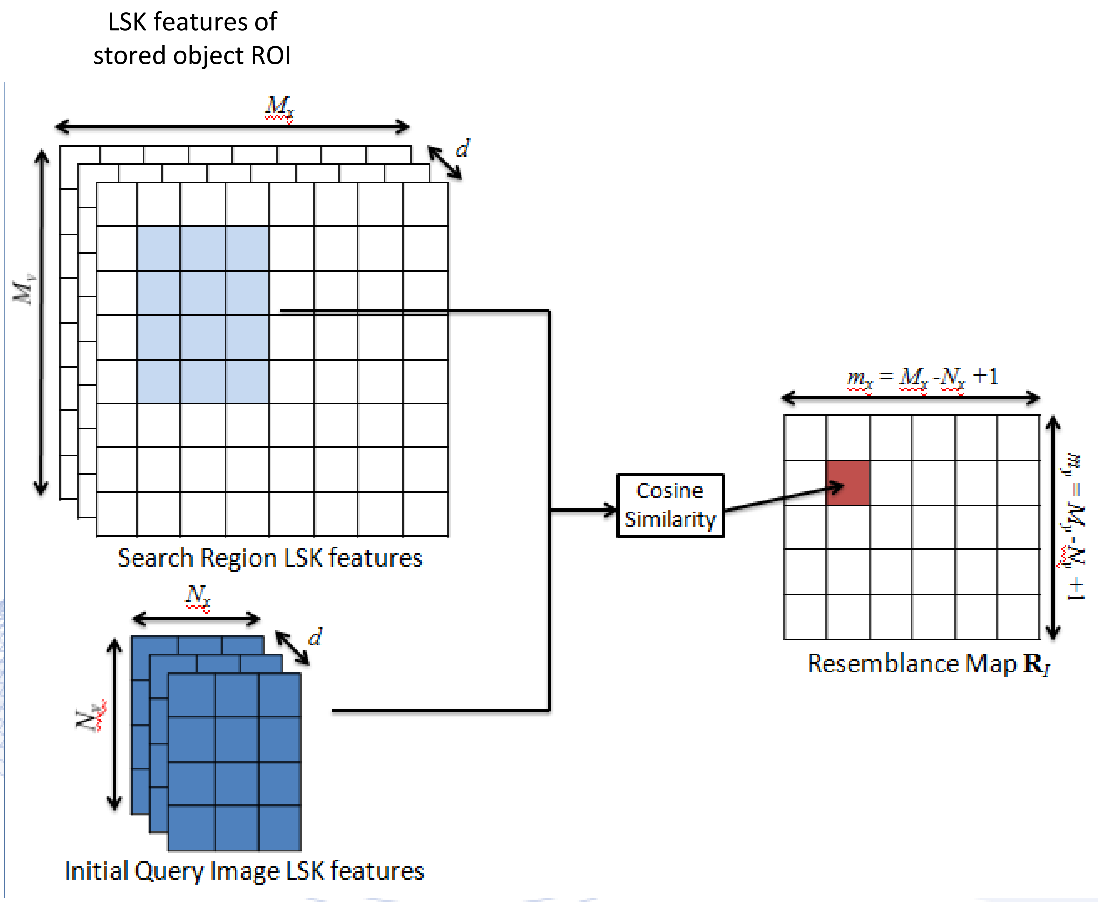
$$S = \frac{s^2}{1-s^2},$$

$$s(\mathbf{h}_1, \mathbf{h}_2) = \cos(\theta) = \frac{\mathbf{h}_1^T \mathbf{h}_2}{\|\mathbf{h}_1\| \|\mathbf{h}_2\|}.$$

- It can be used both for image color and structure feature vectors.

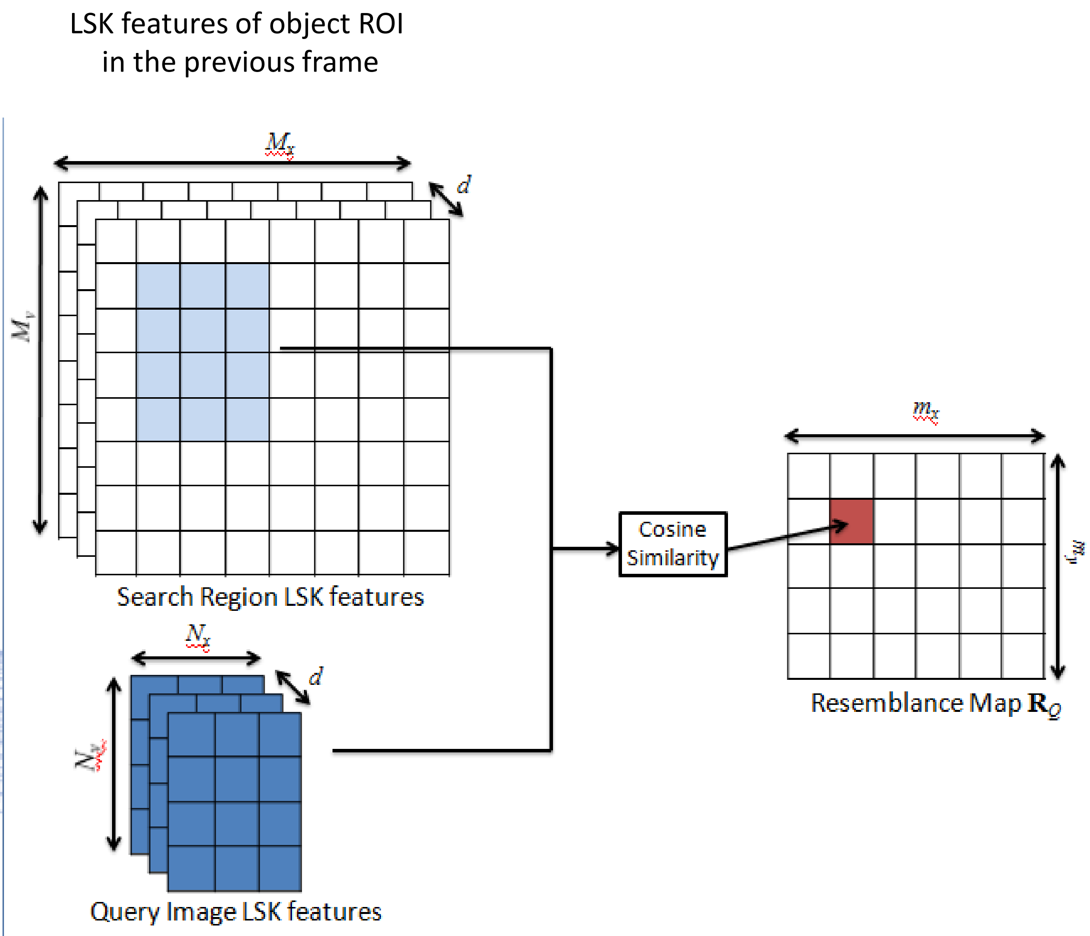
Region Similarity Tracking

- Extract LSKs resemblance map $\mathbf{R}_I$ to a stored object ROI in a previous frame (***object appearance model***).


LSK features of stored object ROI
M_x
M_y
d
Search Region LSK features
N_x
N_y
d
Initial Query Image LSK features
Cosine Similarity
$m_x = M_x - N_x + 1$
$m_y = M_y - N_y + 1$
Resemblance Map $\mathbf{R}_I$

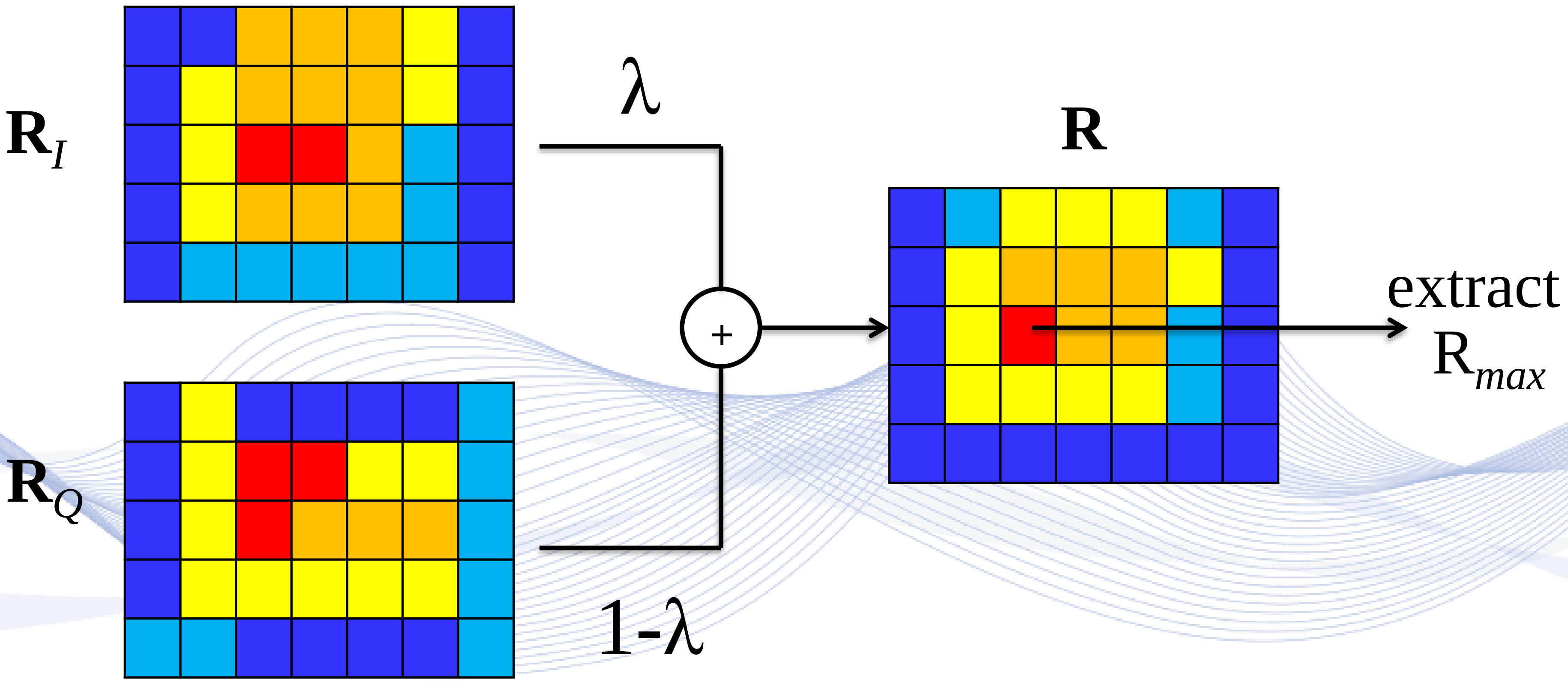
Region Similarity Tracking

- Extract LSKs resemblance map $\mathbf{R}_Q$ to the object ROI instance in the previous video frame.

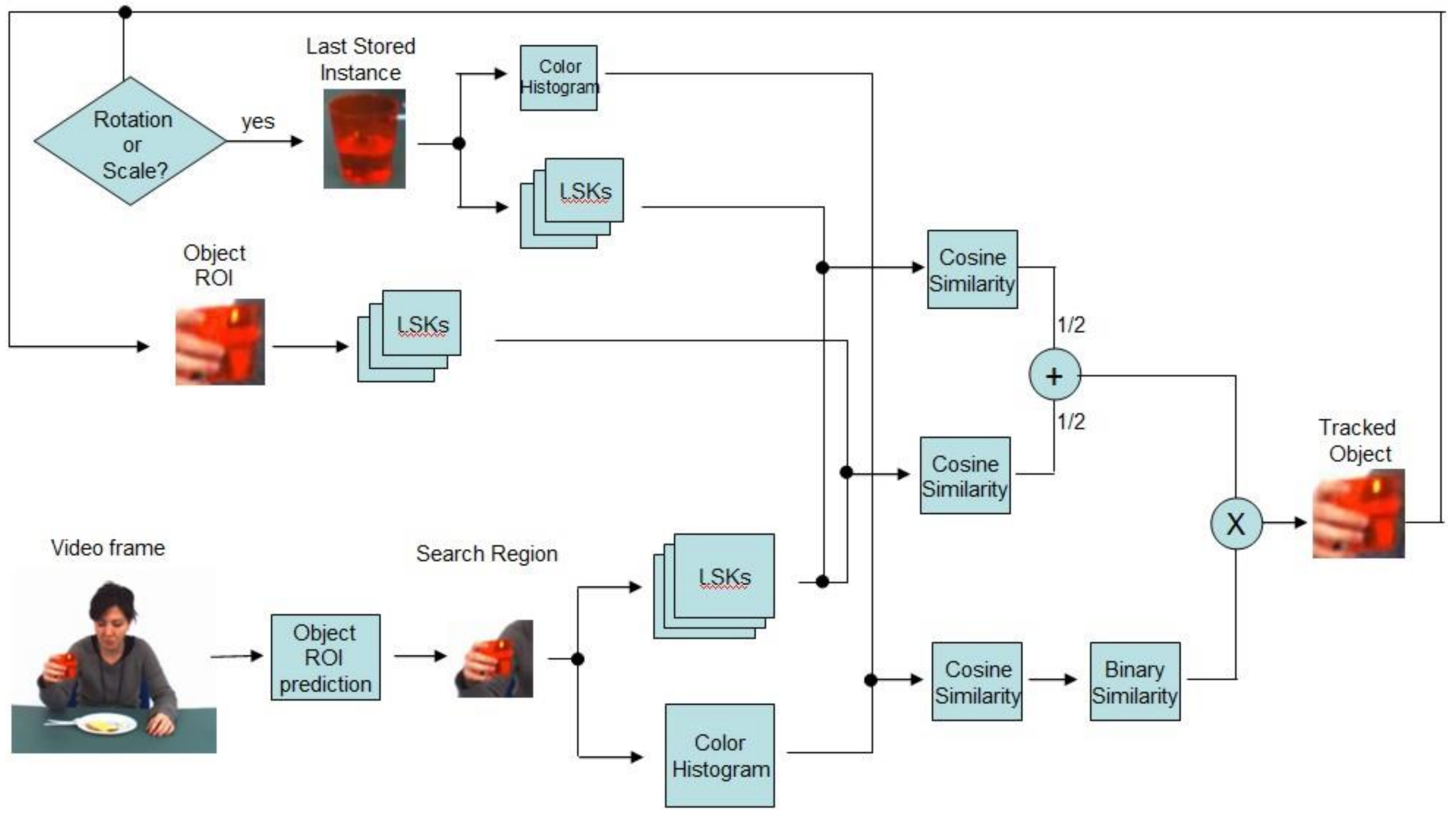

LSK features of object ROI in the previous frame
M_x
d
M_y
Search Region LSK features
N_x
d
N_y
Query Image LSK features
Cosine Similarity
m_x
m_y
Resemblance Map $\mathbf{R}_Q$

Face/Object Tracking based on LSKs

- Extract the new object position.

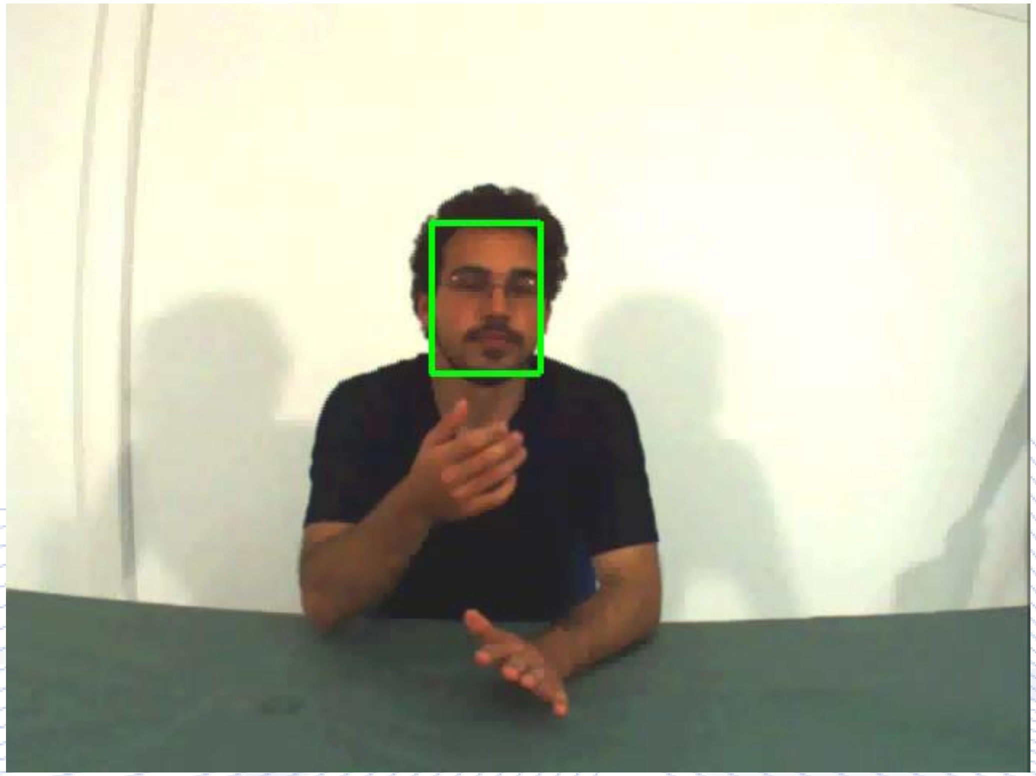


Region Similarity Tracking



Overall LSK tracker block diagram.

Region Similarity Tracking

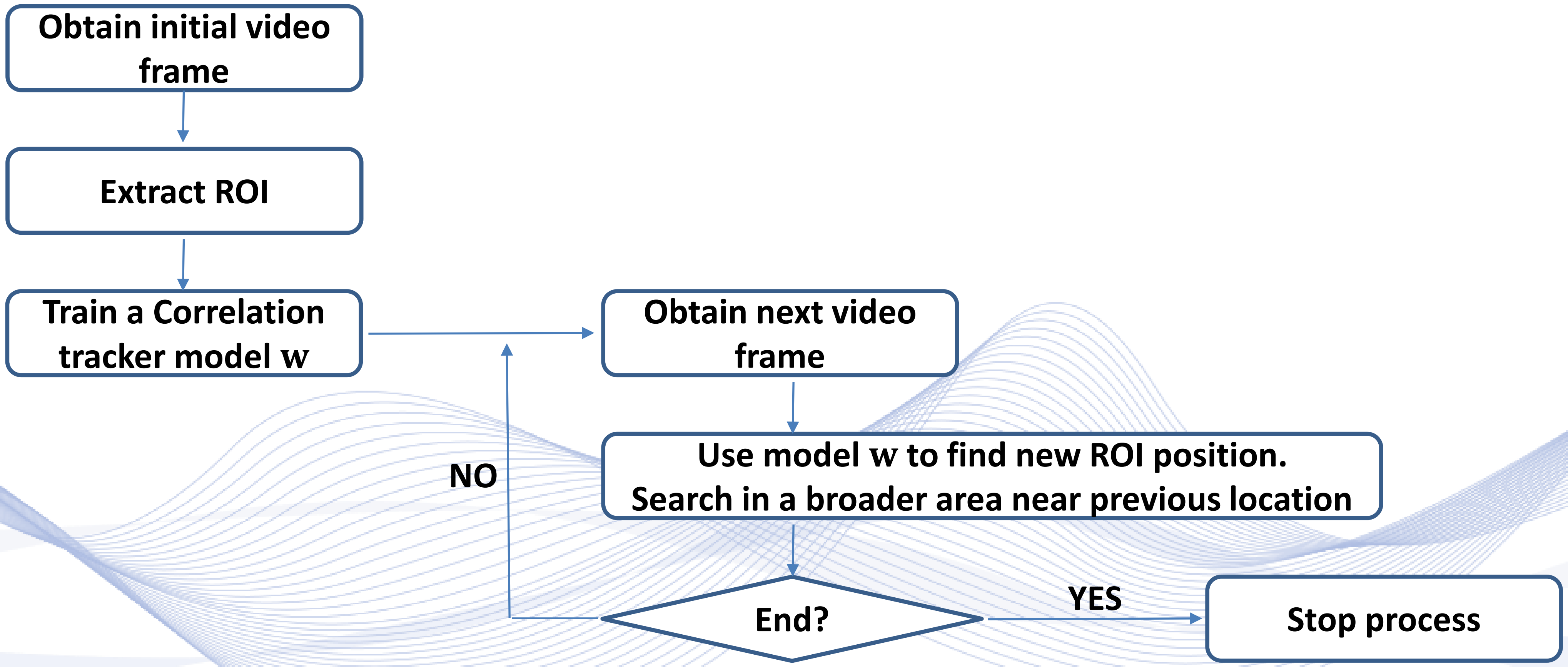


LSK tracker results.

2D Object Tracking

- Introduction
- Prediction in object tracking
- Feature Point based trackers
- Region similarity trackers
- **Correlation trackers**
- Object Detection Performance Metrics

Correlation trackers


Obtain initial video frame
Extract ROI
Train a Correlation tracker model w
Obtain next video frame
Use model w to find new ROI position. Search in a broader area near previous location
End?
NO
YES
Stop process

Correlation trackers

Kernelized Correlation Filter (***KCF***) ***tracker***:

- KCF is a very fast video tracker. Ideal for embedded system applications.
- It can be adapted to use various features (pixel intensity, HOG, etc.) or even use deep features provided by Convolutional Neural Networks (CNNs).
- Standard KCF has no scaling adaptation mechanism but can be modified to this end.

KCF Tracker

- Various image descriptors can be used with KCF.

- Current implementations deploy:
  - Grayscale features
  - HOG
  - Features calculated with deep neural networks

KCF Tracker

Linear regression

- Goal: training a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ that minimizes the objective function:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2.$$

- $\mathbf{x}_i \in \mathbb{R}^n, i = 1, \dots, N$: object ROI feature vectors
- $y_i, i = 1, \dots, N$: regression targets
- $\mathbf{w}$: ***KCF tracker model*** (unknown KCF weight vector).
- λ: regularization parameter.

KCF Tracker

Linear regression solution:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y},$$

- $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]$: $N \times n$ ***data matrix***.
- $\mathbf{y} = [y_1, \dots, y_N]^T$: regression target vector.
- $\mathbf{w}$: unknown weight vector.
- Regularized pseudoinversion is used.
- It forms the theoretical basis for KCF object tracking.
- In the following, $\mathbf{X}^T$ will be used in the place of $\mathbf{X}$, as is the typical notation in the literature.

KCF Tracker

Cyclic Shift trick:

- Train a classifier with one ***object template image vector*** $\mathbf{x} = [x_0 \ldots x_{N-1}]^T$ and several virtual negative samples obtained by permutating its entries.
- All N vector $\mathbf{x}$ permutations produce the ***circulant matrix*** $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ x_{N-1} & x_0 & x_1 & \cdots & x_{N-2} \\ x_{N-2} & x_{N-1} & x_0 & \cdots & x_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & x_3 & \cdots & x_0 \end{bmatrix}.$$

KCF Tracker

- DFT matrix $\mathbf{F}$:

$$\mathbf{F} = \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & \cdots & W_N^{N-1} \\ \cdots & \cdots & \ddots & \vdots \\ W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}.$$

- N complex roots of unity $W_N = e^{-2\pi i/N}$,

$$\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*.$$

- Unitary DFT matrix normalization:

$$\mathbf{U} = \mathbf{F}/\sqrt{N}, \qquad \mathbf{U}^H = (\mathbf{U}^*)^T,\ \mathbf{U}^{-1} = \mathbf{U}^*, \qquad |\det(\mathbf{U})| = 1.$$

Artificial Intelligence & Information Analysis Lab

KCF Tracker

Circulant matrix properties:

- Circulant Matrices can be diagonalized in the Fourier domain:

$$\mathbf{X} = \mathbf{U}\hat{\boldsymbol{X}}\mathbf{U}^H,$$

- $\hat{\boldsymbol{X}} = \mathrm{diag}(\hat{\mathbf{x}})$ is the diagonal matrix of eigenvalues of $\mathbf{X}$, i.e., the DFT of the first row of $\mathbf{X}$:

$$\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}.$$

- The inverse of a circulant matrix is also circulant.
- The sum or product of two circulant matrices is also a circulant matrix.

KCF Tracker

- Example:
  - Circulant matrix $\mathbf{X}$ containing all the possible translations of a vector $\mathbf{x} = [1\ 2\ 3\ 4]$:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}.$$

KCF Tracker

- Unitary 4×4 DFT matrix:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5i & -0.5 & 0.5i \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5i & -0.5 & -0.5i \end{bmatrix},$$

$$\mathbf{U} = \mathbf{U}^* = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5i & -0.5 & 0.5i \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5i & -0.5 & -0.5i \end{bmatrix}, \quad \mathbf{U}\mathbf{U}^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

KCF Tracker

Using the vector $\mathbf{x} = [1\ 2\ 3\ 4]^T$:

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 30 & 24 & 22 & 24 \\ 24 & 30 & 24 & 22 \\ 22 & 24 & 30 & 24 \\ 24 & 22 & 24 & 30 \end{bmatrix}$$

$$\mathrm{diag}(\hat{\mathbf{x}}) = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & -2+2i & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2-2i \end{bmatrix}$$

$$\hat{\mathbf{X}}' = \mathrm{diag}(\hat{\mathbf{x}}^* \otimes \hat{\mathbf{x}}) = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

$$\mathbf{U}\hat{\mathbf{X}}'\mathbf{U}^{\mathrm{H}} = \begin{bmatrix} 30 & 24 & 22 & 24 \\ 24 & 30 & 24 & 22 \\ 22 & 24 & 30 & 24 \\ 24 & 22 & 24 & 30 \end{bmatrix}$$

Artificial Intelligence & Information Analysis Lab

KCF Tracker

Regression result in the DFT domain.

$$\mathbf{X}^T\mathbf{X} = \mathbf{F}\mathrm{diag}(\hat{\mathbf{x}}^* \otimes \hat{\mathbf{x}})\mathbf{F}^H.$$

- $\hat{\mathbf{x}}^*$: conjugate of vector $\hat{\mathbf{x}}$.

Using the above equation in:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y},$$

we find:

$$\widehat{\mathbf{w}} = (\hat{\mathbf{x}}^* \otimes \hat{\mathbf{y}}) \oslash (\hat{\mathbf{x}}^* \otimes \hat{\mathbf{x}} + \lambda),$$

$$\mathbf{w} = \mathbf{F}^{-1}\widehat{\mathbf{w}}.$$

- $\otimes$,$\oslash$: pointwise vector multiplication/division.
- $\mathbf{w}$ is essentially a normalized correlation of vectors $\mathbf{x}, \mathbf{y}$.

Artificial Intelligence & Information Analysis Lab

KCF Tracker

Non-linear regression

- ***Kernel trick*** allows non-linear regression functions $f(\mathbf{z})$.
- Nonlinear mapping $\boldsymbol{\phi}(\mathbf{x})$ of input vector $\mathbf{x} \in \mathbb{R}^n$ to a ***kernel space*** $\mathbb{R}^L$: $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^L,\ \ L > n$.
- Correlation coefficient vector $\mathbf{w}$ can be expressed as a linear combination input samples:

$$\mathbf{w} = \sum_{i=1}^{N} a_i \boldsymbol{\phi}(\mathbf{x}_i).$$

- Instead of seeking $\mathbf{w}$**,** it is enough to find $a_i, i = 1,..,N$.

KCF Tracker

Non-linear regression

- The inner product on kernel space $\mathbb{R}^L$ can be computed using the nonlinear ***kernel function*** κ:

$$\boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j).$$

- It can be precomputed and stores in the $N \times N$ ***kernel matrix*** $\mathbf{K}$:

$$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j).$$

- Function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is known, even if $\boldsymbol{\phi}(\mathbf{x})$ may be unknown.

KCF Tracker

Common kernel functions:

- Linear kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j.$
- Polynomial kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (a\mathbf{x}_i^T \mathbf{x}_j + b)^d.$
- RBF kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-(\|\mathbf{x}_i - \mathbf{x}_j\|^2)/2\sigma^2}.$
- Adaptive kernels (e.g., intersection):

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \min(x_{ik}, x_{jk}).$$

KCF Tracker

Non-linear regression

- Regression function:

$$f(\mathbf{z}) = \mathbf{w}^T\mathbf{z} = \sum_{i=1}^{N} a_i \kappa(\mathbf{z}, \mathbf{x}_i).$$

- Regression function complexity grows linearly with the number of training samples N.

KCF Tracker

Non-linear regression

Theorem: Given N permutated versions $\mathbf{x}_i = 0, \dots, N-1$ of vector $\mathbf{x}$, their corresponding kernel matrix $\mathbf{K}$ is circulant, if kernel function satisfies $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\mathbf{P}\mathbf{x}_i, \mathbf{P}\mathbf{x}_j)$, for any permutation matrix $\mathbf{P}$.

KCF Tracker

The following kernels satisfy the above theorem:

- Radial Basis Function kernels, e.g., the Gaussian kernel;
- Dot-product kernels, e.g., linear, polynomial kernels;
- Adaptive kernels , e.g., the intersection kernel;
- Exponentiated additive kernels.

KCF Tracker

- Mapping the linear regression solution:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y},$$

to kernel space $\mathbb{R}^L$ produces the kernel coefficient vector $\mathbf{a} = [a_1, \ldots a_N]^T$:

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

- If $\mathbf{K}$ is circulant, this solution can be diagonalized to obtain a fast solution:

$$\hat{\mathbf{a}} = \hat{\mathbf{y}} \oslash (\hat{\mathbf{k}}_{\mathbf{xx}} + \lambda).$$

- $\hat{\mathbf{k}}_{\mathbf{xx}}$ is the 1D Fourier transform of the N elements $\mathbf{k}_{\mathbf{xx}} = \kappa(\mathbf{x}, \mathbf{x})$.

KCF Tracker

Multiple channels

- Working in the dual space has the advantage of allowing operation on multiple data channels:
- For example, the orientation bins of a HOG descriptor can be used by summing over them in the Fourier domain.
- To deal with multiple channels, it can be assumed that a vector $\mathbf{x}$ results from the concatenation of the individual vectors for C channels $\mathbf{x} = [\mathbf{x}_1^T \cdots \mathbf{x}_C^T]^T$.

KCF Tracker

Multiple channels

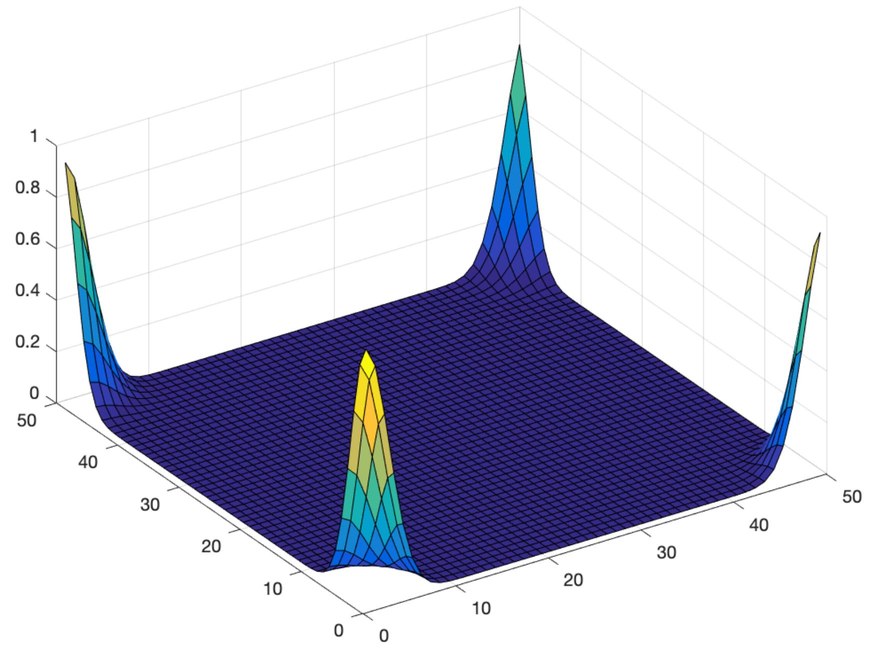
- Previously discussed kernels are based on vector dot products or vector norms.
- A dot product can be computed by simply summing the individual dot products for each channel.
- DFT linearity allows the summation of the result for each channel in the Fourier domain.
- Example: multi-channel analogue of the Gaussian kernel:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{\sigma^2}\left(\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{F}^{-1}\left(\Sigma_c \hat{\mathbf{x}}_{ic}^{*} \odot \hat{\mathbf{x}}_{jc}\right)\right)\right).$$

KCF Tracker

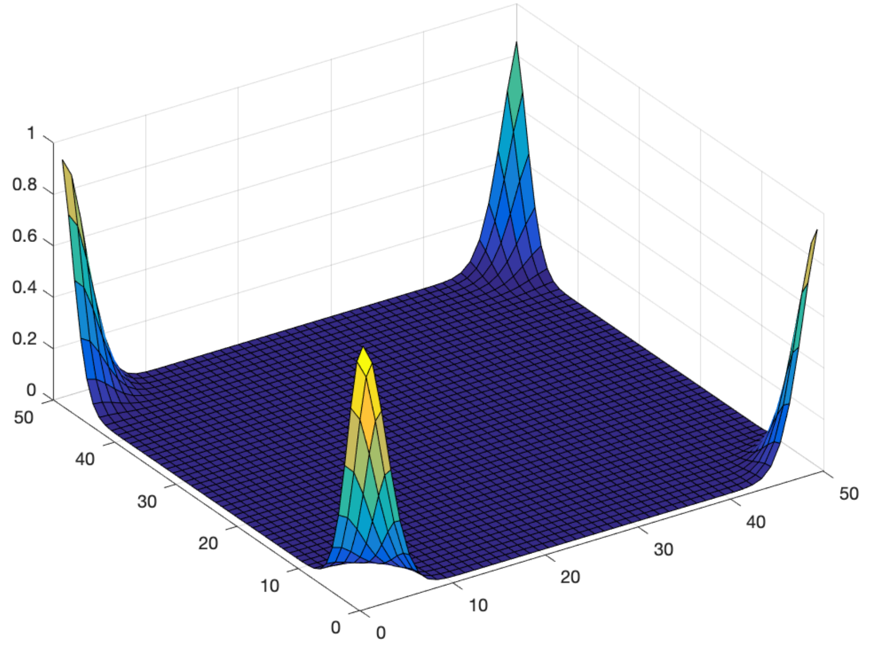
Definition of the regression target $\mathbf{y}$:

- A 2D Gaussian distribution is used.
- Higher value corresponds to the non permuted data vector $\mathbf{x}$.



KCF Tracker

- Placing the peak in the middle will unnecessarily cause the detection output to be shifted by half a window.
- Placing the peak at the top-left element (and wrapping around) correctly centers the detection output.



KCF Tracker

- On video frame t, calculate:

$$\hat{\mathbf{x}} = \mathbf{F}\mathbf{x},$$

$$\hat{\mathbf{w}} = (\hat{\mathbf{x}}^* \otimes \hat{\mathbf{y}}) \oslash (\hat{\mathbf{x}}^* \otimes \hat{\mathbf{x}} + \lambda),$$

$$\mathbf{w} = \mathbf{F}^{-1}\hat{\mathbf{w}}.$$

- To find the new position on video frame $t+1$ maximize:

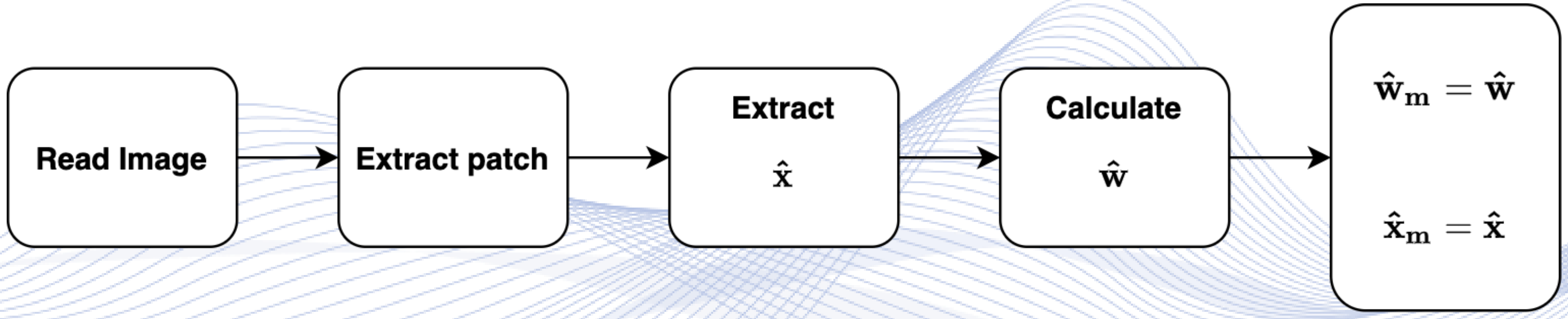
$$f(\mathbf{z}) = \mathbf{w}^T\mathbf{z},$$

  - $\mathbf{z}$: feature vector of an image patch in the search region.
  - Highest $f(\mathbf{z})$ value location indicates the new object position.

Artificial Intelligence & Information Analysis Lab

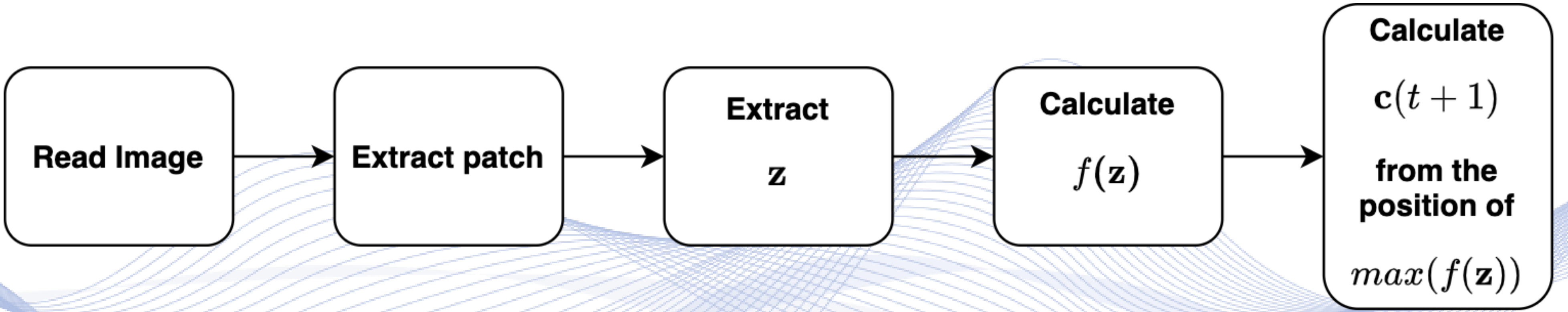
KCF Algorithm implementation

- Initial frame ($t = 0$):
  - Initialize $\hat{\mathbf{w}}$,
  - Initialize $\hat{\mathbf{w}}_m$, $\hat{\mathbf{x}}_m$: interpolated model and target features.


Read Image
Extract patch
Extract $\hat{\mathbf{x}}$
Calculate $\hat{\mathbf{w}}$
$\hat{\mathbf{w}}_m = \hat{\mathbf{w}}$
$\hat{\mathbf{x}}_m = \hat{\mathbf{x}}$

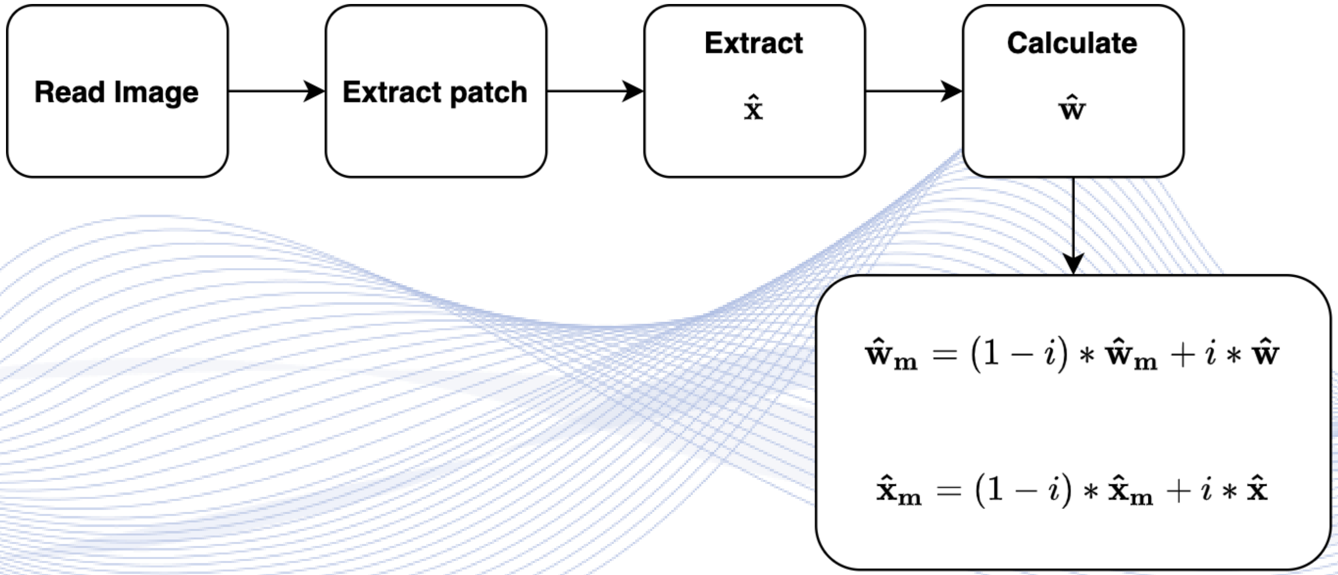
KCF Algorithm implementation

- Next frame ($t = t + 1$)
  - update target position $\mathbf{c}(t+1)$:


Read Image
Extract patch
Extract $\mathbf{z}$
Calculate $f(\mathbf{z})$
Calculate $\mathbf{c}(t+1)$ from the position of $max(f(\mathbf{z}))$

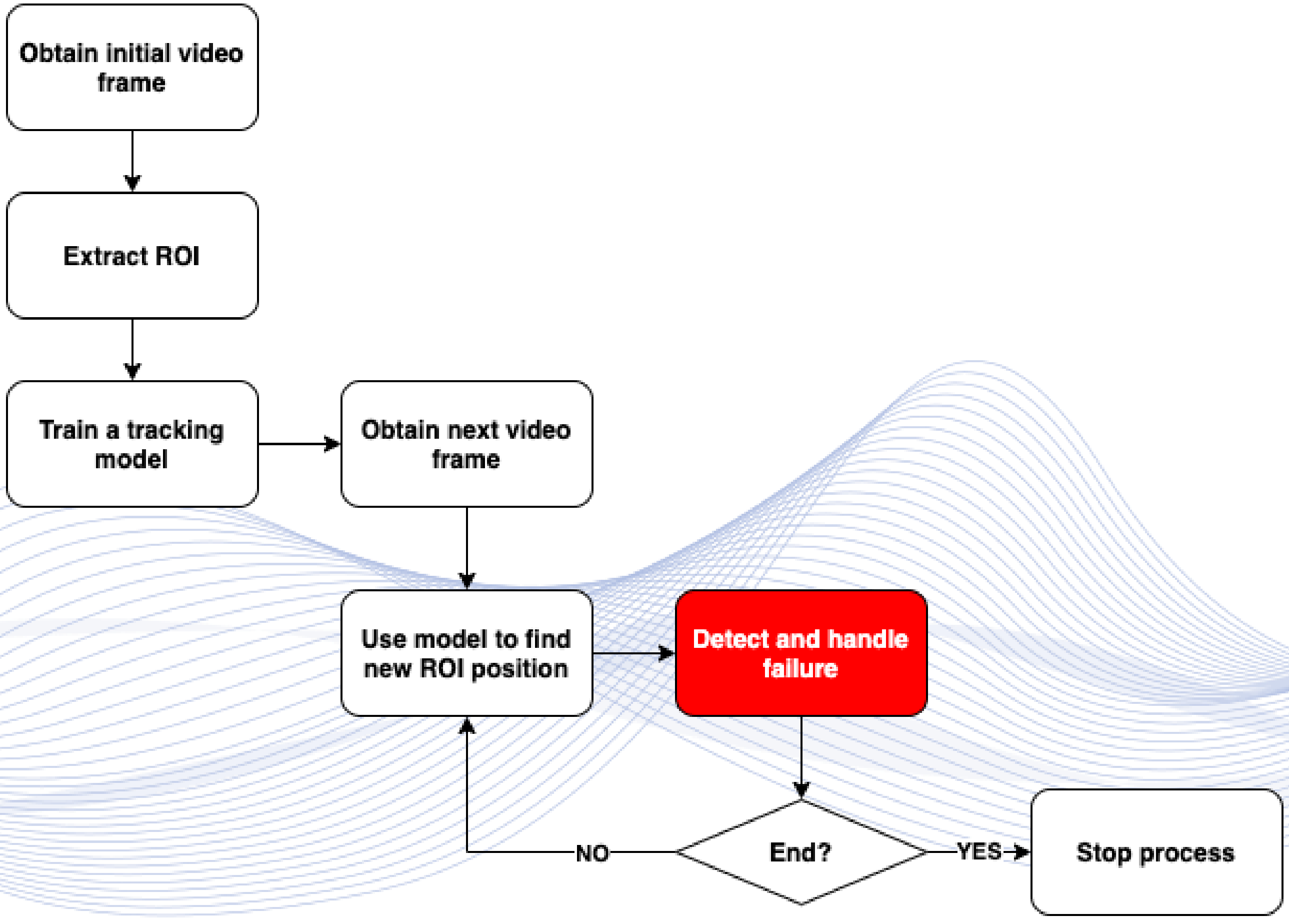
KCF Algorithm implementation

- Next frame ($t = t + 1$)
  - interpolate $\hat{\mathbf{w}}_m, \hat{\mathbf{x}}_m$:



Read Image
Extract patch
Extract $\hat{\mathbf{x}}$
Calculate $\hat{\mathbf{w}}$
$\hat{\mathbf{w}}_\mathbf{m} = (1 - i) * \hat{\mathbf{w}}_\mathbf{m} + i * \hat{\mathbf{w}}$
$\hat{\mathbf{x}}_\mathbf{m} = (1 - i) * \hat{\mathbf{x}}_\mathbf{m} + i * \hat{\mathbf{x}}$

Handling Tracking Failure


Obtain initial video frame
Extract ROI
Train a tracking model
Obtain next video frame
Use model to find new ROI position
Detect and handle failure
End?
NO
YES
Stop process

Handling Tracking Failure

Occlusion handling:

- If the tracker knows somehow that the target is occluded, then it acts accordingly:
  - ***Partial occlusion***: The tracker tracks but does not update its model.
  - ***Total occlusion***: The tracker is employed as a detector in a surrounding area.

Handling Tracking Failure

Two approaches have been investigated for ***occlusion handling***:

- ***Peak-to-Sidelobe-Ratio*** r of the tracker responses $R = f(\mathbf{z})$:

$$r = \frac{\max(R) - \mathrm{mean}(R)}{\mathrm{std}(R)}.$$

- Low r values indicate tracker failure, e.g., due to occlusion.

Artificial Intelligence & Information Analysis Lab

Handling Tracking Failure

- Learn a two-class SVM tracker response model (occlusion/no-occlusion) from the tracker responses in other videos.
- ***Challenge***: Partial occlusions are difficult to handle, and the tracker might confuse them with heavy rotations. In some cases, the model still needs to be trained, and in other cases, not.

Handling Tracking Failure

Occlusion handling consists of 3 components:

- Baseline tracker (e.g., KCF, Staple, etc.);
- Occlusion detector trained from the tracker responses on occluded/non occluded items;
- Re-detection scheme (employing the tracker).

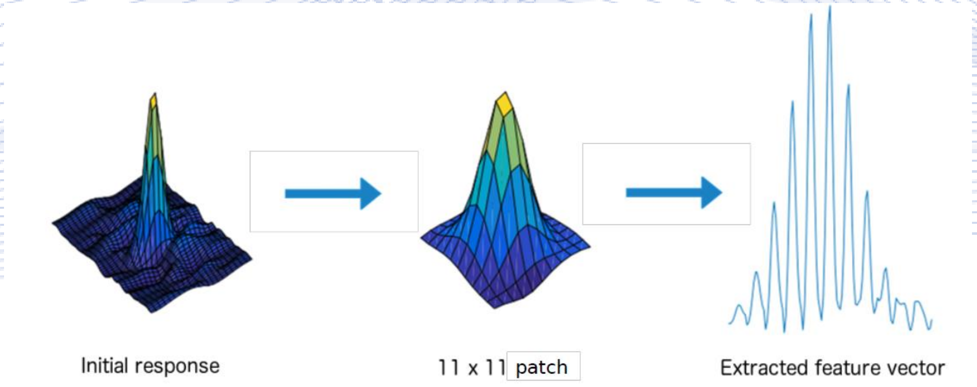
Algorithm

- Initialize tracking;
- Deploy a 2-class SVM classifier in order to detect if target is occluded or not;
- If occlusion is detected, stop training the tracker;
- Re-detect the target.

Handling Tracking Failure

Occlusion handling (features for SVM classifier):

- Obtain the tracker response $f(\mathbf{z})$ on a search region, crop and vectorize the central area (11×11 patch), corresponding to low frequencies.
- When no occlusions or heavy translation occur, response values $f(\mathbf{z})$ are larger in this patch.

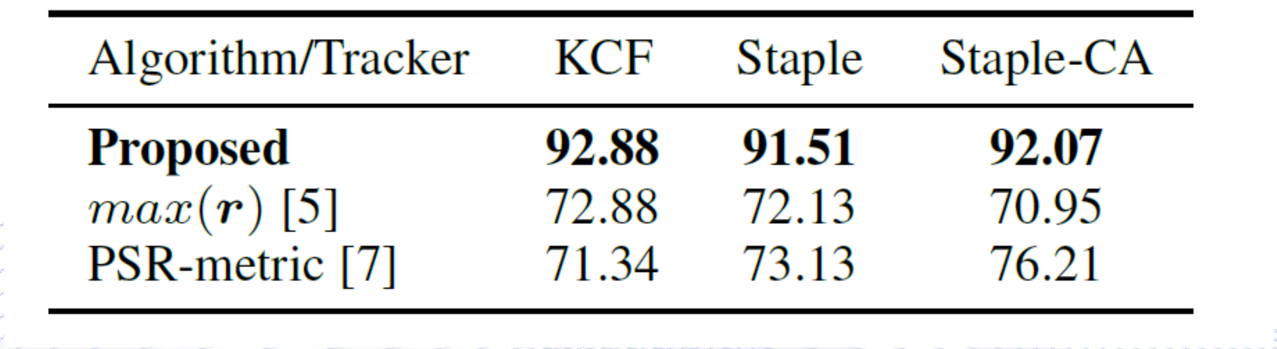


Handling Tracking Failure

- VOT 2017 contains occluded/non occluded frame annotations.
- We obtained the responses of KCF, Staple and Context-Aware (CA) Staple, using standard HoG features
- We compared our proposed SVM-based Occlusion detector with other well-known Occlusion detectors from statistical features (max tracker response, PSR-metric)

Handling Tracking Failure

Cross-validation tracking accuracies.

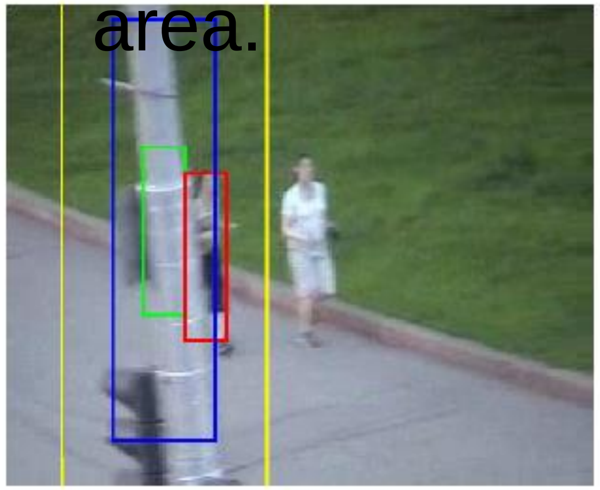


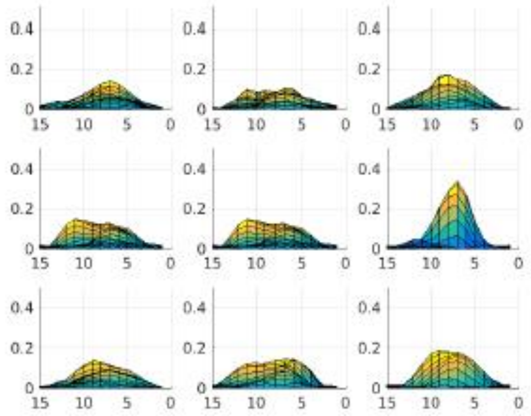
Algorithm/Tracker	KCF	Staple	Staple-CA
Proposed	**92.88**	**91.51**	**92.07**
$max(\boldsymbol{r})$ [5]	72.88	72.13	70.95
PSR-metric [7]	71.34	73.13	76.21

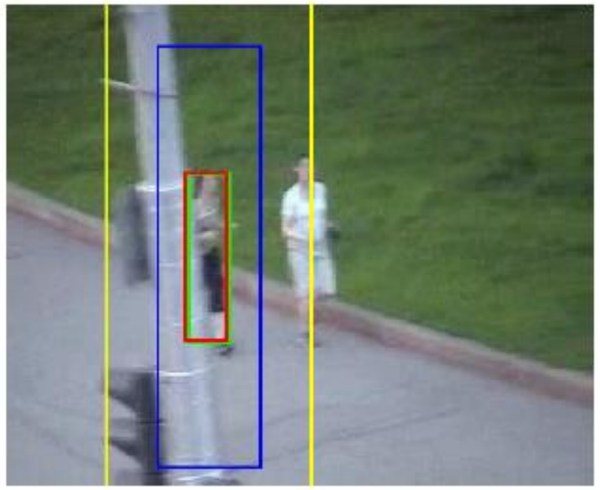
Handling Tracking Failure

Object re-detection:

- If occlusion is detected, the tracker model is employed at the ***re-detection area***.
- The re-detection area is larger than the standard search area.







Tracker outputs (green), Window size area (blue), Re-detection area (yellow), target position (red).

Correlation trackers

Baseline Correlation Trackers:

- Minimum Output Sum of Squared Errors (MOSSE).
- Circulant Structure Kernels (CSK).
- Spatio-Temporal Context (STC).
- Kernelized Correlation Filters (KCF) / Dual Correlation Filters (DCF).

Correlation trackers

Scaling Handling Correlation Trackers:

- Discriminative Scale Space Tracker (DSST).
- Scalable Kernel Correlation Filter (SKCF).
- Scale Adaptive with Multiple Features (SAMF).
- Kernelized Correlation Filter with Detection Proposal (KCFDP).

2D Object Tracking

- Introduction
- Prediction in object tracking
- Feature Point based trackers
- Region similarity trackers
- Correlation trackers
- **Object Detection Performance Metrics**

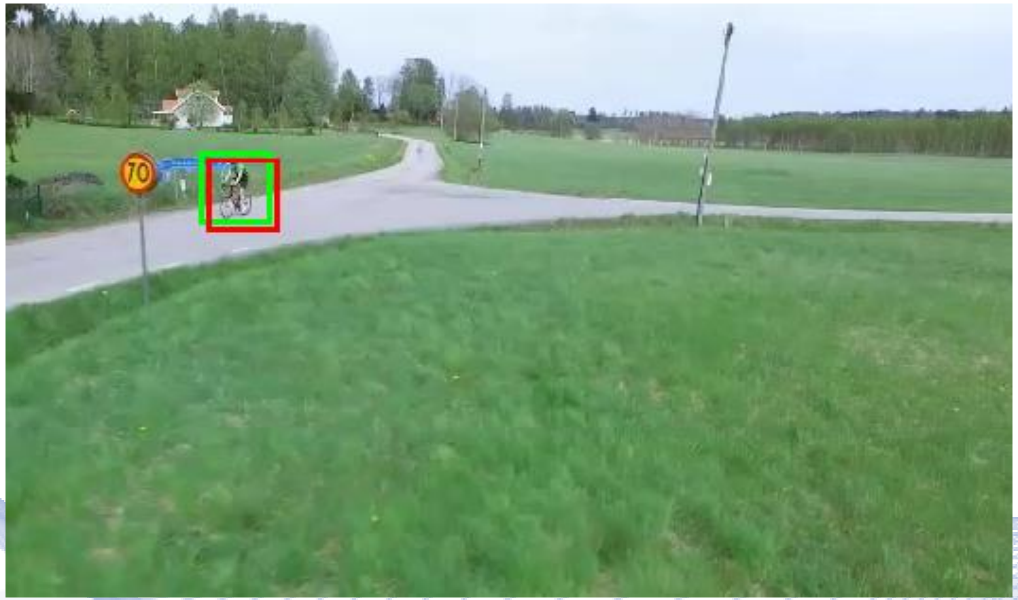
Object Detection Performance Metrics

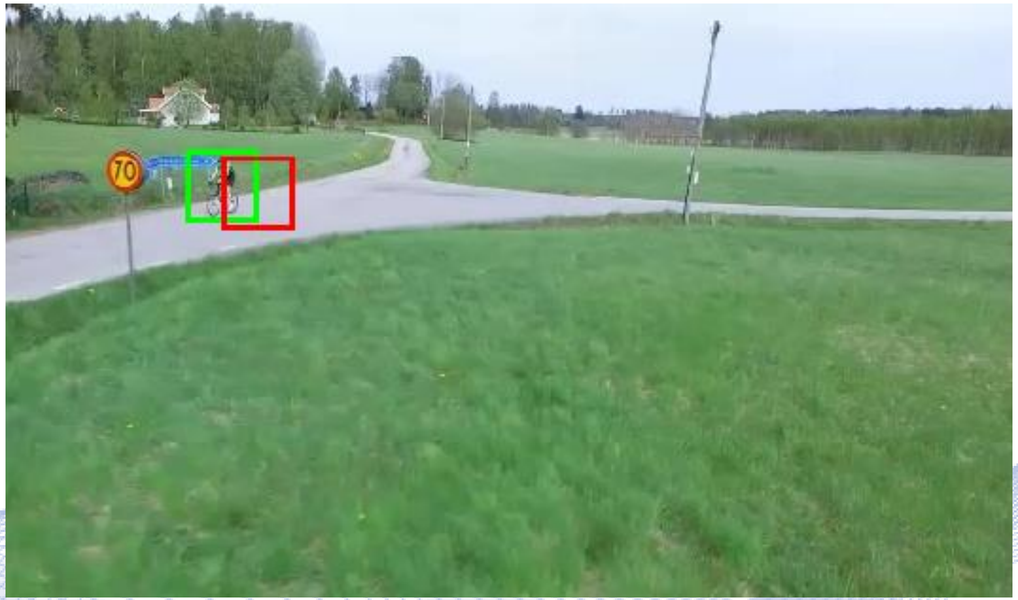
Intersection over Union (***IoU***):

$$J(\mathcal{A},\mathcal{B}) = |\mathcal{A} \cap \mathcal{B}|/|\mathcal{A} \cup \mathcal{B}|.$$

- $\mathcal{A}, \mathcal{B}$: estimated, ground truth ROIs (sets, bounding boxes).
- $|\mathcal{A}|$: set cardinality (area counted in pixels)
- Also called ***Jaccard Similarity Coefficient*** or ***Overlap Score.***

Object Detection Performance Metrics





Object detection: a) $J(\mathcal{A},\mathcal{B}) = 0.67$; b) $J(\mathcal{A},\mathcal{B}) = 0.27$.

2D Tracking Performance metrics

- Overlap Score $J(\mathcal{A},\mathcal{B})$ is calculated in a per frame basis.
- Its average value resulting on all frames can be used as the tracking success metric.

- Whenever $J(\mathcal{A},\mathcal{B})$ is below 0.5 tracking failure can be assumed

Region Similarity Tracking

- ***Frame Detection Accuracy*** (frame-based):

$$FDA(t) = \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{|G_i(t) \cap D_i(t)|}{|G_i(t) \cup D_i(t)|}$$

- ***Average Tracking Accuracy*** (video-based):

$$ATA = \frac{1}{N}\sum_{t=1}^{N} FDA(t)$$

- ***Overall Tracking Accuracy***:

$$OTA = \frac{1}{N_T}\sum_{i=1}^{n} N_i ATA_i$$

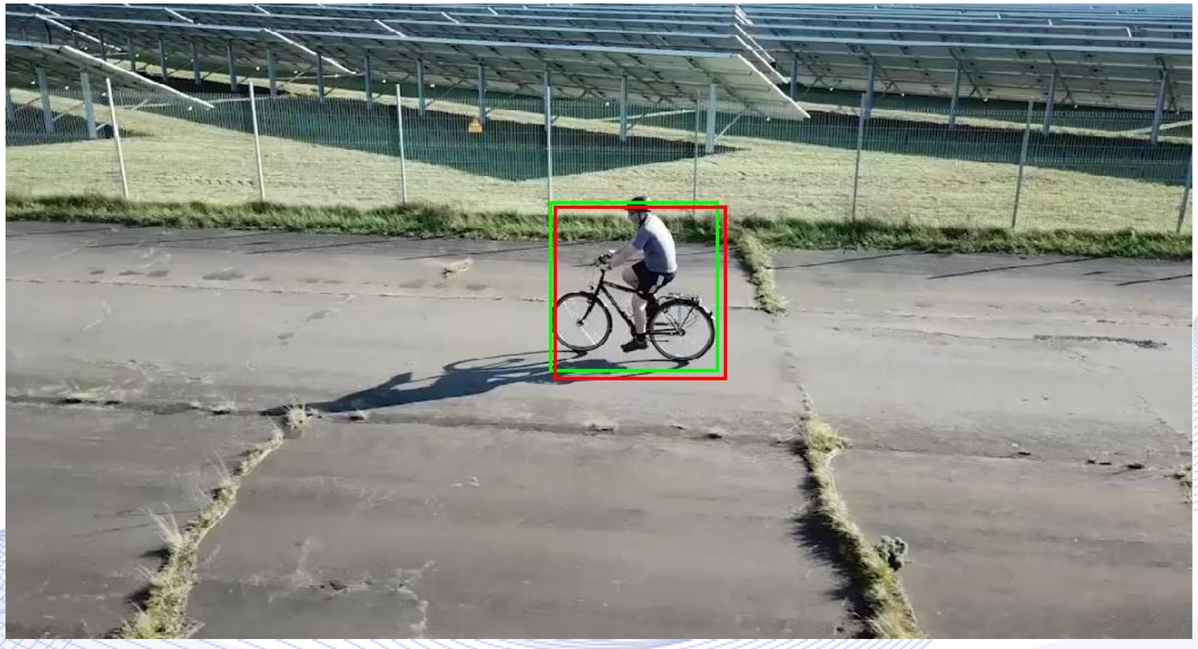
Region Similarity Tracking

Other tracking performance metrics:

- ***Tracking precision*** is the percentage of frames, in which the estimated locations are within a given threshold (e.g., Euclidean distance 20 pixels) from the ground truth target location.

2D Tracking Performance metrics

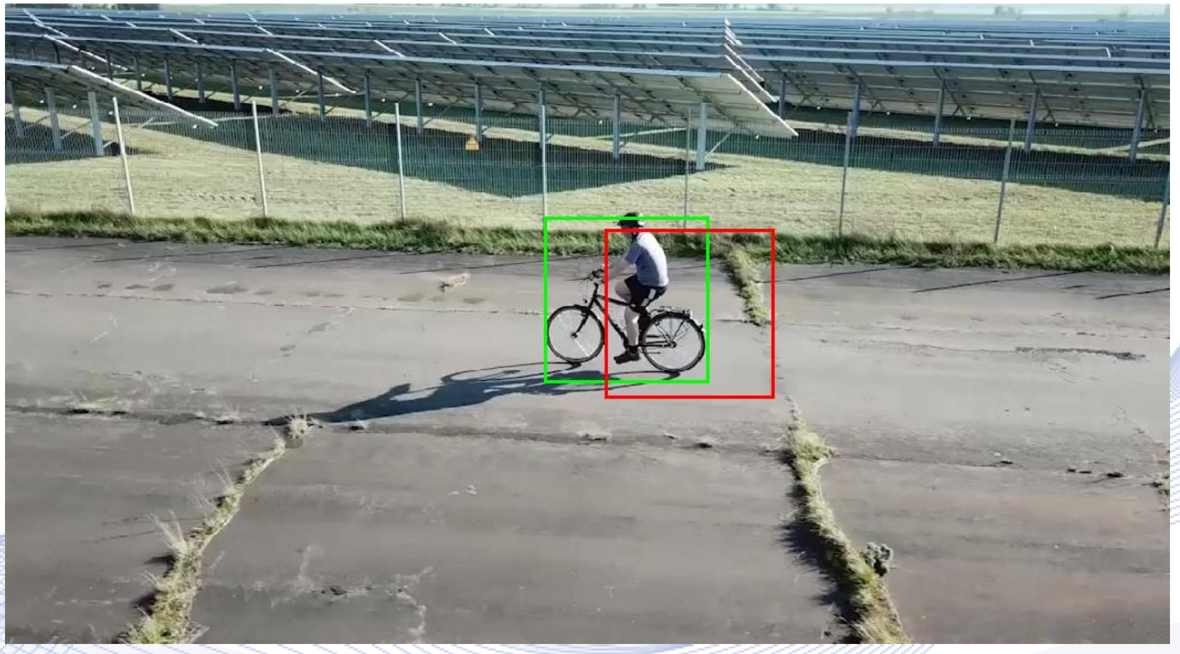
- Red box: tracking results.
- Green box: ground-truth.
- Euclidean distance of central points: 6 pixels.
- $J(\mathcal{A},\mathcal{B}) = 0.9$
- Tracking: ***SUCCESS***.



Artificial Intelligence & Information Analysis Lab

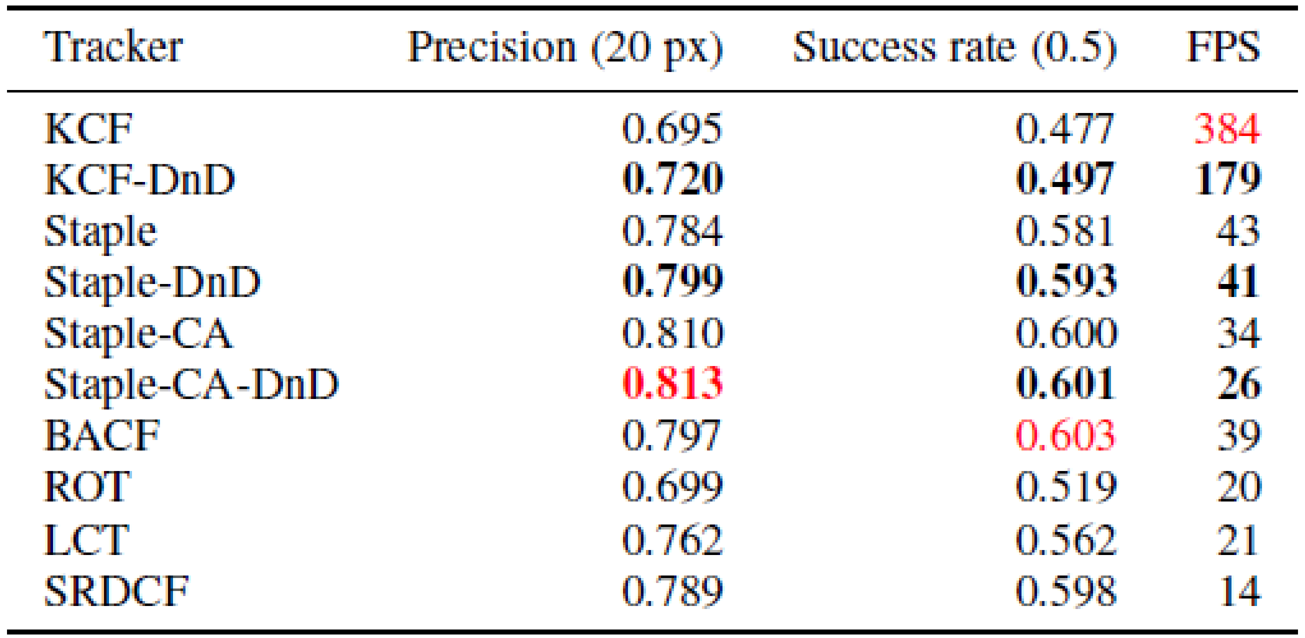
2D Tracking Performance metrics

- Red box: tracking results.
- Green box: ground-truth.
- Euclidean distance of central points: 61 pixels.
- $J(\mathcal{A},\mathcal{B}) = 0.4$
- Tracking: ***FAILURE***.



Tracker accuracy

- DnD denotes the tracking enhancing framework implementation.
- OTB-100 results:



Tracker	Precision (20 px)	Success rate (0.5)	FPS
KCF	0.695	0.477	384
KCF-DnD	**0.720**	**0.497**	**179**
Staple	0.784	0.581	43
Staple-DnD	**0.799**	**0.593**	**41**
Staple-CA	0.810	0.600	34
Staple-CA-DnD	**0.813**	**0.601**	**26**
BACF	0.797	0.603	39
ROT	0.699	0.519	20
LCT	0.762	0.562	21
SRDCF	0.789	0.598	14

Bibliography

[PIT2017] I. Pitas, “Digital video processing and analysis” , China Machine Press, 2017 (in Chinese).

[PIT2013] I. Pitas, “Digital Video and Television” , Createspace/Amazon, 2013.

[PIT2021] I. Pitas, “Computer vision”, Createspace/Amazon, in press.

[NIK2000] N. Nikolaidis and I. Pitas, “3D Image Processing Algorithms”, J. Wiley, 2000.

[PIT2000] I. Pitas, “Digital Image Processing Algorithms and Applications”, J. Wiley, 2000.

Bibliography

[PYI] https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[HEN2014] Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, *37*(3), 583-596.

[KAR2019] Karakostas, I., Mygdalis, V., Tefas, A., & Pitas, I. (2019, September). On Detecting and Handling target occlusions in Correlation-filter-based 2D tracking. In *2019 27th European Signal Processing Conference (EUSIPCO)* (pp. 1-5).

[BOL2010] Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010, June). Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 2544-2550). IEEE.

[OTB100] http://cvlab.hanyang.ac.kr/tracker_benchmark/index.html

[SEO2010] H.J. Seo and P. Milanfar, “Training-free, Generic Object Detection using Locally Adaptive Regression Kernels”, *IEEE Trans. On PAMI*, vol. 32, no.9, pp. 1688-1704, Sept. 2010

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr