# Fast Correlation Filter Tracking

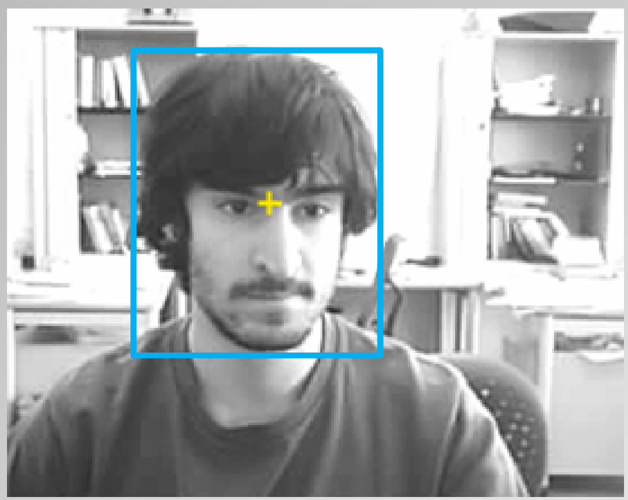
## Tricks of the Trade
(and some history)

J.F. Henriques

Institute of Systems and Robotics
University of Coimbra

# Visual tracking

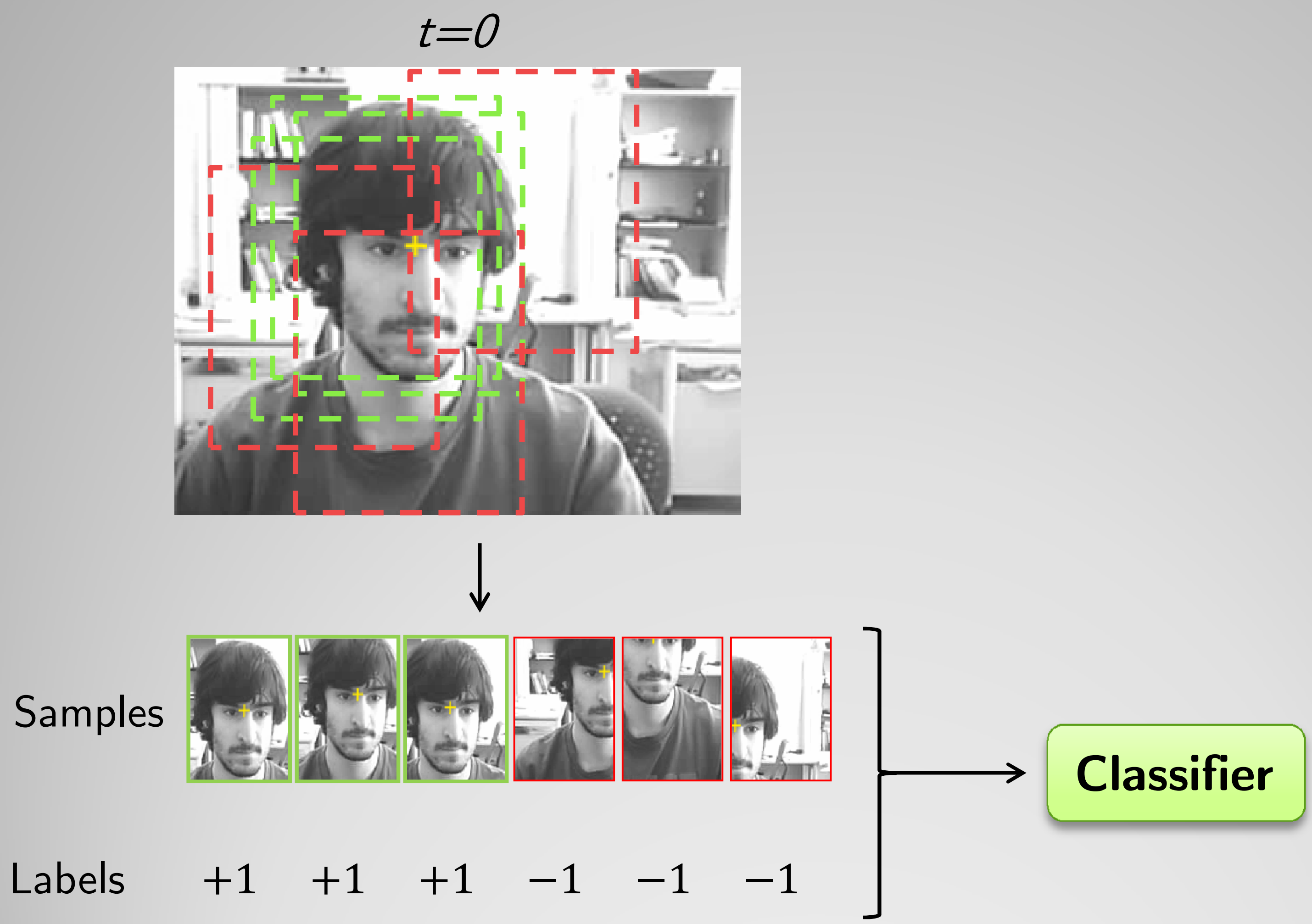
*t=0*



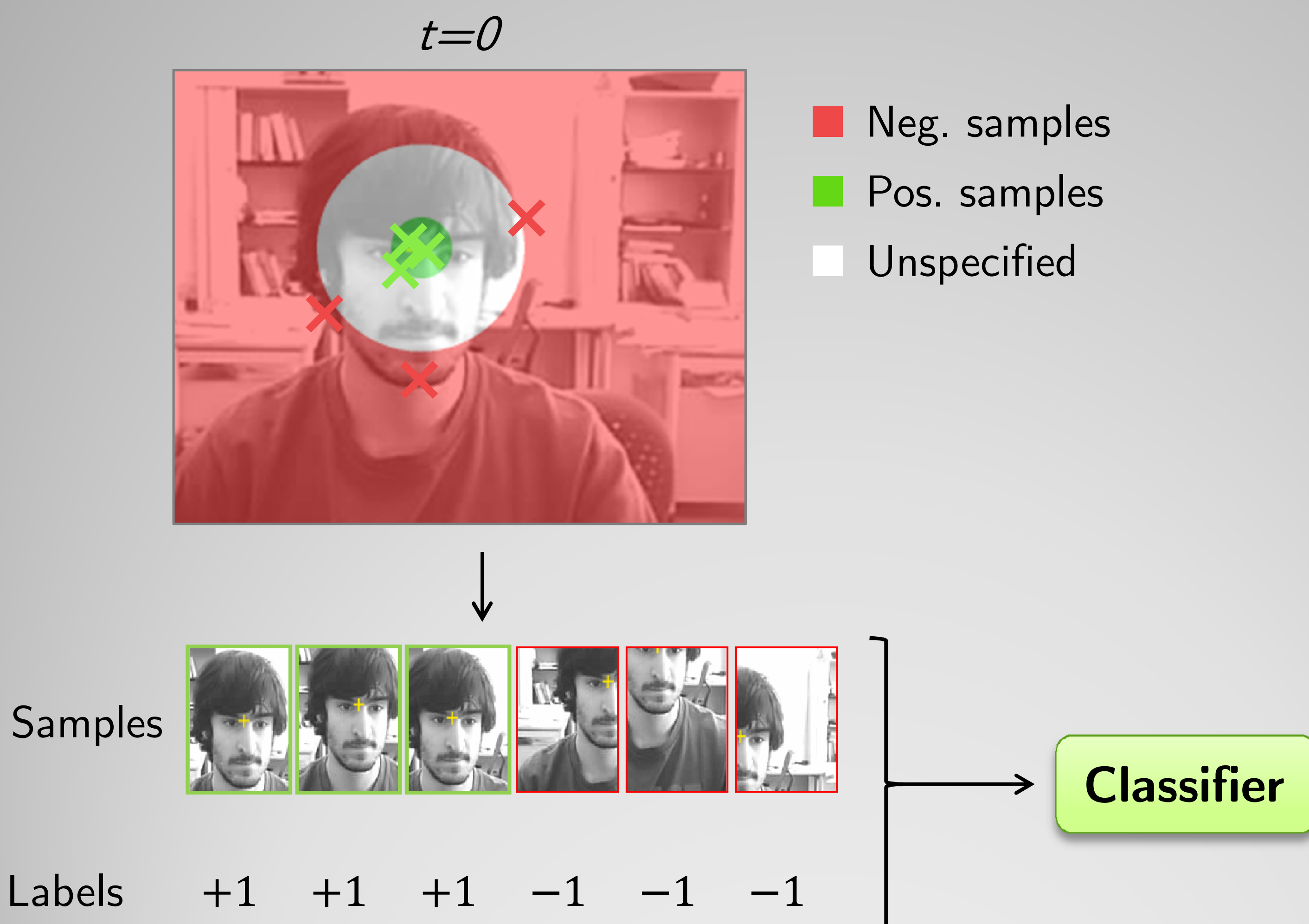
*t=1*

- We are given the **initial** bounding box (BB) of a target.
- **Estimate** its BB in later frames of a video ("track the target").
- Important component of many Computer Vision pipelines (simpler/faster than detection; ensures temporal consistency).
- Successfully tracked frames yield **more** information on target appearance.

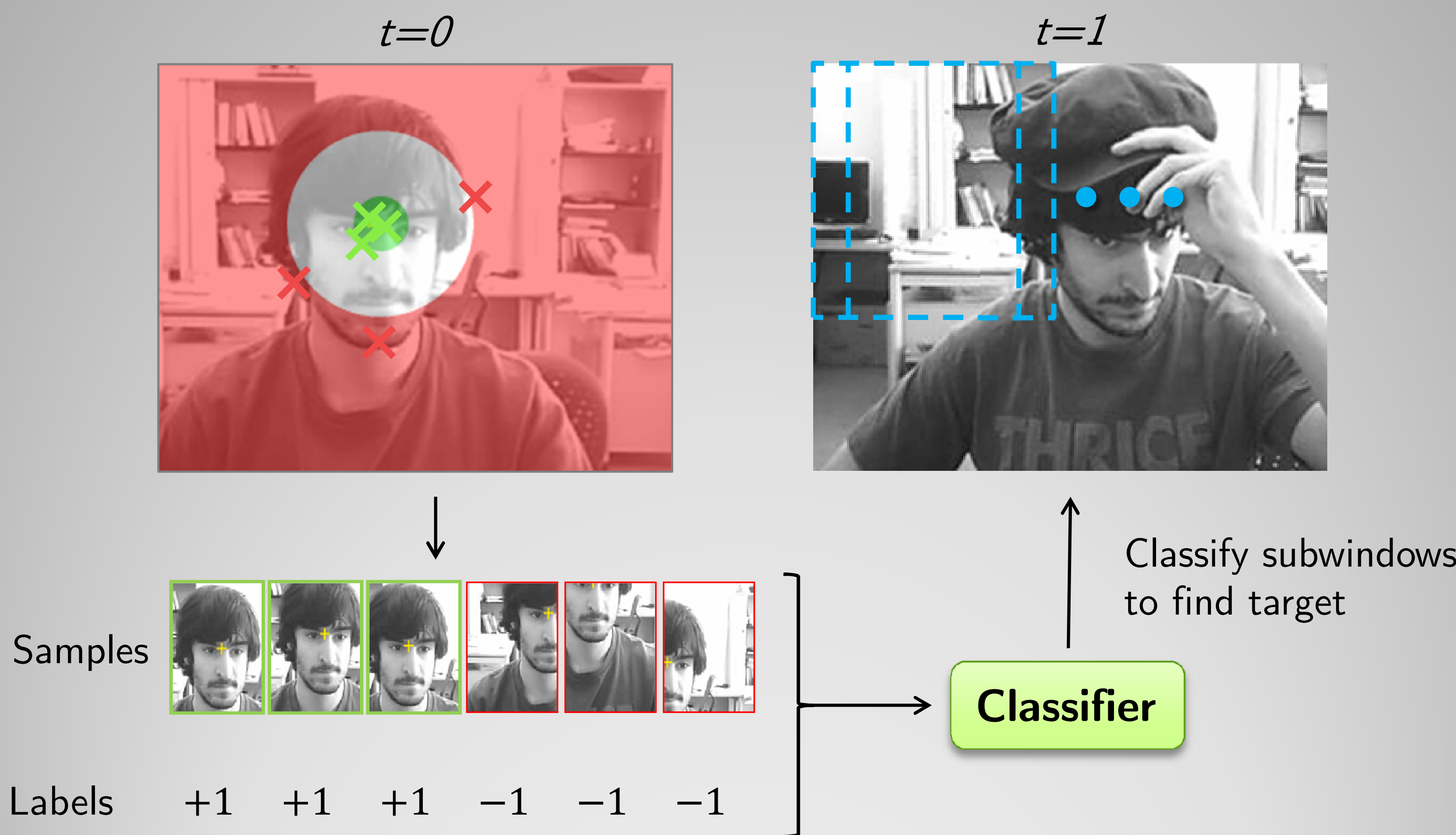
# Visual tracking – discriminative



# Visual tracking – discriminative



# Visual tracking – discriminative



# The Convolution Connection

- Linear classifier with weights $\mathbf{w}$ :

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

# The Convolution Connection

- Linear classifier with weights $\mathbf{w}$ :

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

- Evaluate it at subwindows $\mathbf{x}_i$ :

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

$i = 1$

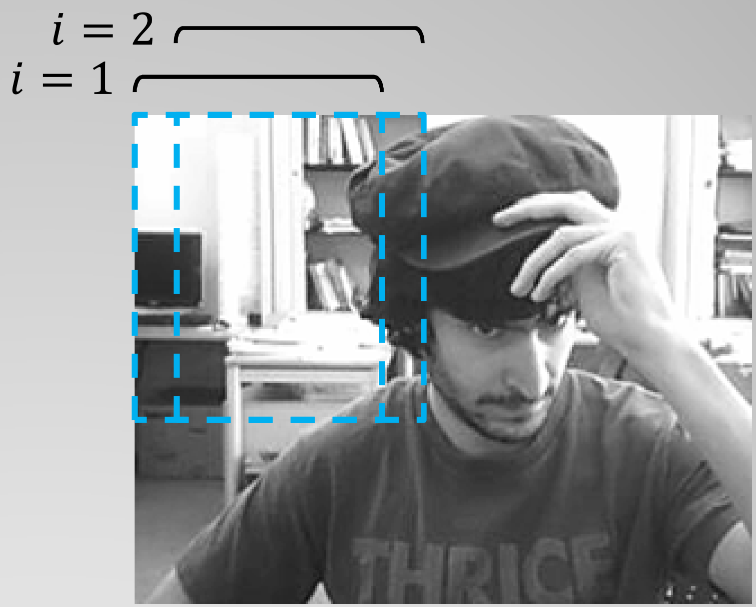
# The Convolution Connection

- Linear classifier with weights $\mathbf{w}$ :

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

- Evaluate it at subwindows $\mathbf{x}_i$ :

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



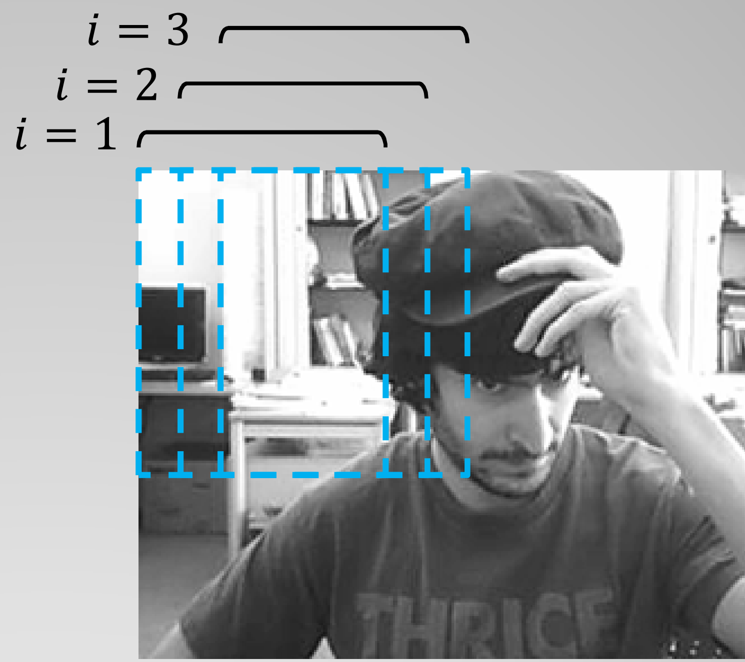
# The Convolution Connection

- Linear classifier with weights $\mathbf{w}$ :

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

- Evaluate it at subwindows $\mathbf{x}_i$ :

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



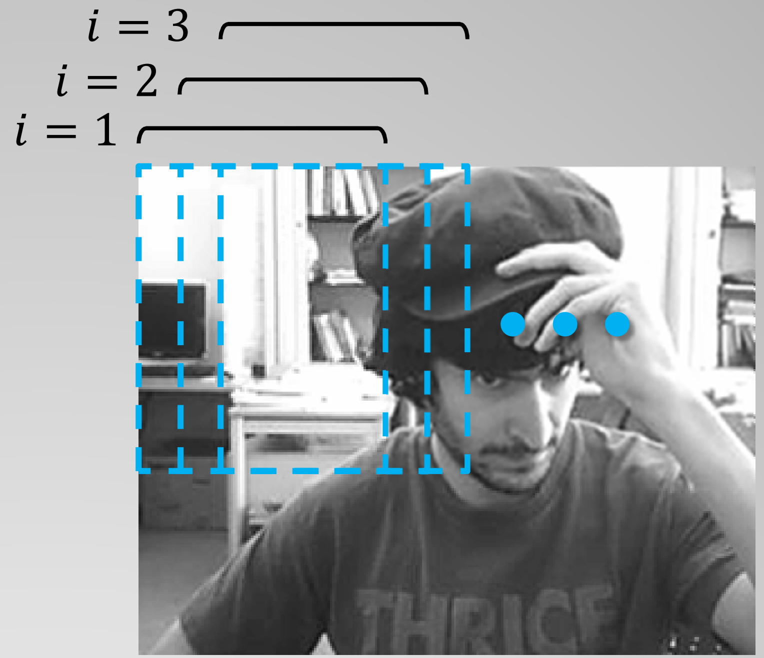
# The Convolution Connection

- Linear classifier with weights $\mathbf{w}$ :

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

- Evaluate it at subwindows $\mathbf{x}_i$ :

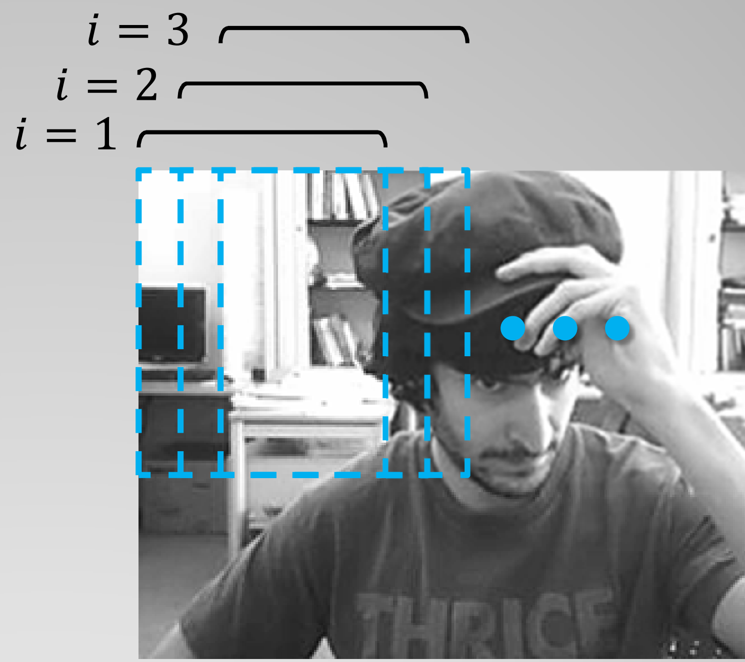
$$y_i = \mathbf{w}^T \mathbf{x}_i$$



# The Convolution Connection

- Linear classifier with weights $\mathbf{w}$ :

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




- Evaluate it at subwindows $\mathbf{x}_i$ :

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

- Concatenate $y_i$ into a vector $\mathbf{y}$.
- Equivalent to **cross-correlation** (or *correlation* for short)

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

- 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:

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

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

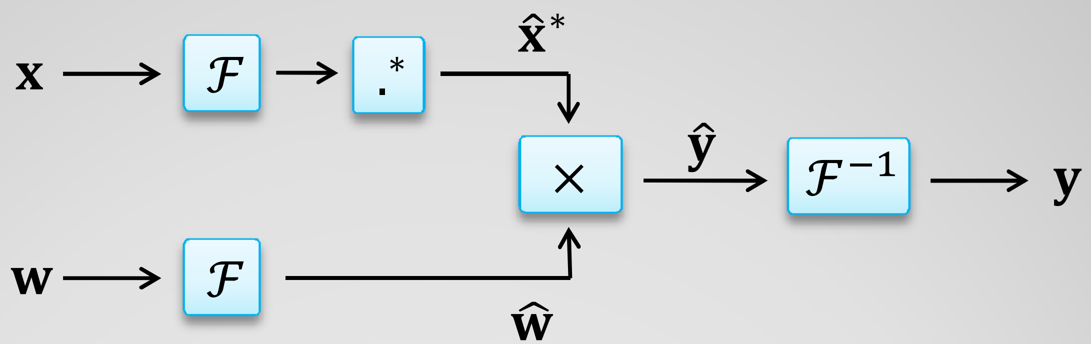
- Note that cross-correlation, and the DFT, are **cyclic** (the window wraps at the image edges).

# The Convolution Theorem

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

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

- In practice:




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

# The Convolution Theorem

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

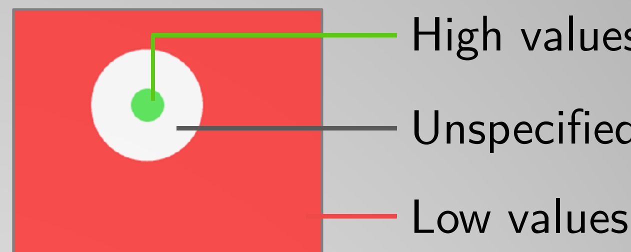
  (Not just for tracking.)

- What about **training**?

- It turns out that **Signal Processing** studied this problem for decades, almost separately from mainstream Computer Vision!

# Objective

$$\circledast \; \mathbf{w} \; =$$




**Intuition of training objective:**

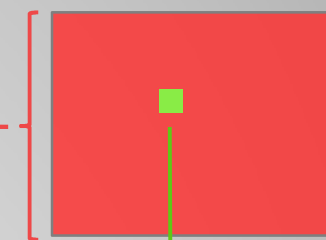
- Cross-correlation of classifier $\mathbf{w}$ and a training image $\mathbf{x}$ should have:
  - A **high peak** near the true location of the target.
  - **Low values** elsewhere (to minimize false positives).

# Historical Perspective

- **Minimum Average Correlation Energy** (MACE) filters (1980's)

  - Bring the average correlation output towards 0:

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




  - While keeping the peak value fixed:

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

  - The goal is to produce a **sharp peak** at the target location.

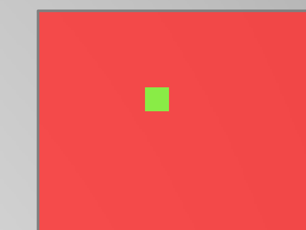
# Historical Perspective

- **Minimum Average Correlation Energy** (MACE) filters (1980's)
  - The solution is:

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

where division and product ($\times$) are element-wise.

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



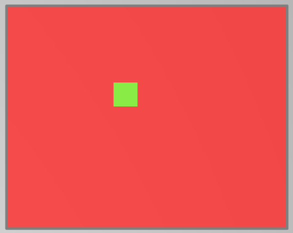
# Historical Perspective

- **Minimum Average Correlation Energy** (MACE) filters (1980's)
  - The solution is:

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

where division and product ($\times$) are element-wise.

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



  - $\hat{\mathbf{x}}^* \times \hat{\mathbf{x}}$ is called the **spectrum** and is real-valued.
  - Dividing by the spectrum is a common feature of many filters; it brings the auto-correlation to 0.
  - **Sharp peak = good localization**! Are we done?

# Historical Perspective

The MACE filter suffers from 2 issues:

- **Hard constraints easily lead to overfitting**.
    - UMACE ("Unconstrained MACE") attempts to solve this by instead maximizing the average classifier output on positive samples.
    - Unfortunately, it still suffers from the second problem...

# Historical Perspective

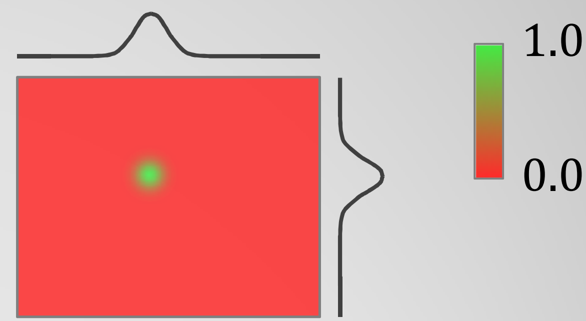
The MACE filter suffers from 2 issues:

- Enforcing a **sharp peak** is also a too strong condition; overfits.
    - This led to the development of **Gaussian-MACE / MSE-MACE**, which encourages the peak to follow a nice 2D Gaussian shape:

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

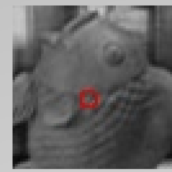
subject to: $\mathbf{w}^T\mathbf{x} = 1$

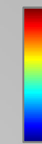
$\mathbf{g} =$




- In the original papers (1990's), the minimization was *still* subject to the MACE hard constraint: $\mathbf{w}^T\mathbf{x} = 1$ .
(It later turned out to be unnecessary!)

# Sharp vs. Gaussian peaks

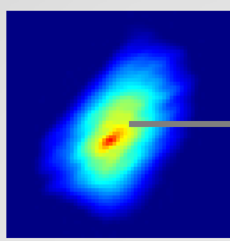
Training image: $\mathbf{x} =$ 




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

Classifier
$(\mathbf{w})$

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

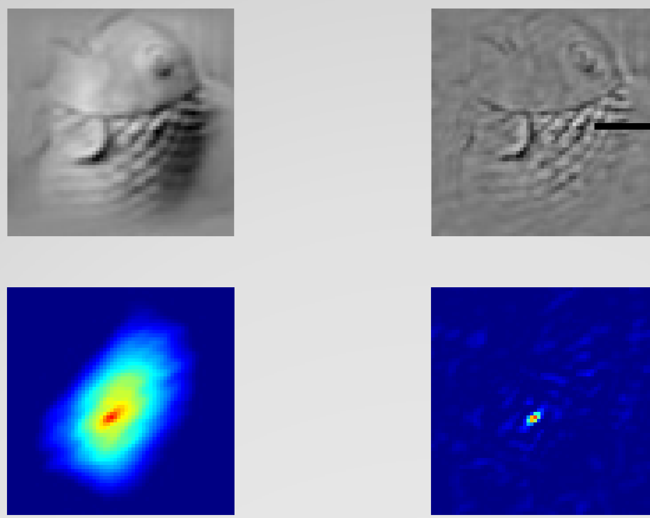


- Very broad peak is hard to localize (especially with clutter).
- State-of-the-art classifiers (e.g. SVM) show **same** behavior!

# Sharp vs. Gaussian peaks

Training image: $\mathbf{x} =$ 

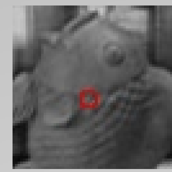

| | Naïve filter ($\mathbf{w} = \mathbf{x}$) | Sharp peak (UMACE) |
| --- | --- | --- |
| Classifier ($\mathbf{w}$) | | |
| Output ($\mathbf{w} * \mathbf{x}$) | | |

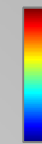


- A very sharp peak is obtained by emphasizing **small image details** (like the fish's scales here).

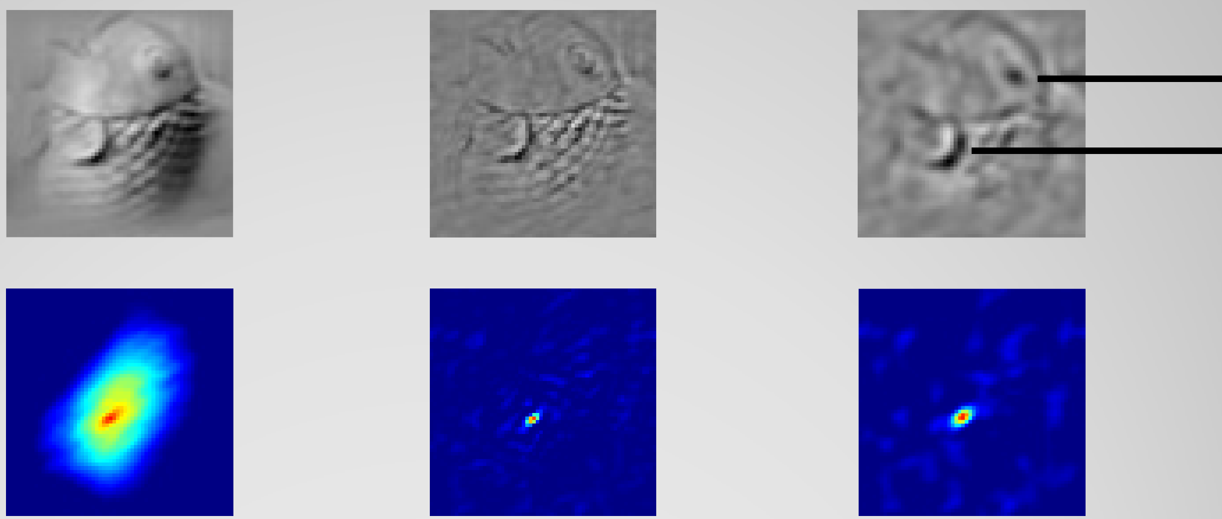
- Unfortunately, this classifier **generalizes poorly**:
  If the details are not exactly the same, the output is 0.

# Sharp vs. Gaussian peaks

Training image: $\mathbf{x} =$ 




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



- A Gaussian peak is a good compromise.
- Tiny details are ignored.
- Instead, the classifier focuses on **larger, more robust structures**.

# Min. Output Sum of Sq. Errors (MOSSE)

In their CVPR 2010 paper, David Bolme and colleagues brought these techniques back to the spotlight.

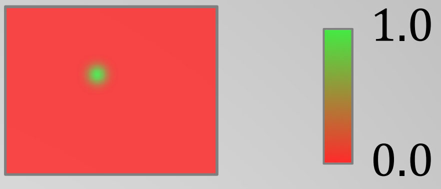
- They presented a tracker that:
  - Processed videos at over **600 frames-per-second** (!)
  - Was **very simple** to implement
    - No features.
    - Only FFT and element-wise operations on raw pixels.
  - Despite this fact, it performed **similarly** to the **most sophisticated** trackers of the time.

# Min. Output Sum of Sq. Errors (MOSSE)

**How did they do it?**

- They focused on just the "Gaussian peak" objective (no constraints):

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

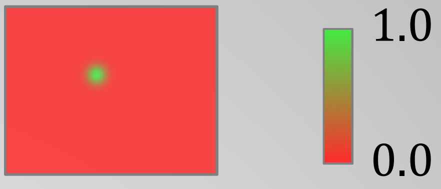


# Min. Output Sum of Sq. Errors (MOSSE)

**How did they do it?**

- They focused on just the "Gaussian peak" objective (no constraints):

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




- Found the following solution using the Convolution Theorem:

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

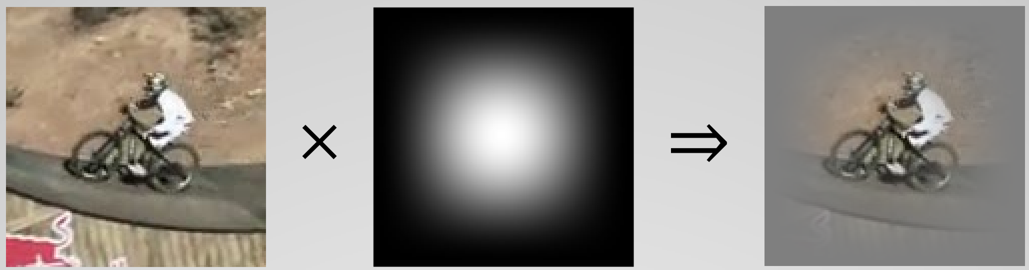
($\lambda = 10^{-4}$ is added to prevent divisions by 0)

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

# Min. Output Sum of Sq. Errors (MOSSE)

Practical aspects:

- **Cosine (or sine) window**



$$\mathbf{x}'_{rc} = (\mathbf{x}_{rc} - 0.5)\sin(\pi r/n)\sin(\pi c/n) \qquad \text{where } \mathbf{x} \text{ is } n \times n$$

- Smoothly interpolates image with a constant value at the edges.
- **The filter sees an image that is "cyclic"**:
  no discontinuity between edges (e.g. top and bottom).
- Bonus: gives more importance to the target center.

# Min. Output Sum of Sq. Errors (MOSSE)

Practical aspects:

- **Simple update**

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

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

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

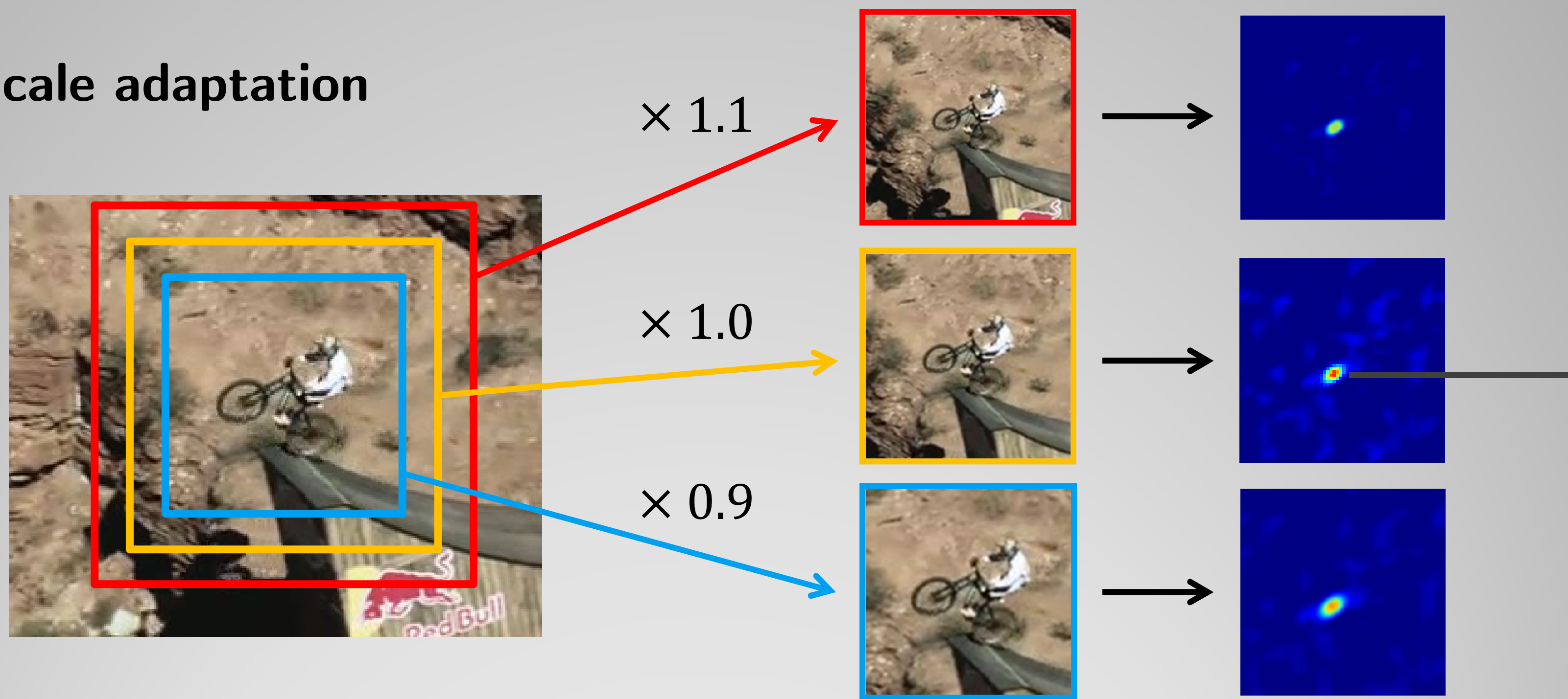
- Update previous solution $\hat{\mathbf{w}}_{t-1}$ with $\hat{\mathbf{w}}_{\text{new}}$ by linear interpolation.

- $\eta$ is the learning rate (higher → faster adaptation).
- This gives the tracker some memory.
- A variant is to update the numerator and denominator separately.

# Min. Output Sum of Sq. Errors (MOSSE)

Practical aspects:

- **Scale adaptation**




- Extract patches from BB's with 3 scales, resize them to the **same** size.
- Run detection, **use BB with the highest detection score**.
- Can also be adapted for rotation, and other transformations.

# Circulant matrices

**Why does the MOSSE filter work so well in practice?**

→ We need tools to connect **correlation filters** with **machine learning**.

- Consider the original minimization: $$\min_{\mathbf{w}} \|\mathbf{x} \circledast \mathbf{w} - \mathbf{g}\|^2$$

# Circulant matrices

**Why does the MOSSE filter work so well in practice?**

→ We need tools to connect **correlation filters** with **machine learning**.

- Consider the original minimization: $$\min_{\mathbf{w}} \|\mathbf{x} \circledast \mathbf{w} - \mathbf{g}\|^2$$

- We can **replace** the correlation with a **special matrix** $C(\mathbf{x})$: $$\boxed{\min_{\mathbf{w}} \|C(\mathbf{x})\mathbf{w} - \mathbf{g}\|^2}$$

- $C(\mathbf{x})$ is a **circulant matrix**:

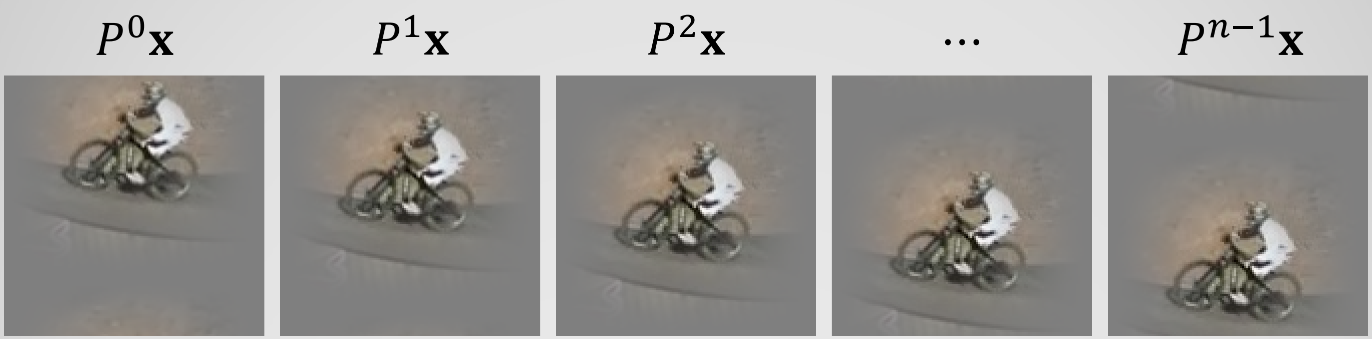
$$C(\mathbf{u}) = \begin{bmatrix} u_0 & u_1 & u_2 & \cdots & u_{n-1} \\ u_{n-1} & u_0 & u_1 & \cdots & u_{n-2} \\ u_{n-2} & u_{n-1} & u_0 & \cdots & u_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & u_3 & \cdots & u_0 \end{bmatrix}$$

# Circulant matrices

- We can see $X = C(\mathbf{x})$ as a **dataset** with **cyclically shifted** versions of $\mathbf{x}$:

$$X = \begin{bmatrix} (P^0\mathbf{x})^T \\ (P^1\mathbf{x})^T \\ \vdots \\ (P^{n-1}\mathbf{x})^T \end{bmatrix}$$

- $P$ is a permutation matrix that shifts the pixels down 1 element.
- Arbitrary shift $i$ obtained with power $P^i\mathbf{x}$.
- Cyclic: $P^n\mathbf{x} = P^0\mathbf{x} = \mathbf{x}$.



# Circulant matrices

- Circulant matrices have many nice properties.

$$X = \underbrace{\begin{bmatrix} (P^0\mathbf{x})^T \\ (P^1\mathbf{x})^T \\ \vdots \\ (P^{n-1}\mathbf{x})^T \end{bmatrix}}_{\text{Data matrix is circulant}} \Rightarrow \mathcal{F}(X) = \underbrace{\begin{bmatrix} \hat{\mathbf{x}}_1 & 0 & \cdots & 0 \\ 0 & \hat{\mathbf{x}}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{\mathbf{x}}_n \end{bmatrix}}_{\text{Becomes diagonal in Fourier domain}}$$

Data matrix is **circulant** ⇒ Becomes **diagonal** in Fourier domain

- Similar role to the Convolution Theorem.
- **Most of the "data" is 0 and can be ignored!** ⇒ Massive speed-up

# Circulant matrices

Back to our question:
**Why does the MOSSE filter work so well in practice?**

- Consider a simple Ridge Regression (RR) problem:

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

RR = Least-squares with regularization (avoids overfitting!)

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

# Circulant matrices

Back to our question:
**Why does the MOSSE filter work so well in practice?**

- Consider a simple Ridge Regression (RR) problem:

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

RR = Least-squares with regularization (avoids overfitting!)

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

- Now replace X $= C(\mathbf{x})$ (circulant data), and $\mathbf{y} = \mathbf{g}$ (Gaussian targets).
- **Diagonalizing** the involved circulant matrices with the DFT yields:

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

# Circulant matrices

Back to our question:
**Why does the MOSSE filter work so well in practice?**

- Consider a simple Ridge Regression (RR) problem:

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

RR = Least-squares with regularization (avoids overfitting!)

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

- Now replace $\mathrm{X} = C(\mathbf{x})$ (circulant data), and $\mathbf{y} = \mathbf{g}$ (Gaussian targets).
- **Diagonalizing** the involved circulant matrices with the DFT yields:

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

- Which is exactly the MOSSE solution!
- So MOSSE is equivalent to a **good learning algorithm** (RR) with **lots of data** (circulant/shifted samples).

# Kernelized Correlation Filters

- Circulant matrices are a **very general tool**, replacing standard operations with fast Fourier operations.

- For example, we can apply the same idea to **Kernel Ridge Regression**:

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

- For many kernels, circulant data ⇒ circulant $K$:

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

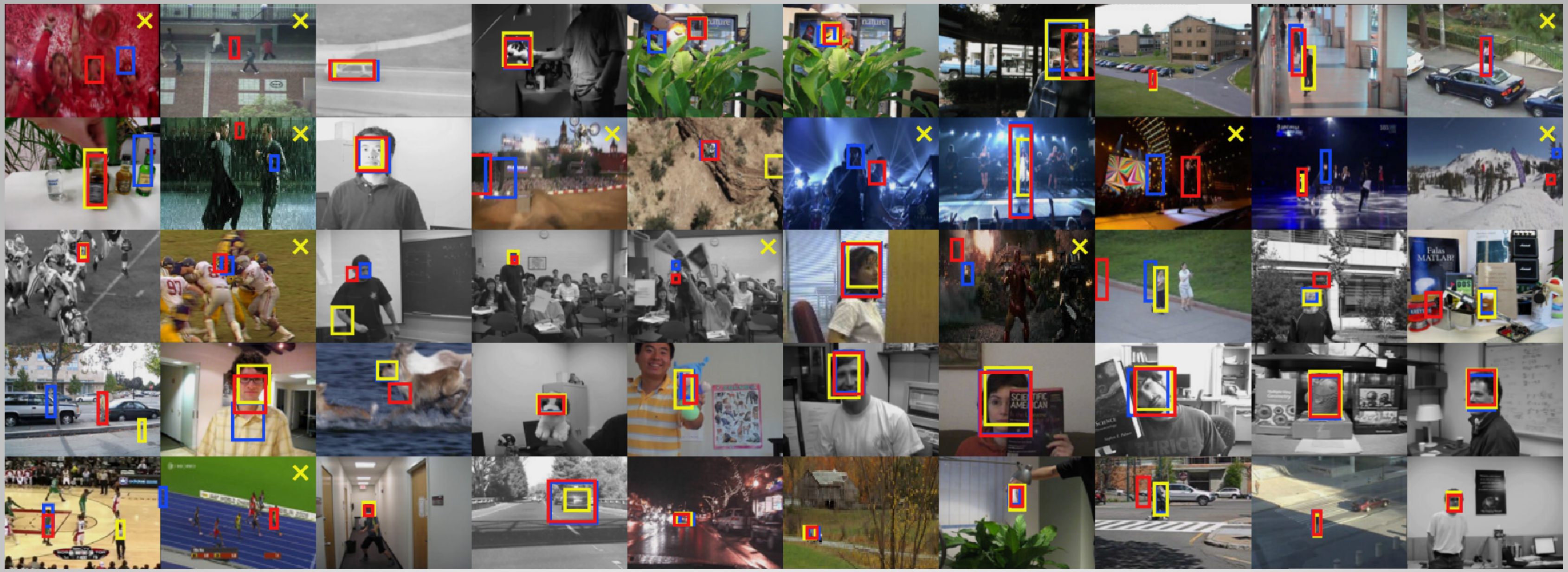
where $\mathbf{k}$ is the first row of $K$ (small, and easy to compute)

- Diagonalizing with the DFT yields:

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

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

# KCF – Qualitative results

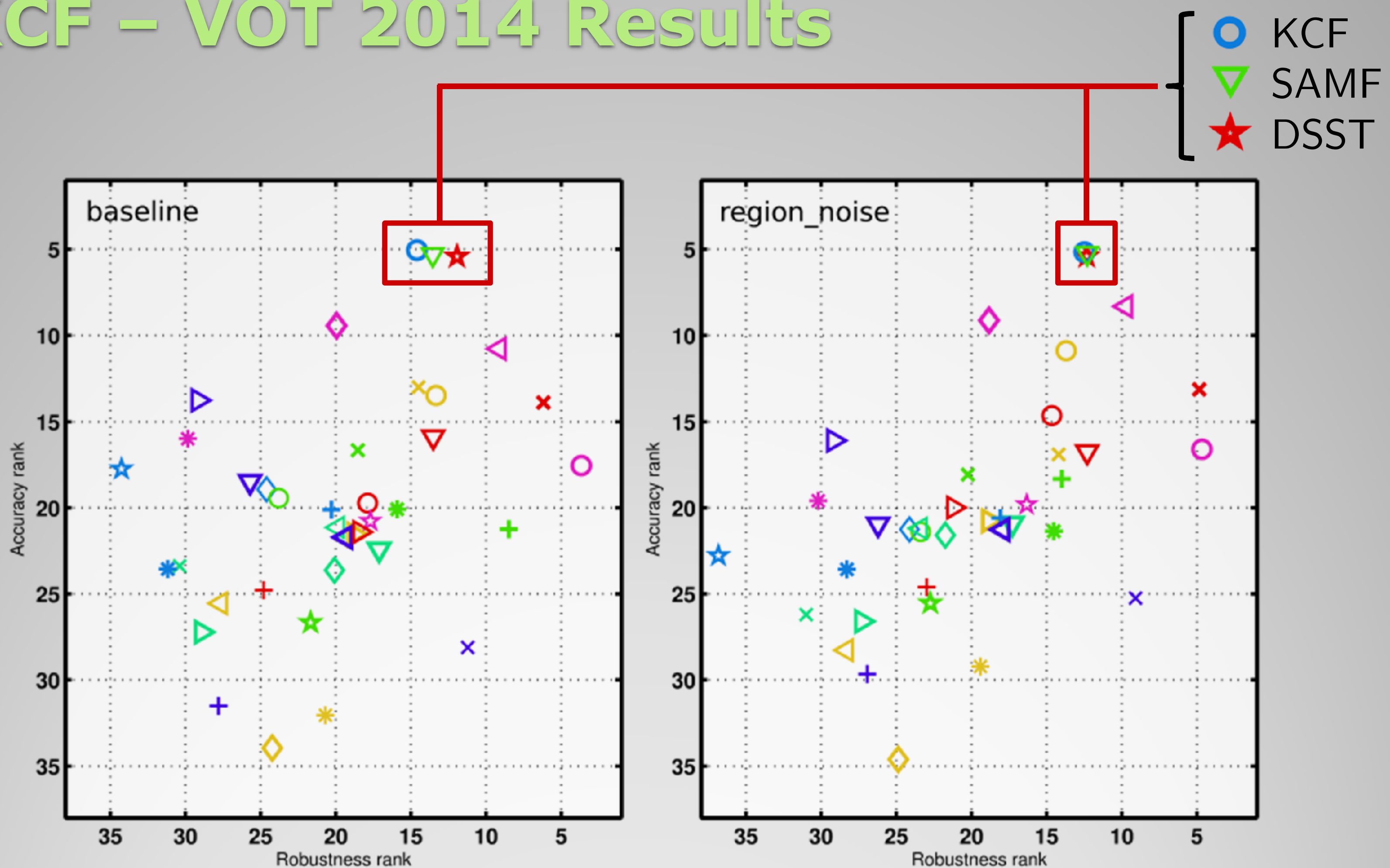


Kernelized Correlation Filter (KCF) TLD Struck

**KCF Tracker**

- Open-source (ported to Matlab/Python/Java/C)
- $\sim$ **300 FPS**
- Code base for **top 3 trackers in VOT 2014**.

# KCF – VOT 2014 Results




**KCF Tracker**

- Open-source (ported to Matlab/Python/Java/C)
- $\sim$ **300 FPS**
- Code base for **top 3 trackers in VOT 2014**.

# KCF – Source code

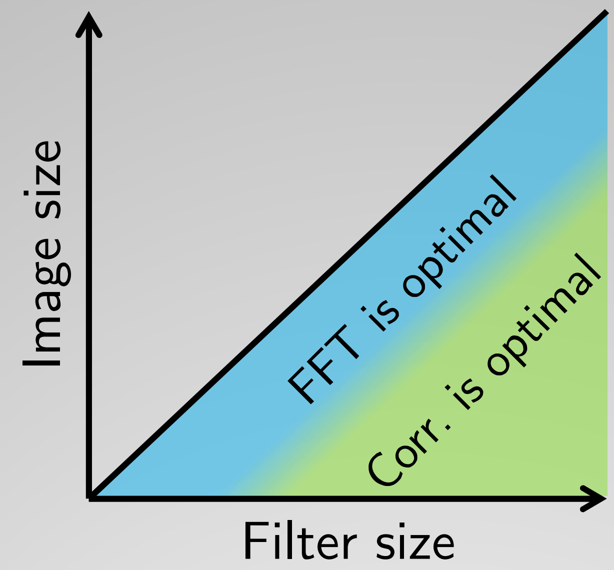
**Training and detection (Matlab)**

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

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

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

**KCF Tracker**

- Very few hyperparameters.
- **Fits on the back of a postcard**, native Matlab functions.

# Practical considerations




- As a rule of thumb, **similarly sized** cross-correlation arguments (e.g. image and filter) take the **best** advantage of the FFT.

- Consider a $n \times n$ image and a $f \times f$ filter.
  - FFT complexity is $\mathcal{O}(n^2 \log n)$ (*independent* of $f$, big or small!).
  - Cross-correlation complexity is $\mathcal{O}(n^2 f^2)$ (better when $f \ll n$).

# Practical considerations

- When performing FFTs, the "classic advice" is to set the image size to a power-of-two if possible:

$$\text{size}(\mathbf{x}) = 2^r \times 2^s, \quad \text{with integer } r, s.$$

- While this theoretically achieves the best speed, modern FFT libraries (such as FFTW) are **optimized for arbitrary sizes**.

- Rounding the size up to the next power-of-two has 2 drawbacks:

  - A mismatched size can **degrade** recognition performance (e.g. by including unnecessary background regions in a filter).

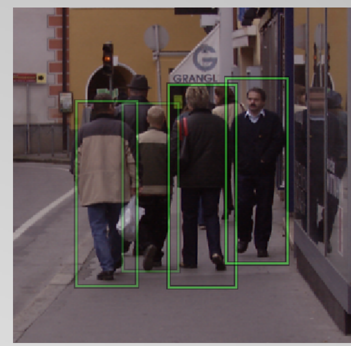
  - If the next power-of-two is significantly larger, we can end up with actually **slower** FFTs!

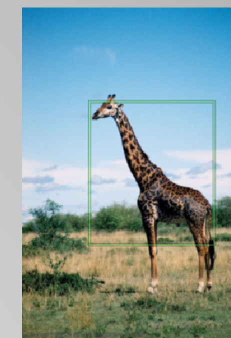
# Other topics

Topics not covered here:

- Considering multiple samples and features simultaneously.
- Circulant trick for other algorithms (Support Vector Regression, etc).

*ICCV'13 (example detections)*





- Generalizing shifts to other transformations (rotations, etc).
- Fast training of classifier ensemble (pose estimator).

*NIPS'14 (example pose estimates)*