

Using translation in learning-based vision

Instructor - Jack Valmadre

Maths for AI - 2022



**AUSTRALIAN INSTITUTE
FOR MACHINE LEARNING**



**THE UNIVERSITY
of ADELAIDE**

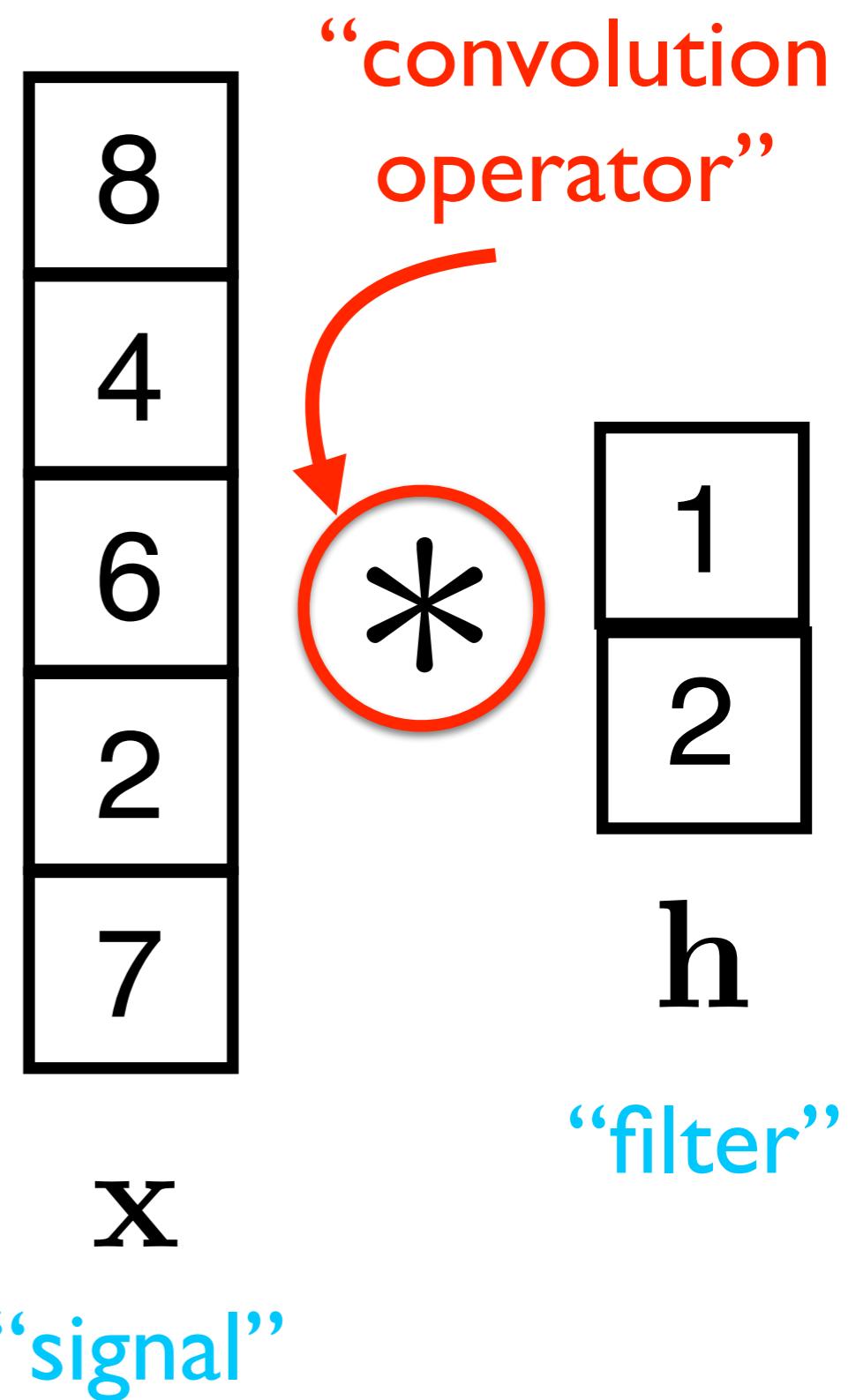
Today

- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks

Today

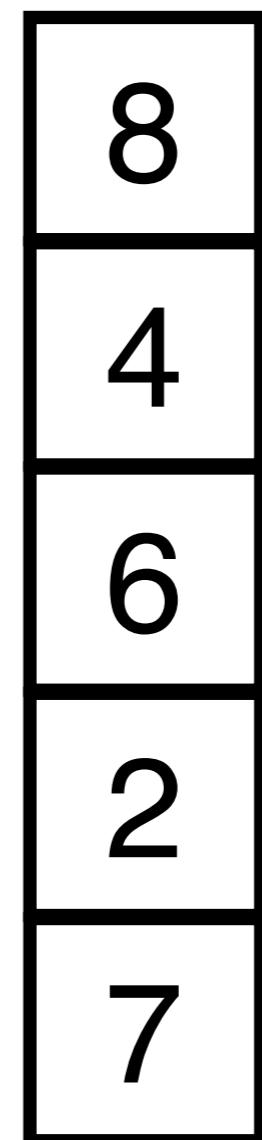
- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks

Reminder: Convolution



Reminder: Convolution

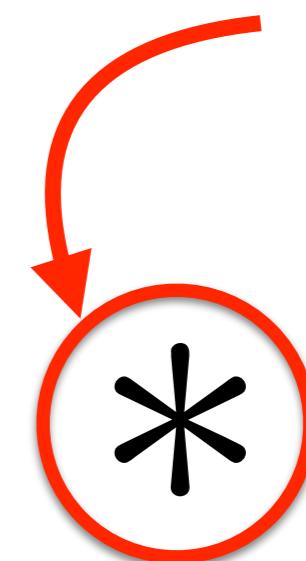
```
>>> from scipy.signal import \
convolve as conv  
  
>>> conv(x, h, 'valid')  
array([20, 14, 14, 11])
```



x

“signal”

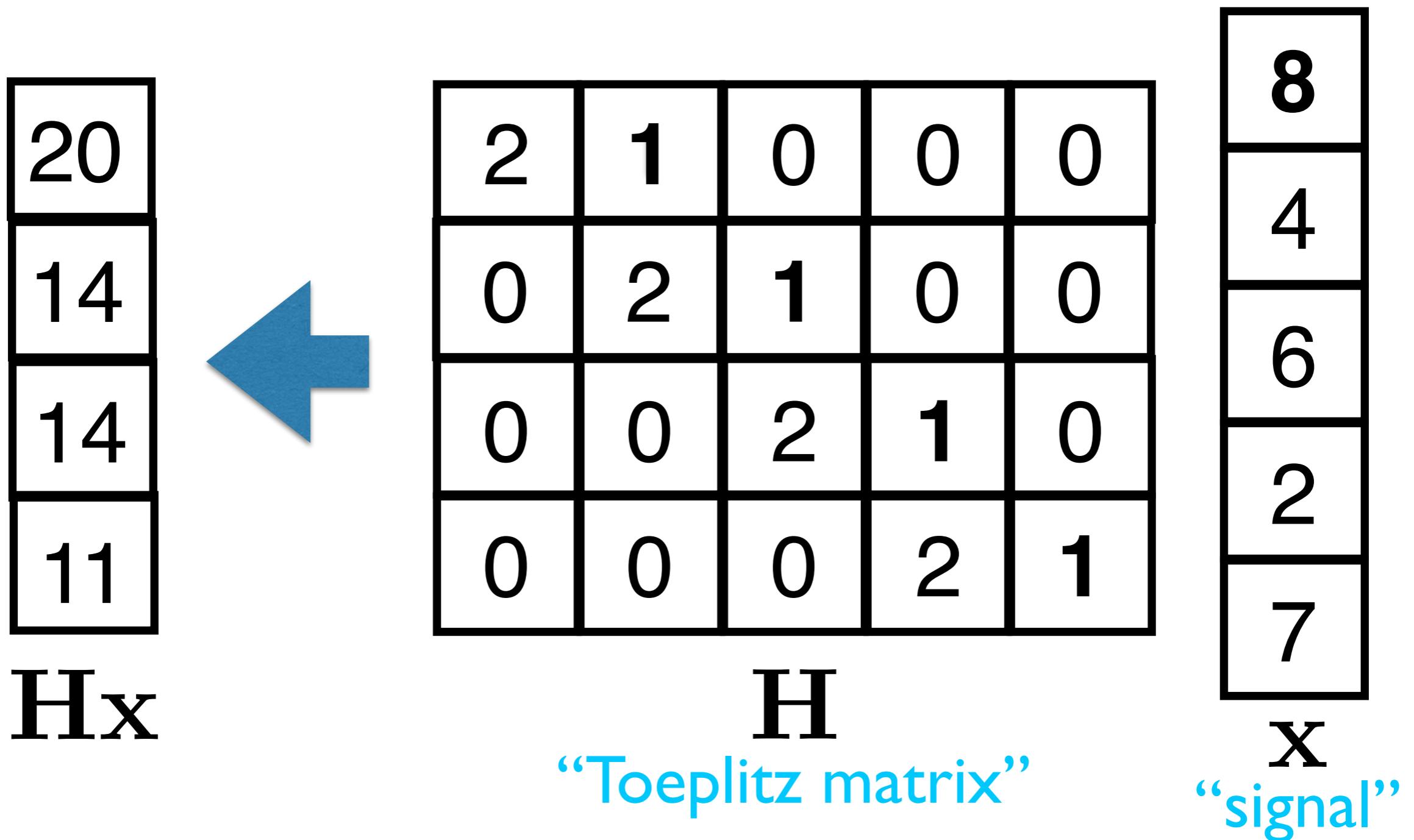
“convolution operator”



h

“filter”

Reminder: Convolution



Boundary conditions

- More than one type of convolutional operator:

```
>>> conv(x, h, 'valid')  
>>> conv(x, h, 'same')  
>>> conv(x, h, 'full')
```

Boundary conditions

- More than one type of convolutional operator:

```
>>> conv(x, h, 'valid')  
>>> conv(x, h, 'same')  
>>> conv(x, h, 'full')
```

- What does H look like for ‘full’ convolution?

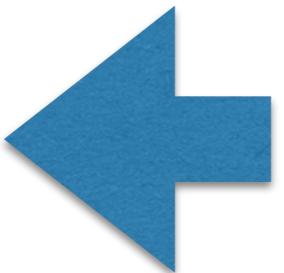
- How about ‘same’?

'valid'

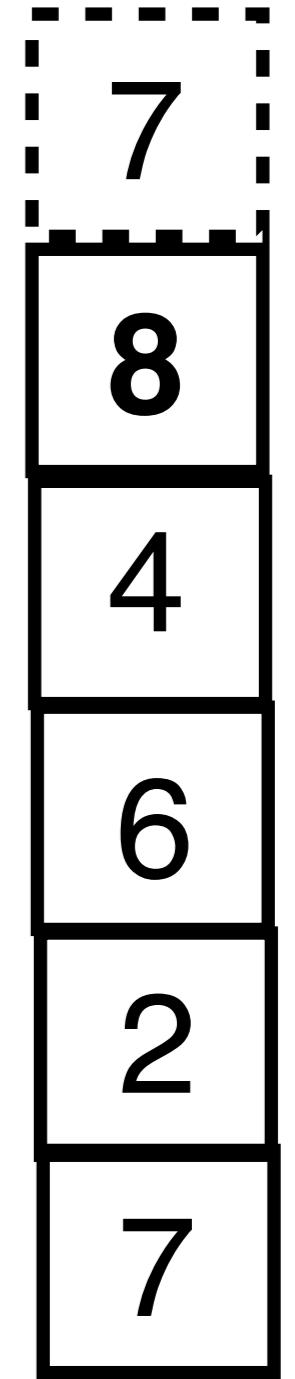
2	1	0	0	0
0	2	1	0	0
0	0	2	1	0
0	0	0	2	1

Circular convolution

22
20
14
14
11

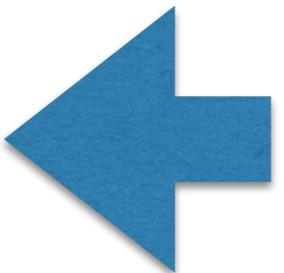


2	1	0	0	0	0
0	2	1	0	0	0
0	0	2	1	0	0
0	0	0	2	1	0
0	0	0	0	2	1



Circular convolution

22
20
14
14
11



1	0	0	0	2	8
2	1	0	0	0	4
0	2	1	0	0	6
0	0	2	1	0	2
0	0	0	2	1	7

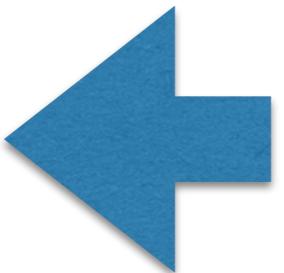
H

x

“circulant matrix”

Circular convolution

22
20
14
14
11



8	7	2	6	4	1
4	8	7	2	6	2
6	4	8	7	2	0
2	6	4	8	7	0
7	2	6	4	8	0

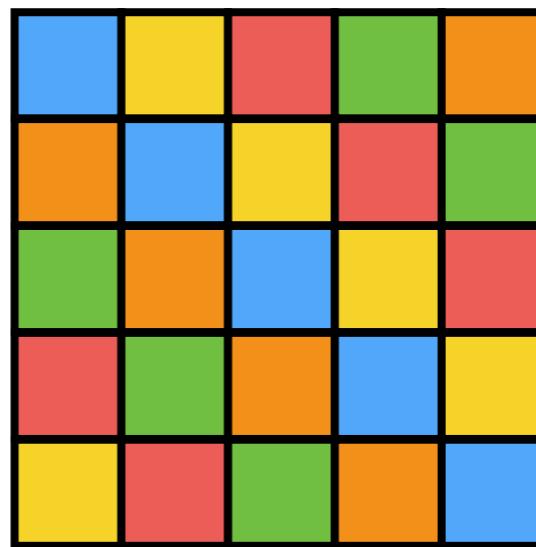
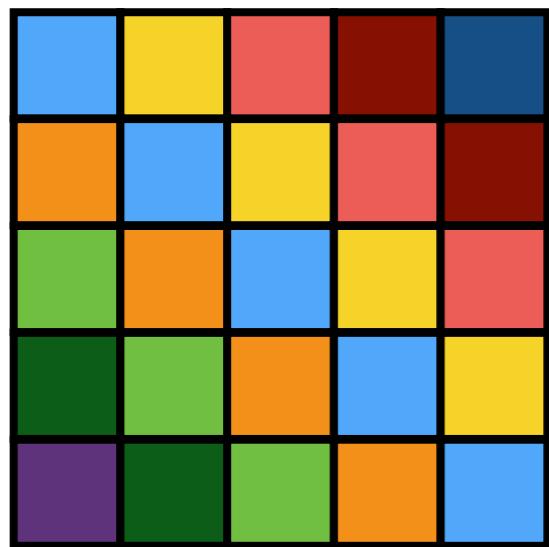
X

h

“circulant matrix”

Toeplitz and circulant matrices

$(D \times D)$

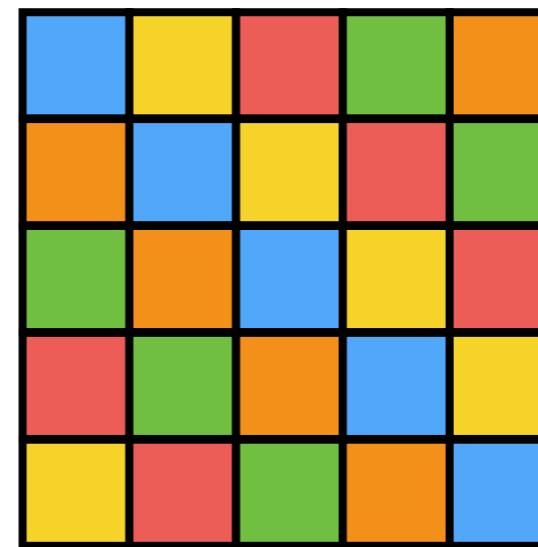
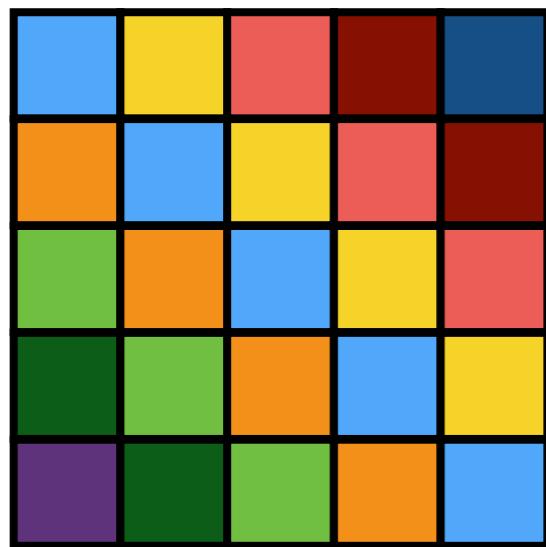


$$X_{ij} = x[j - i]$$

$$X_{ij} = x[j - i \bmod D] \quad \mathbf{x} \in \mathbb{R}^D$$

Toeplitz and circulant matrices

$(D \times D)$



$$X_{ij} = x[j - i]$$

$$X_{ij} = x[j - i \bmod D] \quad \mathbf{x} \in \mathbb{R}^D$$

>>> $X[m, n] = x[\bmod(n-m, D)]$

Discrete Fourier Transform

- Fourier transform

$$\hat{x}(\xi) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi\xi t} dt$$

Discrete Fourier Transform

- Fourier transform

$$\hat{x}(\xi) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi\xi t} dt$$

- Discrete Fourier Transform (DFT)

$$\hat{x}[k] = \sum_{t=0}^{D-1} x[t] e^{-i2\pi \frac{kt}{D}}$$

Discrete Fourier Transform

- Fourier transform

$$\hat{x}(\xi) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi\xi t} dt$$

- Discrete Fourier Transform (DFT)

$$\hat{x}[k] = \sum_{t=0}^{D-1} x[t] e^{-i2\pi \frac{kt}{D}} \quad \hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$$

$$\mathbf{F} = \left(e^{-i2\pi \frac{kt}{D}} \right)_{k,t}$$

Discrete Fourier Transform

- Fourier transform

$$\hat{x}(\xi) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi\xi t} dt$$

- Discrete Fourier Transform (DFT)

$$\hat{x}[k] = \sum_{t=0}^{D-1} x[t] e^{-i2\pi \frac{kt}{D}} \quad \hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$$

$$\mathbf{F} = \left(e^{-i2\pi \frac{kt}{D}} \right)_{k,t}$$

Just a linear transform!

Convolution theorem

$$\mathcal{F}(x * a) = (\mathcal{F}x) \circ (\mathcal{F}a)$$

	Time	Frequency	Conv. thm.
Fourier transform	continuous; $x(t)$	continuous; $X(\xi)$	“linear”
Fourier series	continuous; $x(t)$	discrete; X_k	periodic
DTFT	discrete; $x[t]$	continuous; $X(\xi)$	“linear”
DFT	discrete; $x[t]$	discrete; $X[k]$	periodic

Isomorphism

Convolution

$$y = x * a$$

Time reversal \pmod{D}

$$y = Jx$$

$$Jx[t] \triangleq x[-t]$$

Real

$$\text{Im}\{x\} = 0$$

Product

$$\hat{y} = \hat{x} \circ \hat{a}$$

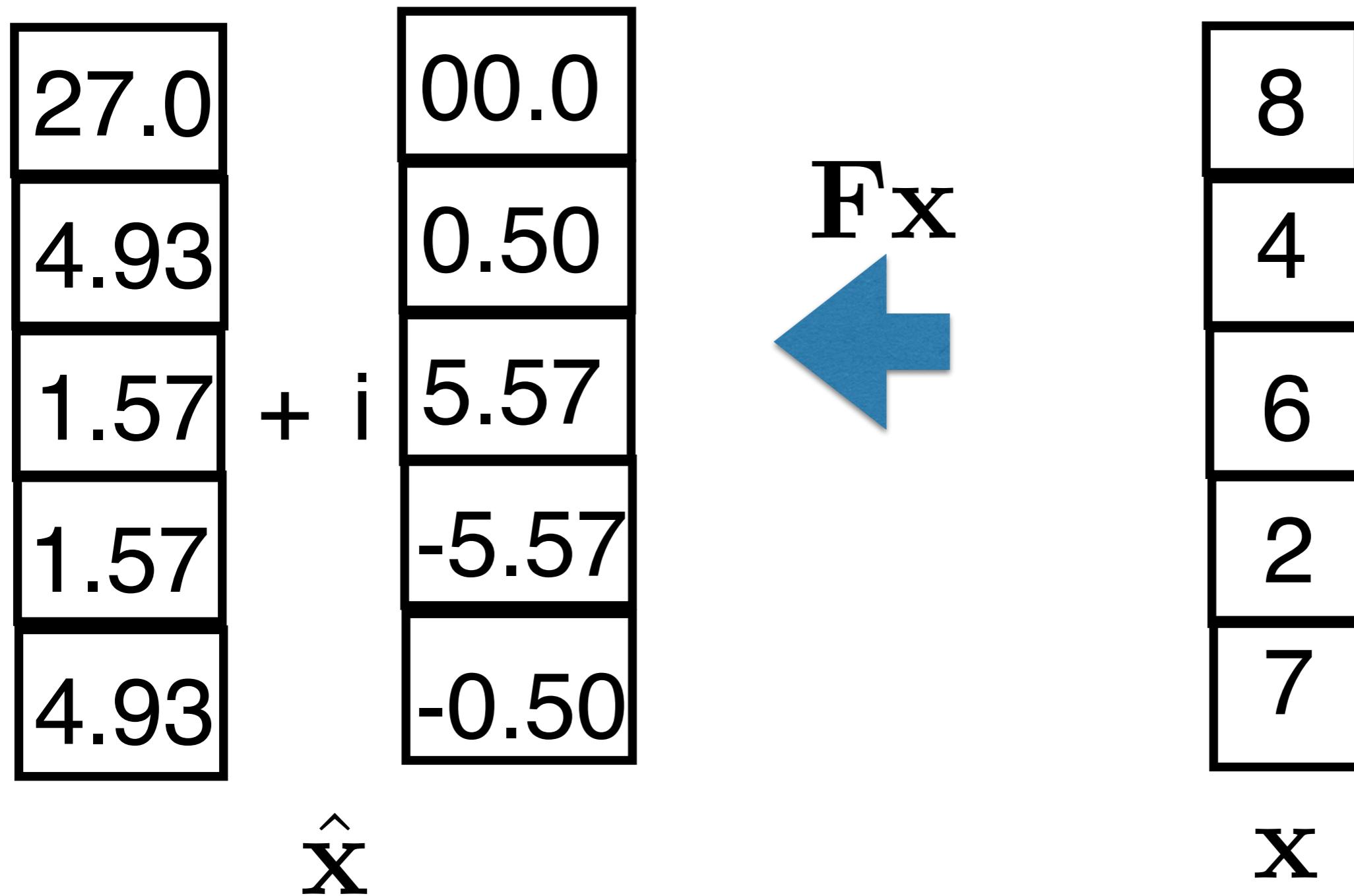
Conjugate

$$\hat{y} = \hat{x}^*$$

Conj. symm. \pmod{D}

$$\hat{x}[t] = \hat{x}^*[-t]$$

Note: DFT is complex



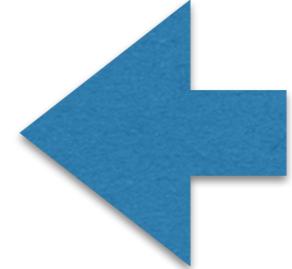
Example: Time reversal

27.0
4.93
1.57
1.57
4.93

- i

00.0
0.50
5.57
-5.57
-0.50

F_x



8
7
2
6
4

$\text{conj}(\hat{x})$

>>> `flipud(roll(x, 4))`

Isomorphism

Cross-correlation

$$y = x \otimes a = Jx * a$$

Conjugate product

$$\hat{y} = \hat{x}^* \circ \hat{a}$$

Isomorphism

Cross-correlation

$$y = x \otimes a = Jx * a$$

Conjugate product

$$\hat{y} = \hat{x}^* \circ \hat{a}$$

Question: How is $x \otimes a$ related to $a \otimes x$?

Isomorphism

Cross-correlation

$$y = x \otimes a = Jx * a$$

Conjugate product

$$\hat{y} = \hat{x}^* \circ \hat{a}$$

Question: How is $x \otimes a$ related to $a \otimes x$?

$$\mathcal{F}(a \otimes x) = \hat{a}^* \circ \hat{x} =$$

Isomorphism

Cross-correlation

$$y = x \otimes a = Jx * a$$

Conjugate product

$$\hat{y} = \hat{x}^* \circ \hat{a}$$

Question: How is $x \otimes a$ related to $a \otimes x$?

$$\mathcal{F}(a \otimes x) = \hat{a}^* \circ \hat{x} = (\hat{a} \circ \hat{x}^*)^* =$$

Isomorphism

Cross-correlation

$$y = x \otimes a = Jx * a$$

Conjugate product

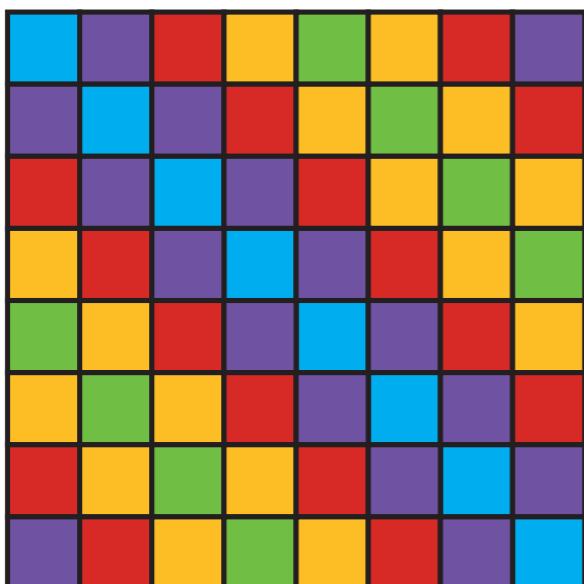
$$\hat{y} = \hat{x}^* \circ \hat{a}$$

Question: How is $x \otimes a$ related to $a \otimes x$?

$$\mathcal{F}(a \otimes x) = \hat{a}^* \circ \hat{x} = (\hat{a} \circ \hat{x}^*)^* = \mathcal{F}(J(x \otimes a))$$

Diagonalising circulant matrices

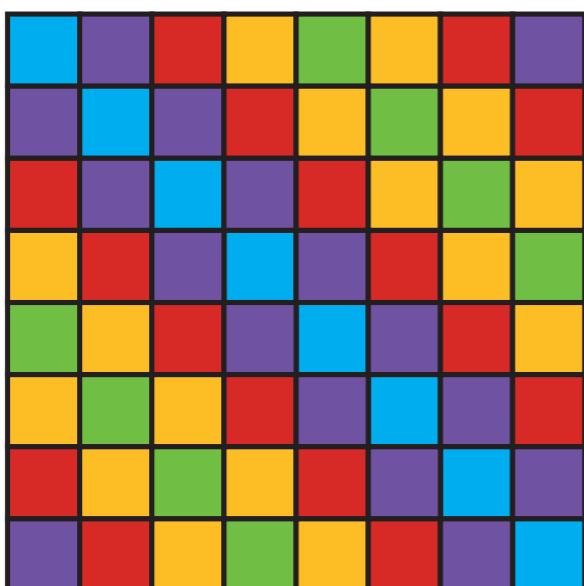
$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{a} * \mathbf{x} = \mathbf{F}^{-1} \mathbf{F}(\mathbf{a} * \mathbf{x}) \\ &= \mathbf{F}^{-1}(\hat{\mathbf{a}} \circ \hat{\mathbf{x}}) = \mathbf{F}^{-1} \operatorname{diag}(\hat{\mathbf{a}}) \mathbf{F}\mathbf{x}\end{aligned}$$



A

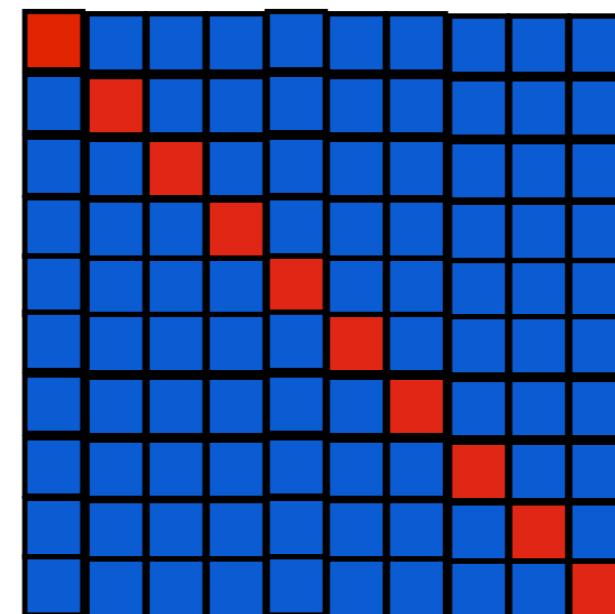
Diagonalising circulant matrices

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{a} * \mathbf{x} = \mathbf{F}^{-1} \mathbf{F}(\mathbf{a} * \mathbf{x}) \\ &= \mathbf{F}^{-1} (\hat{\mathbf{a}} \circ \hat{\mathbf{x}}) = \mathbf{F}^{-1} \operatorname{diag}(\hat{\mathbf{a}}) \mathbf{F} \mathbf{x}\end{aligned}$$



A

→ F

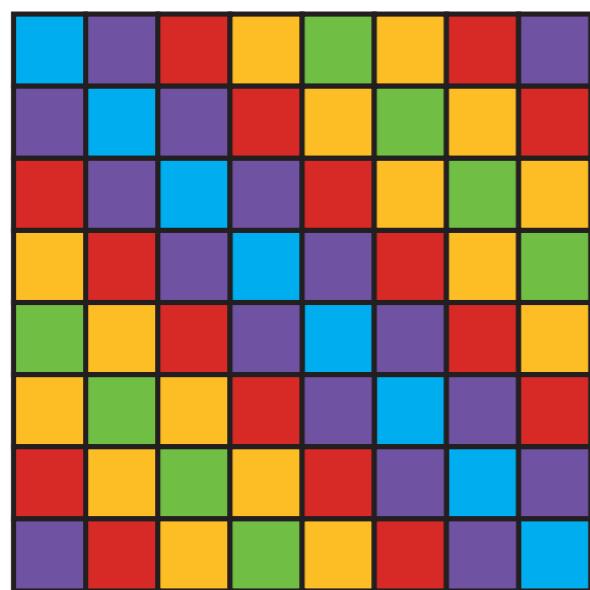


F⁻¹

■ Always Zero

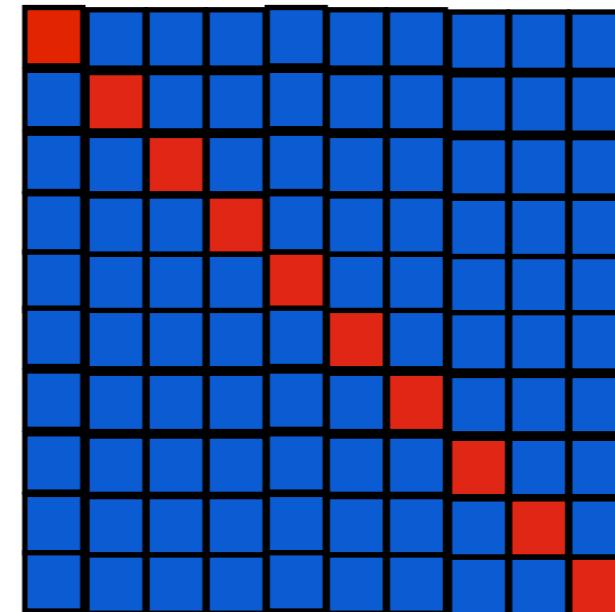
Diagonalising circulant matrices

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{a} * \mathbf{x} = \mathbf{F}^{-1} \mathbf{F}(\mathbf{a} * \mathbf{x}) \\ &= \mathbf{F}^{-1} (\hat{\mathbf{a}} \circ \hat{\mathbf{x}}) = \mathbf{F}^{-1} \text{diag}(\hat{\mathbf{a}}) \mathbf{F} \mathbf{x}\end{aligned}$$



A

→ F



F⁻¹

■ Always Zero

All circulant matrices have the *same eigenvectors!*
Eigenvalues given by Fourier transform

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

$$A = A^T \iff a[t] = a[-t]$$

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

$$\mathbf{A} = \mathbf{A}^T \iff a[t] = a[-t]$$
$$\iff \hat{\mathbf{a}}^* = \hat{\mathbf{a}}$$

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

$$A = A^T \iff a[t] = a[-t]$$

$$\iff \hat{a}^* = \hat{a} \text{ real (eigenvalues)}$$

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

$$A = A^T \iff a[t] = a[-t]$$

$$\iff \hat{a}^* = \hat{a} \text{ real (eigenvalues)}$$

$$a \text{ real} \iff \hat{a}[k] = \hat{a}^*[-k]$$

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

$$A = A^T \iff a[t] = a[-t]$$

$$\iff \hat{a}^* = \hat{a} \text{ real (eigenvalues)}$$

$$a \text{ real} \iff \hat{a}[k] = \hat{a}^*[-k] = \hat{a}[-k]$$

Diagonalising circulant matrices

**Question: What if A is real and symmetric?
Shouldn't its eigendecomposition be real?**

$$A = A^T \iff a[t] = a[-t]$$

$$\iff \hat{a}^* = \hat{a} \text{ real (eigenvalues)}$$

$$a \text{ real} \iff \hat{a}[k] = \hat{a}^*[-k] = \hat{a}[-k]$$

Eigenvalues $k \notin \{0, D/2\}$ have *multiplicity two*
 \Rightarrow Eigenvectors are not unique

Fast Fourier Transform

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

The diagram illustrates the computation of the Fast Fourier Transform (FFT). On the left, a vector $\hat{\mathbf{x}}$ is shown as a vertical stack of four red squares. An equals sign follows. To the right is a matrix multiplication expression: $\mathbf{F} \cdot \cdots \cdot \mathbf{F} \cdot \mathbf{x}$. The matrix \mathbf{F} is represented by a grid of 16 squares. The first row has two red squares at positions (1,1) and (2,1). Subsequent rows show a repeating pattern where each row has one red square at position (i,1) and all other squares are blue. Ellipses indicate the matrix continues horizontally. The vector \mathbf{x} is shown as a vertical stack of four red squares, identical to $\hat{\mathbf{x}}$.



Carl Friedrich Gauss

- Not Always Zero
- Always Zero

$\mathcal{O}(D \log D)$

```
>>> xf = fft(x);
```

FFT as a linear transform

- Using fftmtx.py (see link),

```
>>> F = fftmtx(5)
>>> x = randn(5); h = randn(5);
```

- Apply the FFT in the naive and fast methods.

```
>>> xf_a = dot(F, x)
>>> xf_b = fft(x)
```

- You should see that `xf_a` and `xf_b` are the same.

FFT as a linear transform

- Using fftmtx.py (see link),

```
>>> F = fftmtx(5)
>>> x = randn(5); h = randn(5);
```

- Apply the FFT in the naive and fast methods.

```
>>> xf_a = dot(F, x)
>>> xf_b = fft(x)
```

- You should see that `xf_a` and `xf_b` are the same.

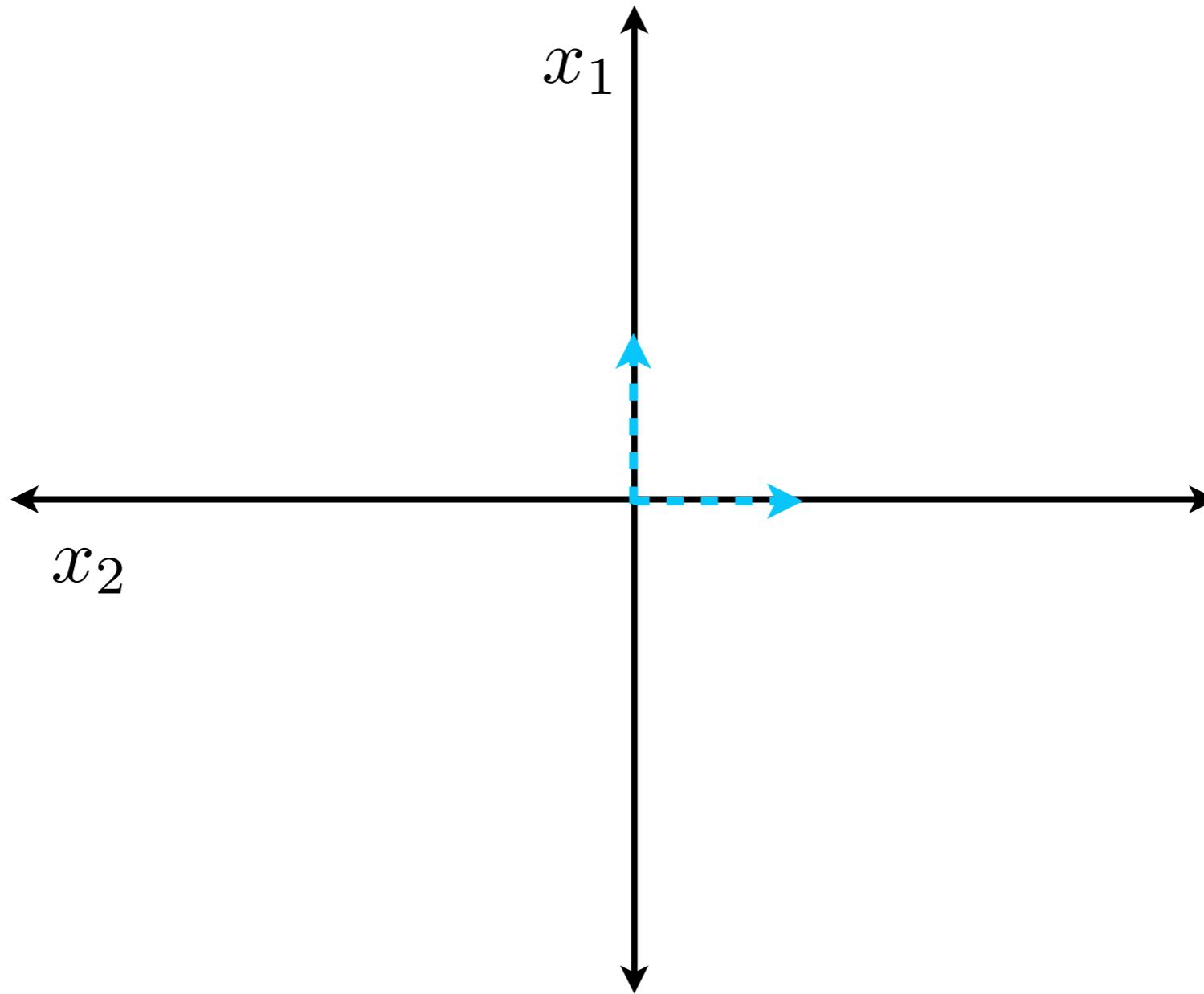
Question: what does `F.H @ F` look like?

Parseval's Theorem



Antoine Parseval

$$\|\mathbf{x}\|_2^2 \propto \|\hat{\mathbf{x}}\|_2^2$$

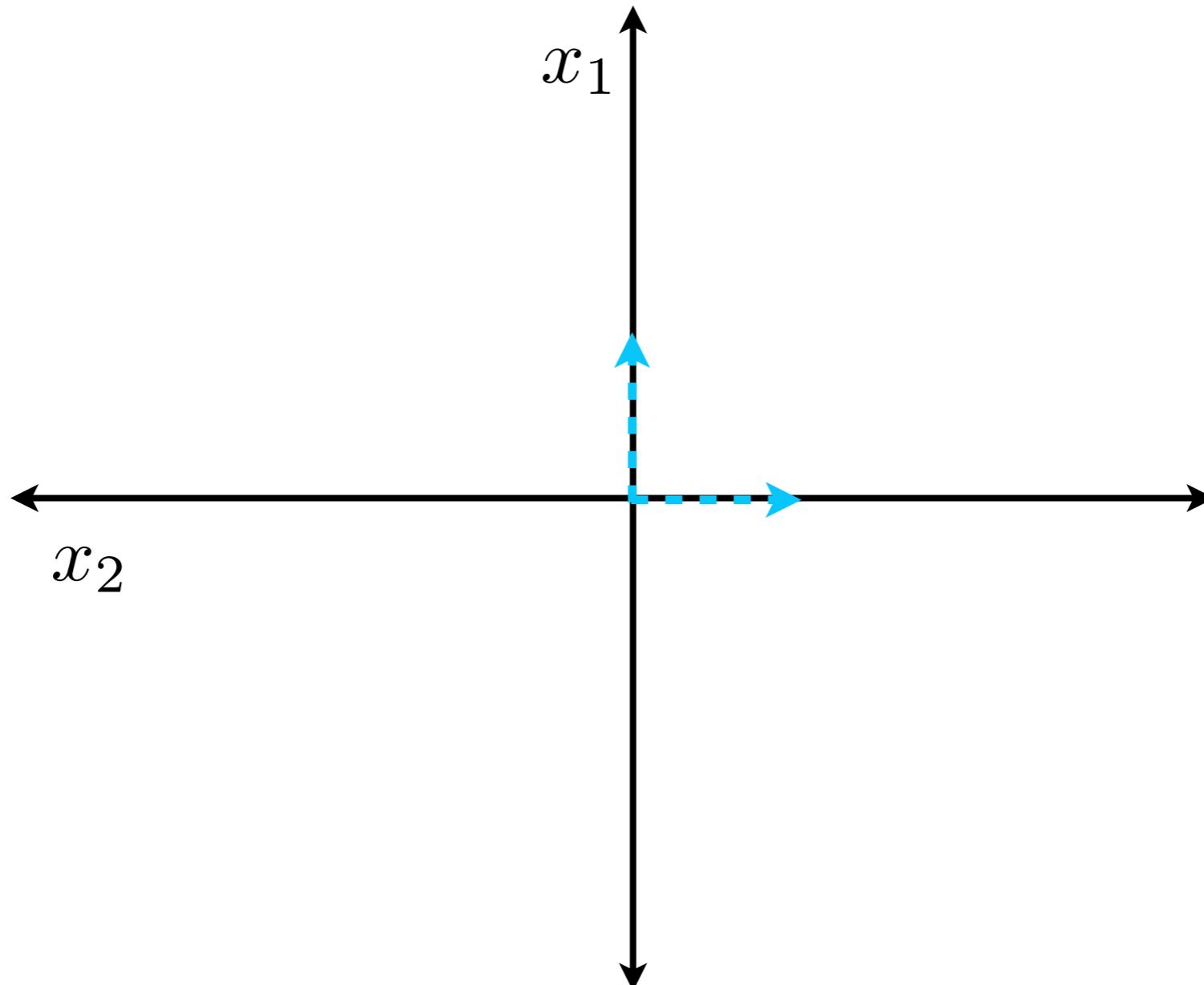


Parseval's Theorem



Antoine Parseval

$$\|\mathbf{x}\|_2^2 = \frac{1}{D} \|\hat{\mathbf{x}}\|_2^2$$

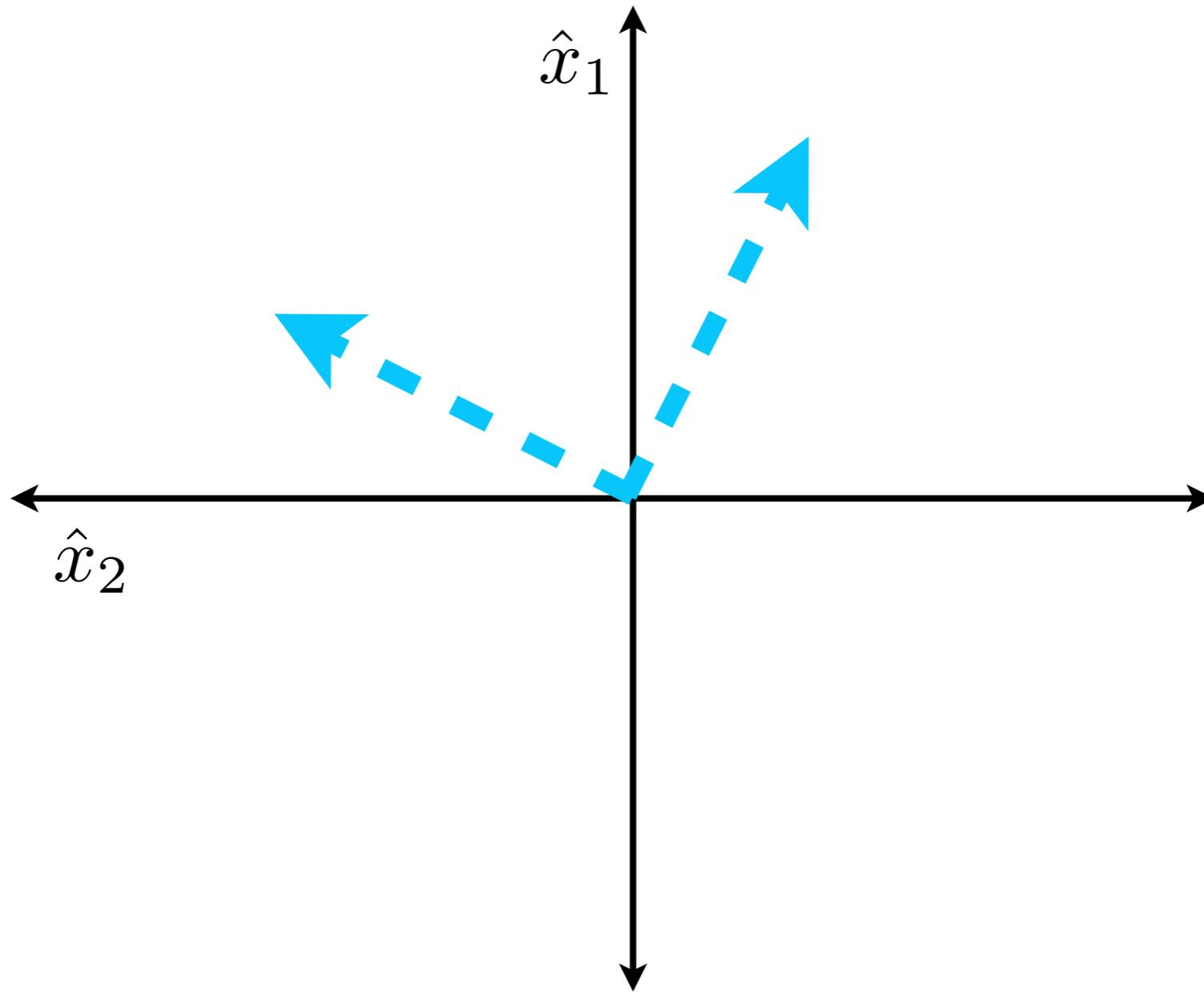


Parseval's Theorem



Antoine Parseval

$$\| \mathbf{x} \|_2^2 = \frac{1}{D} \| \hat{\mathbf{x}} \|_2^2$$

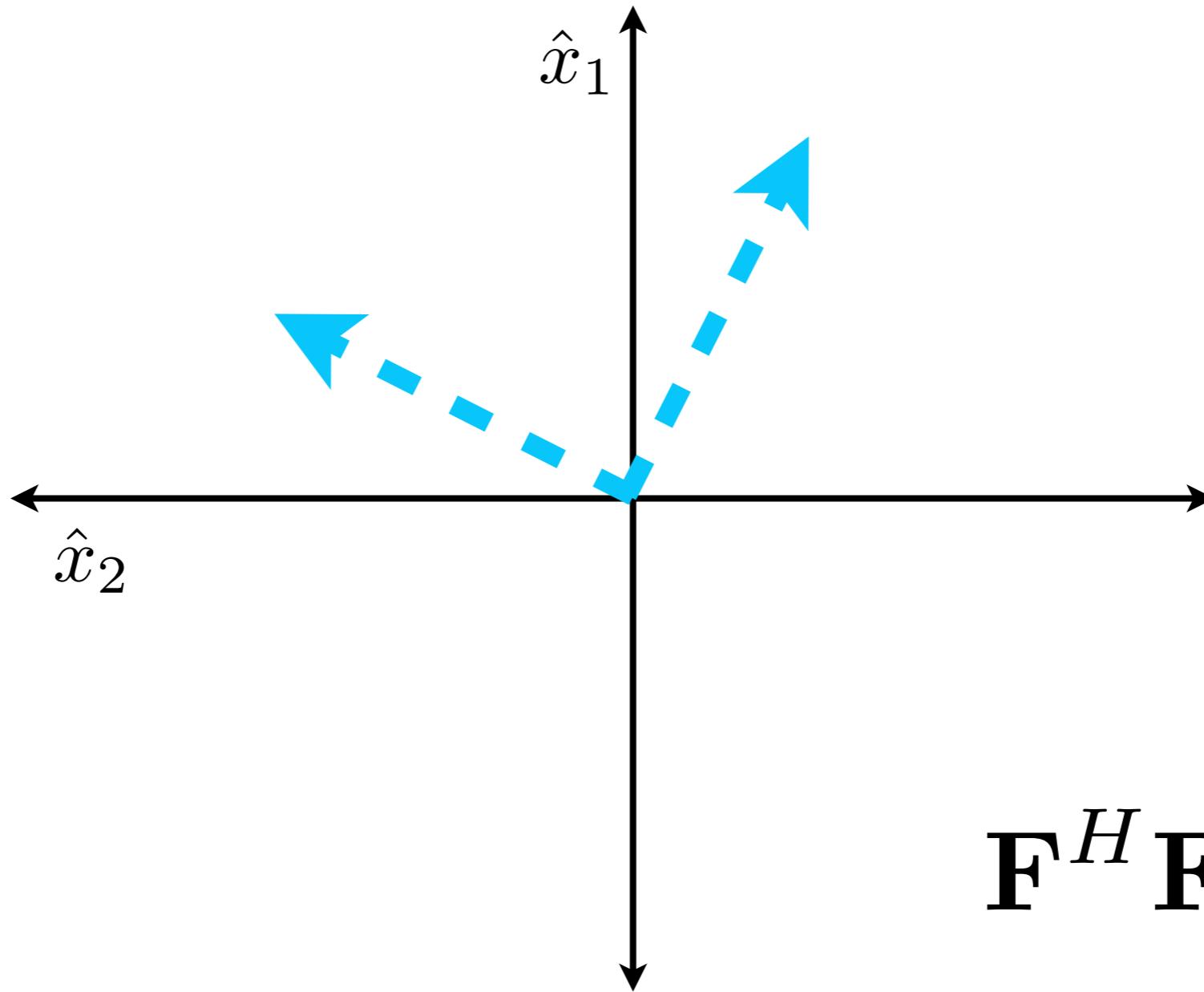


Parseval's Theorem



Antoine Parseval

$$\|\mathbf{x}\|_2^2 = \frac{1}{D} \|\hat{\mathbf{x}}\|_2^2$$



$$\mathbf{F}^H \mathbf{F} = D \mathbf{I}$$

N-dim DFT

- Everything above applies for N-dim Fourier transform and N-dim convolution
- Treat signals as real vectors in $\mathbb{R}^{[D_1] \times [D_2] \times \dots}$

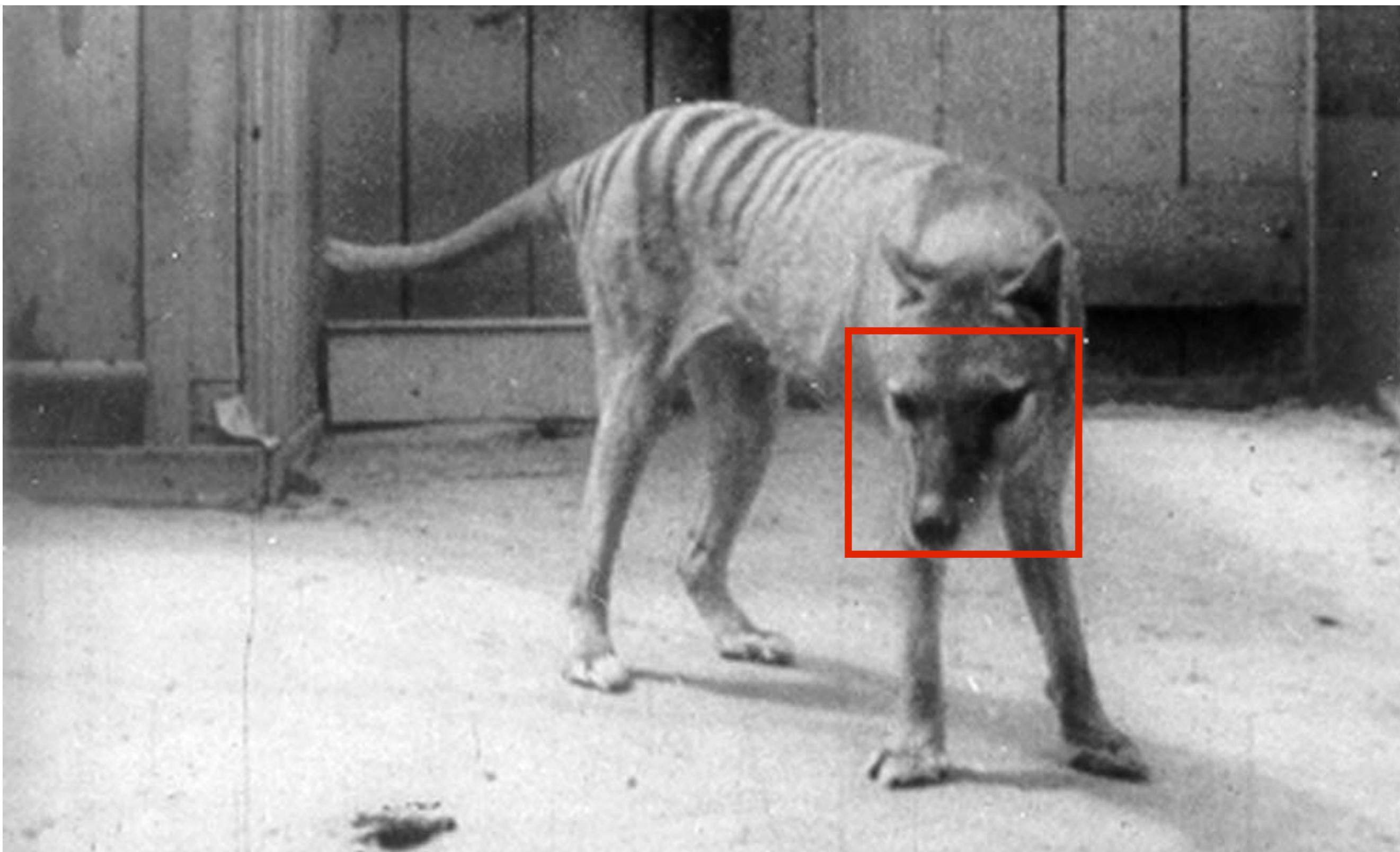
$$[D] = \{0, 1, \dots, D - 1\}$$

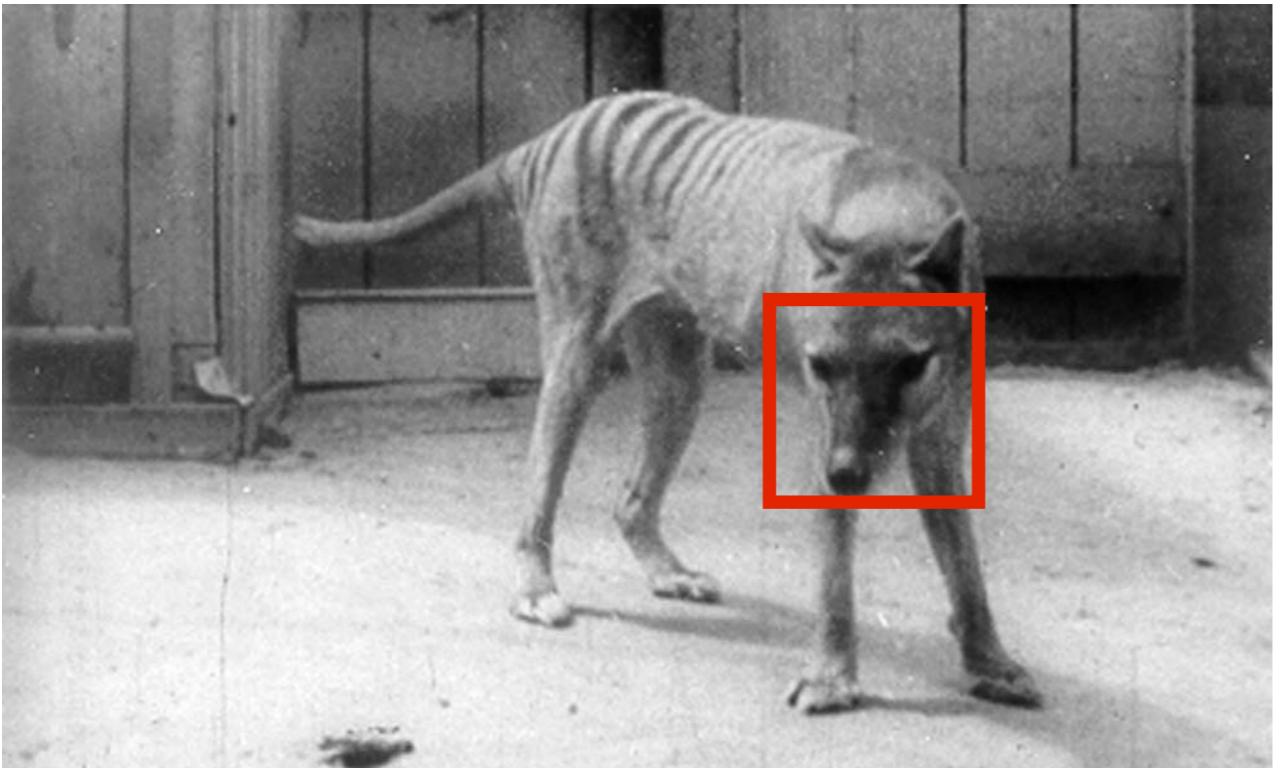
- N-dim FFT obtained independent 1-dim FFTs

Today

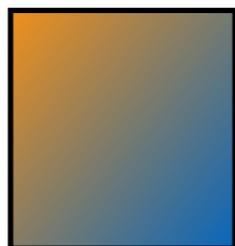
- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks





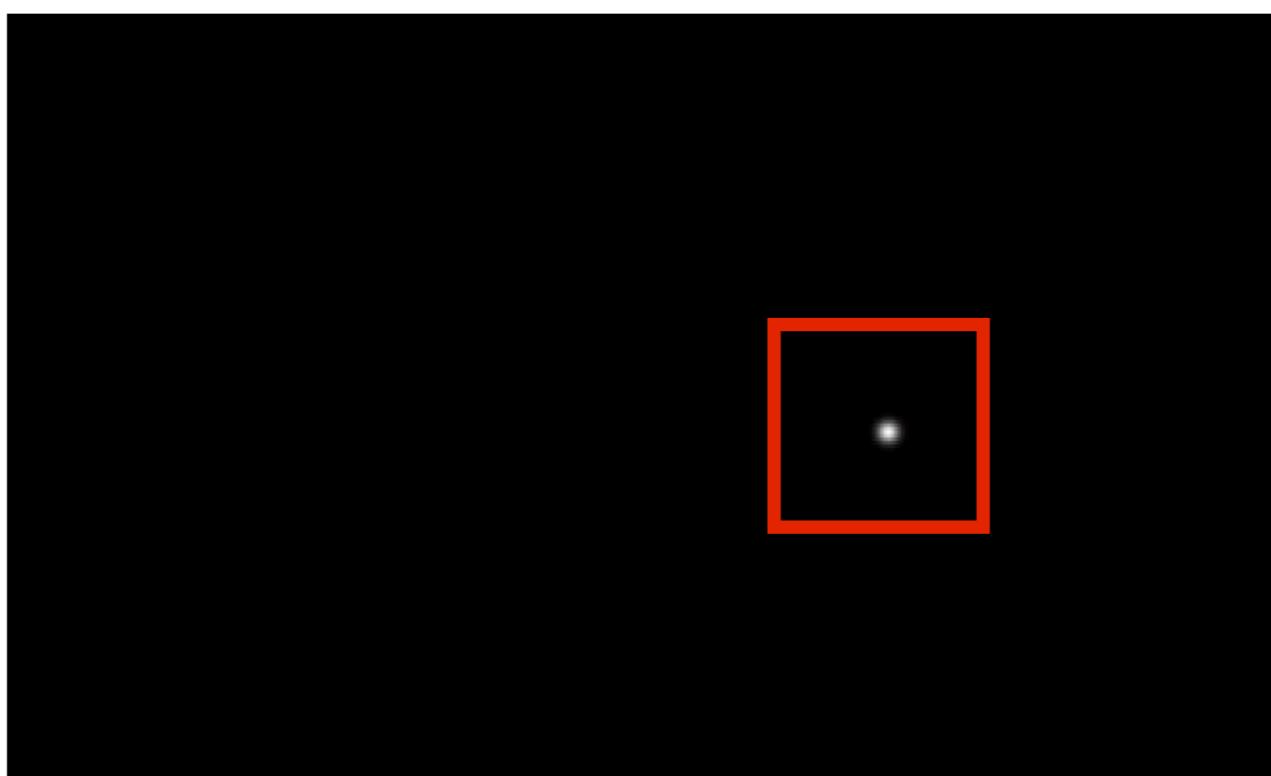


“known signal” \mathbf{X}

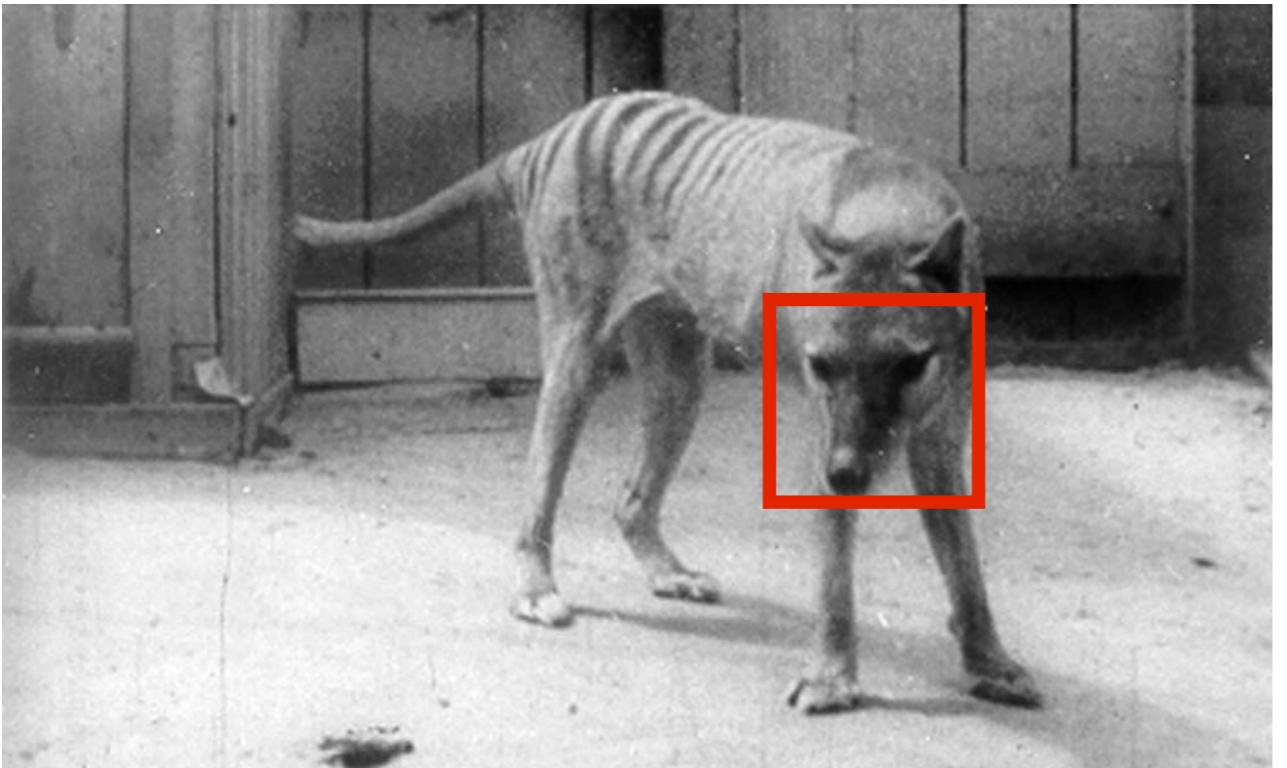


“unknown
filter”

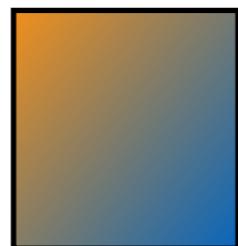
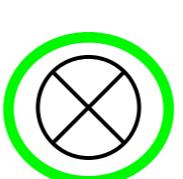
\mathbf{g}



“known response” \mathbf{y}



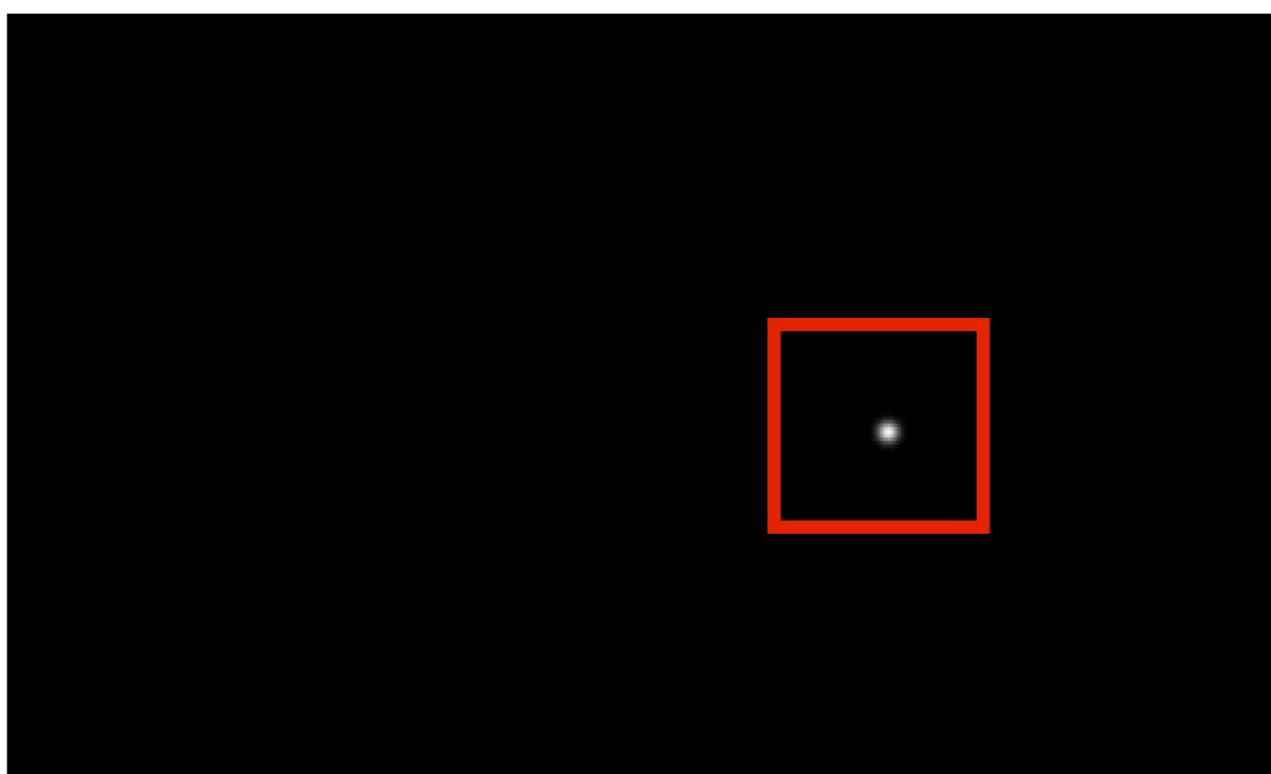
“known signal” \mathbf{X}



“unknown
filter”

\mathbf{g}

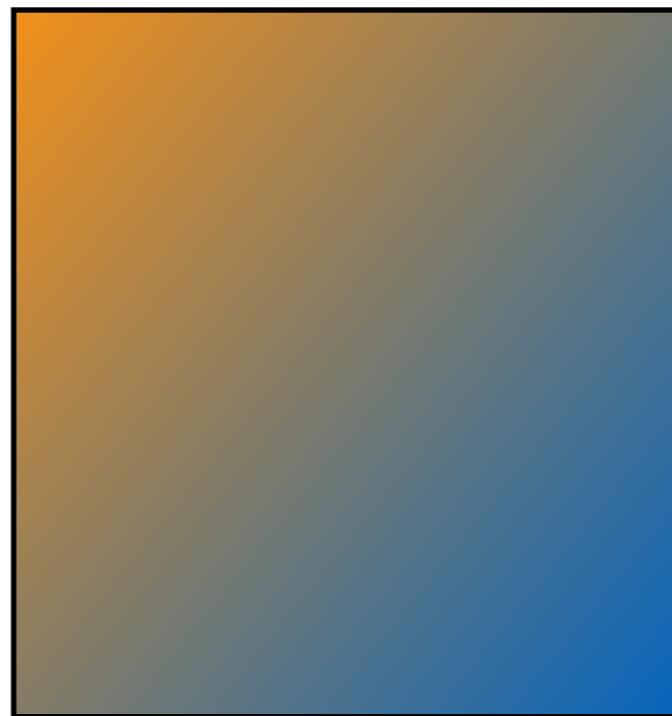
“correlation”



“known response” \mathbf{y}

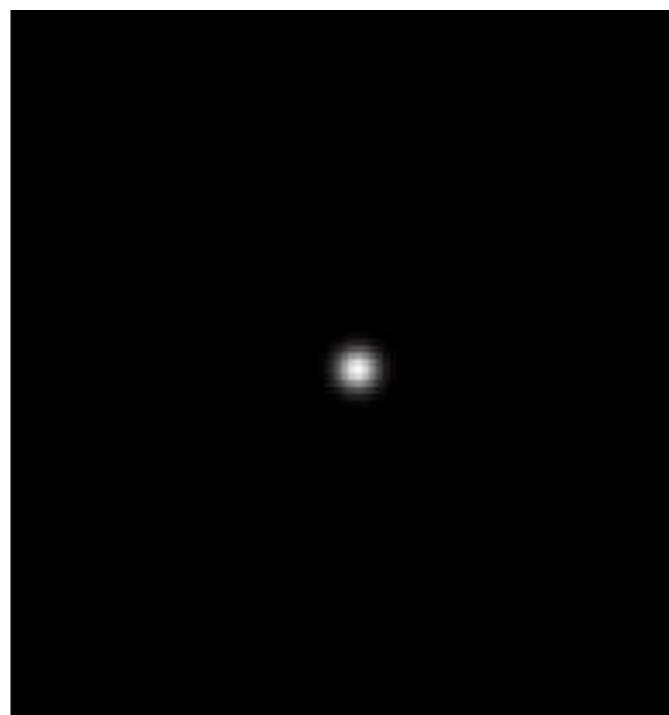


“known signal” **X**



“unknown
filter”
g

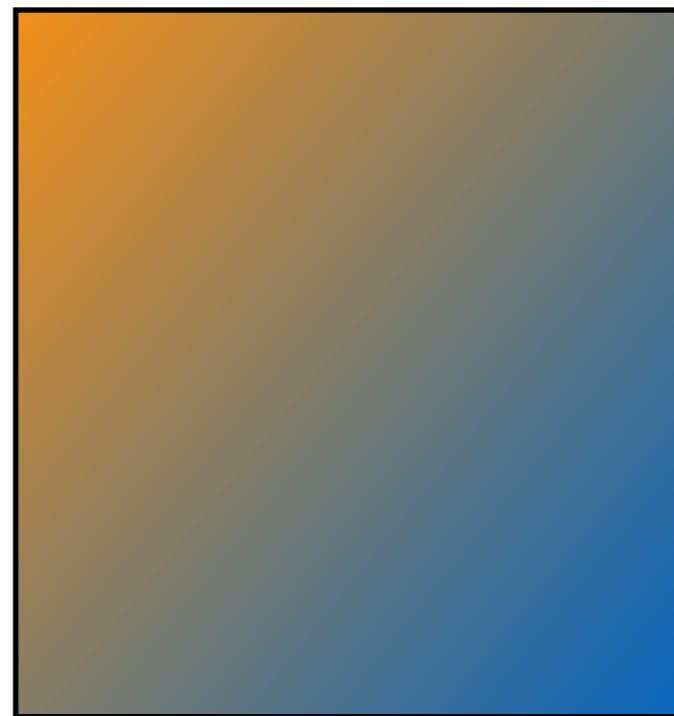
“circular correlation”



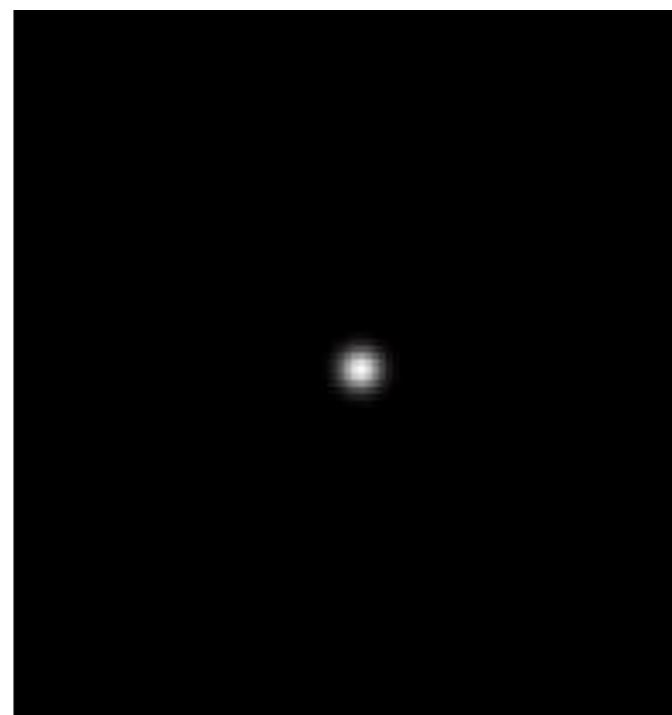
“known response” **y**



“known signal” \mathbf{X}



“unknown
filter”
 \mathbf{g}

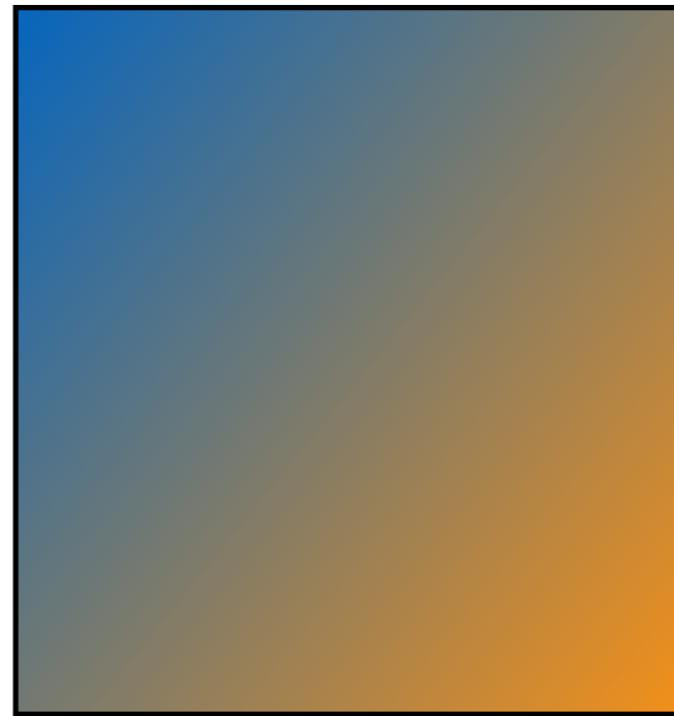


“known response” \mathbf{y}



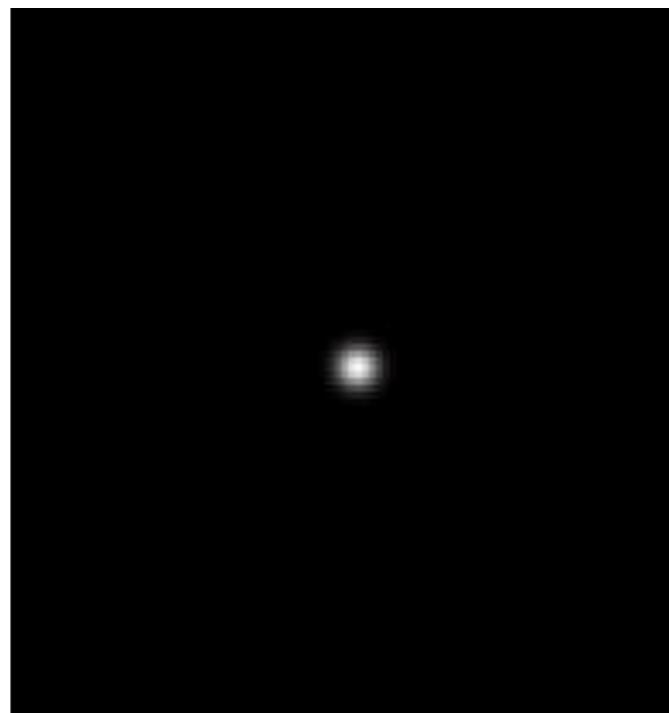
“known signal” **X**

*



“unknown
filter”
h

`g = flipud(fliplr(h))`



“known response” **y**





$$\mathbf{x} \in \mathcal{R}^D$$



$\mathbf{x}[0, 0]$



$\mathbf{x}[0, 0]$



$\mathbf{x}[20, 20]$

```
>>> xshift = circshift(x, (20, 20))
```



$\mathbf{x}[0, 0]$



$\mathbf{x}[20, 20]$



$\mathbf{x}[-20, -20]$

```
>>> xshift = circshift(x, (-20, -20))
```



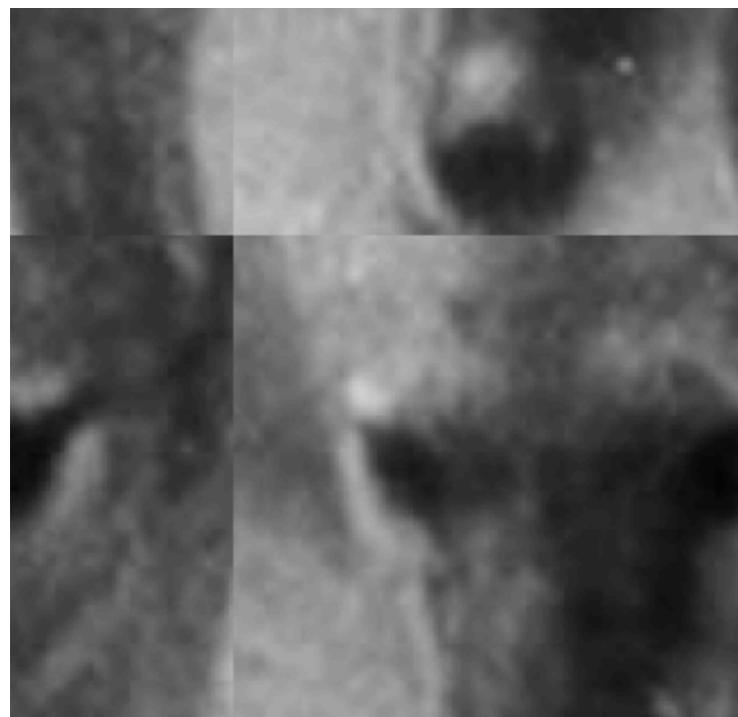
$\mathbf{x}[0, 0]$



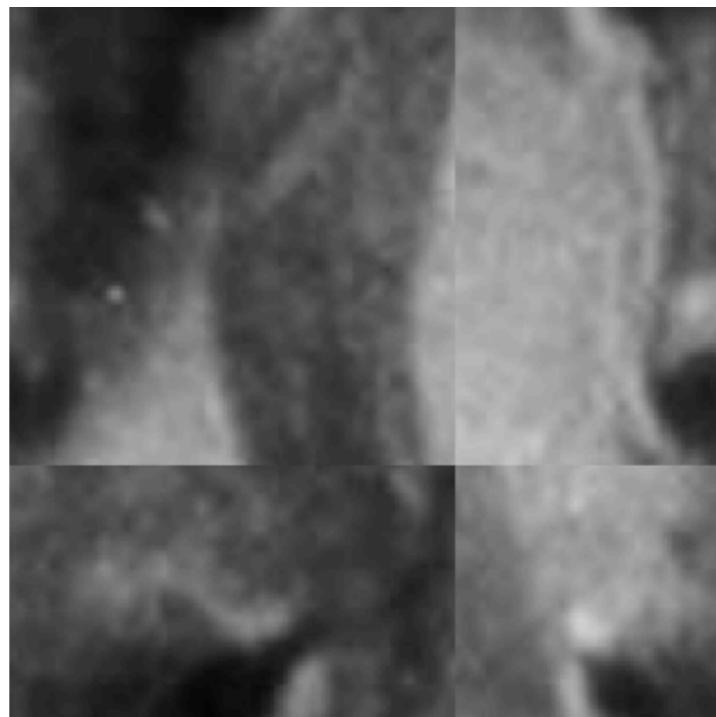
$\mathbf{x}[20, 20]$



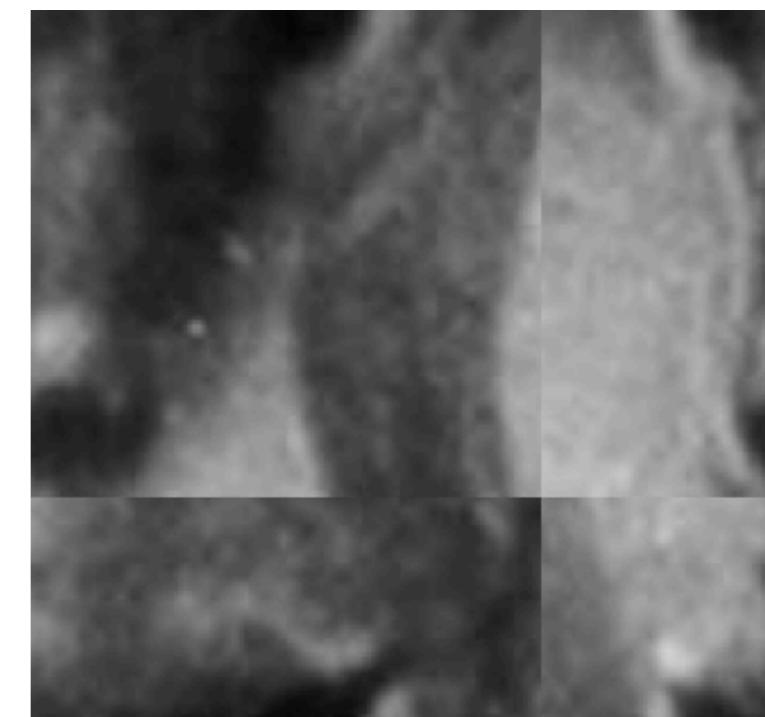
$\mathbf{x}[-20, -20]$



$\mathbf{x}[100, 100]$



$\mathbf{x}[-100, -100]$



$\mathbf{x}[200, 200]$



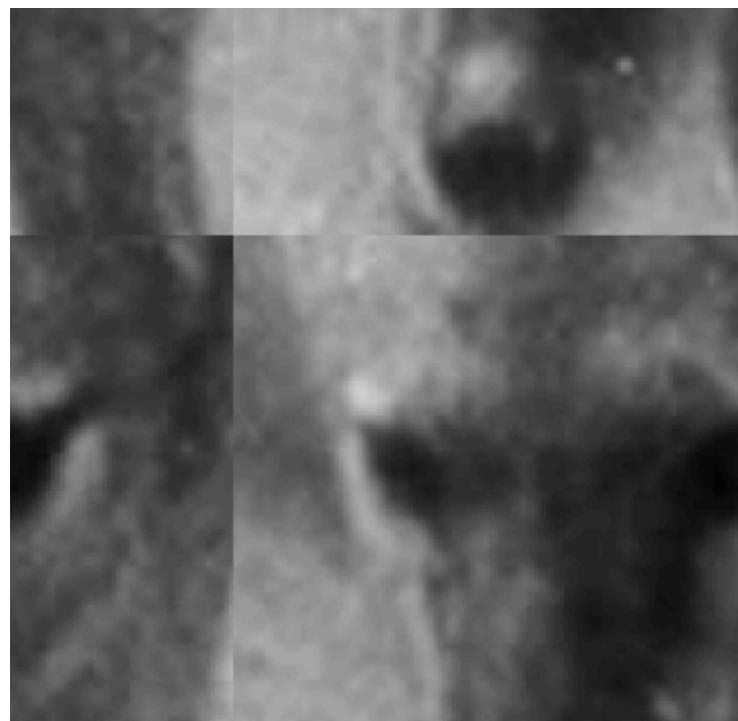
$\mathbf{x}[0, 0]$



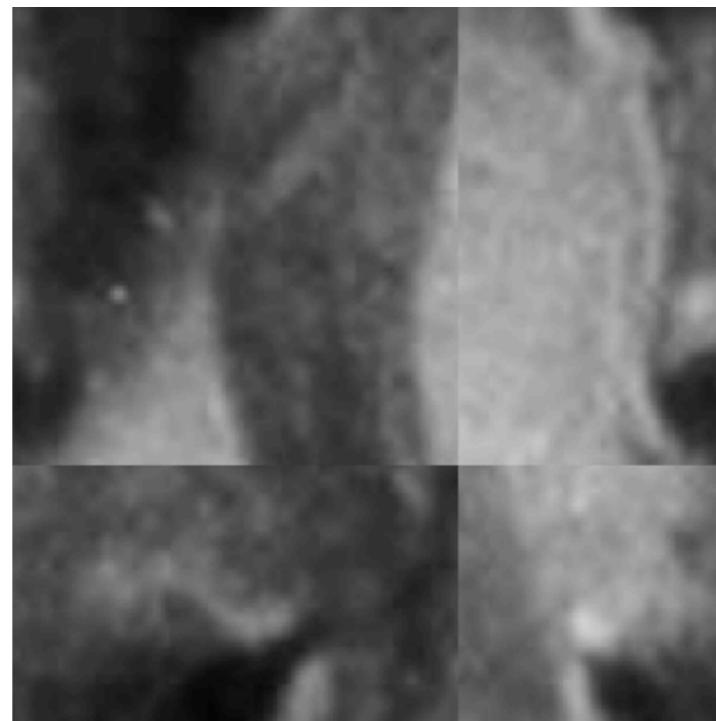
$\mathbf{x}[20, 20]$



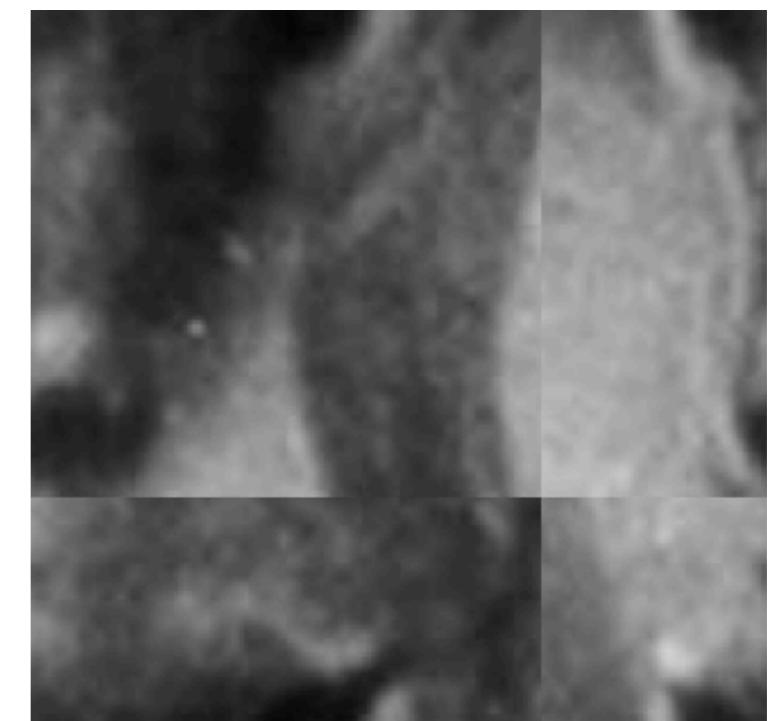
$\mathbf{x}[-20, -20]$



$\mathbf{x}[100, 100]$



$\mathbf{x}[-100, -100]$



$\mathbf{x}[200, 200]$

```
def circshift(im, shift):  
    return roll(roll(im, shift[0], axis=0), shift[1], axis=1)
```

Linear Least Squares Discriminant

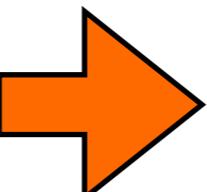
- One can view a correlation filters in the spatial domain as a linear least squares discriminant.
- Made popular by Bolme et al., referred to in literature as a **Minimum Output Sum of Squared Error (MOSSE)** filter.

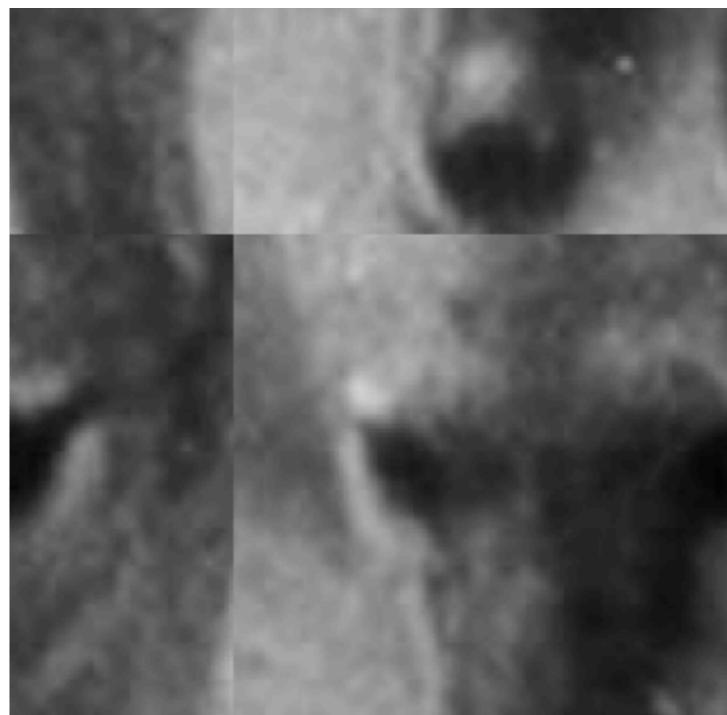
$$\arg \min_{\mathbf{h}} \frac{1}{2} \sum_{\tau \in \mathcal{C}} |y_\tau - \mathbf{x}[\tau]^T \mathbf{h}|^2$$

Linear Least Squares Discriminant

- One can view a correlation filters in the spatial domain as a linear least squares discriminant.
- Made popular by Bolme et al., referred to in literature as a **Minimum Output Sum of Squared Error (MOSSE)** filter.

$$\arg \min_{\mathbf{h}} \frac{1}{2} \sum_{\tau \in \mathcal{C}} |y_\tau - \mathbf{x}[\tau]^T \mathbf{h}|^2$$

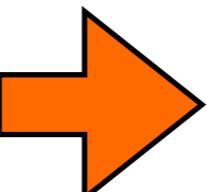
$\mathbf{x}[\tau]$ 

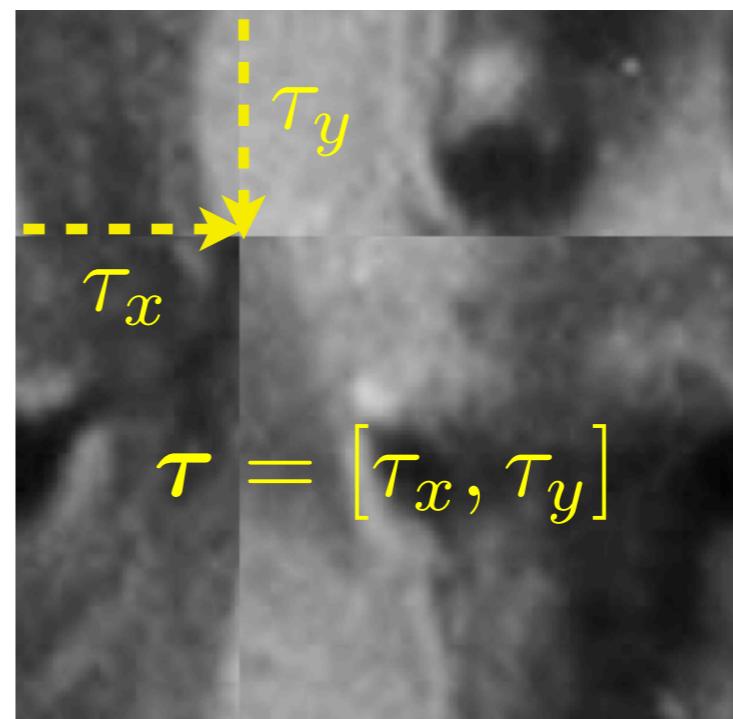


Linear Least Squares Discriminant

- One can view a correlation filters in the spatial domain as a linear least squares discriminant.
- Made popular by Bolme et al., referred to in literature as a **Minimum Output Sum of Squared Error (MOSSE)** filter.

$$\arg \min_{\mathbf{h}} \frac{1}{2} \sum_{\tau \in \mathcal{C}} |y_\tau - \mathbf{x}[\tau]^T \mathbf{h}|^2$$

$\mathbf{x}[\tau]$ 

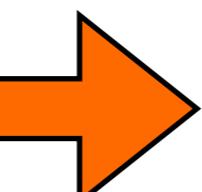


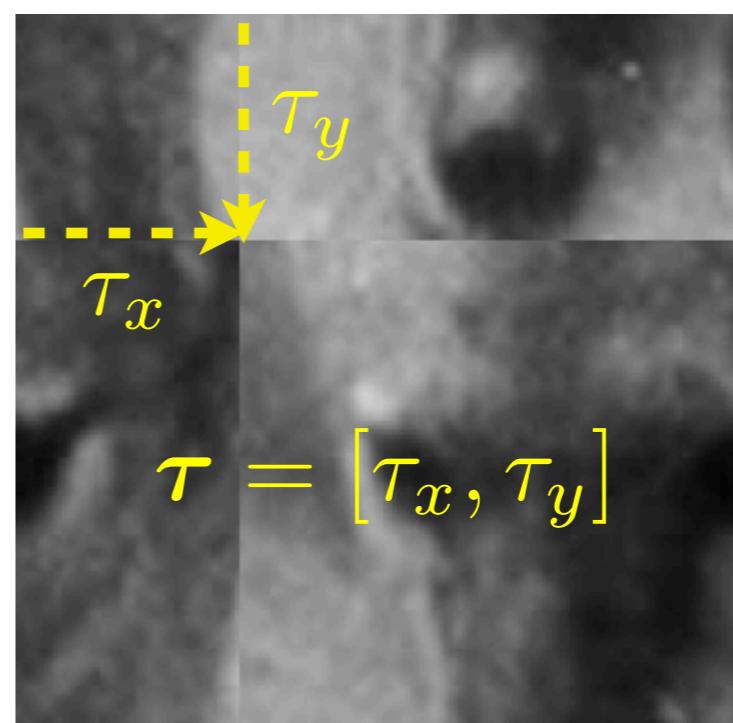
Linear Least Squares Discriminant

- One can view a correlation filters in the spatial domain as a linear least squares discriminant.
- Made popular by Bolme et al., referred to in literature as a **Minimum Output Sum of Squared Error (MOSSE)** filter.

$$\arg \min_{\mathbf{h}} \frac{1}{2} \sum_{\tau \in \mathcal{C}} |y_\tau - \mathbf{x}[\tau]^T \mathbf{h}|^2$$

“set of all shifts”

$\mathbf{x}[\tau]$ 



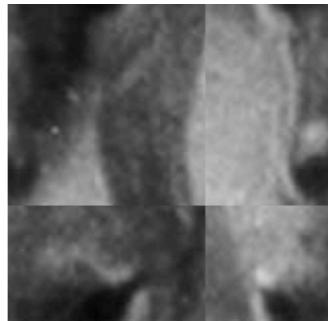
Ridge Regression

$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

“data term”

“regulariser”
or “prior”

$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

$\mathbf{x}[\tau]$			\dots	
y_τ	1	0	\dots	0

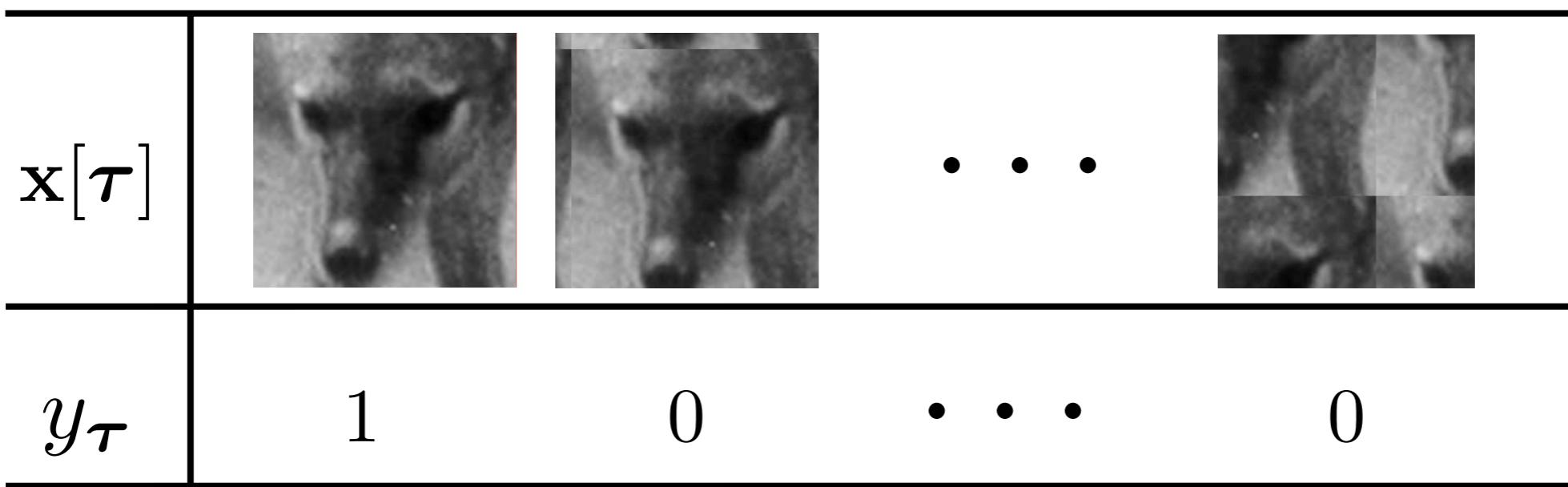
$$\mathbf{X} = [\mathbf{x}[\tau_1], \dots, \mathbf{x}[\tau_D]]$$

$$\mathbf{y} = [y_{\tau_1}, \dots, y_{\tau_D}]^T$$

$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \rightarrow \mathcal{O}(D^3)$$

D = number of samples in \mathbf{x}



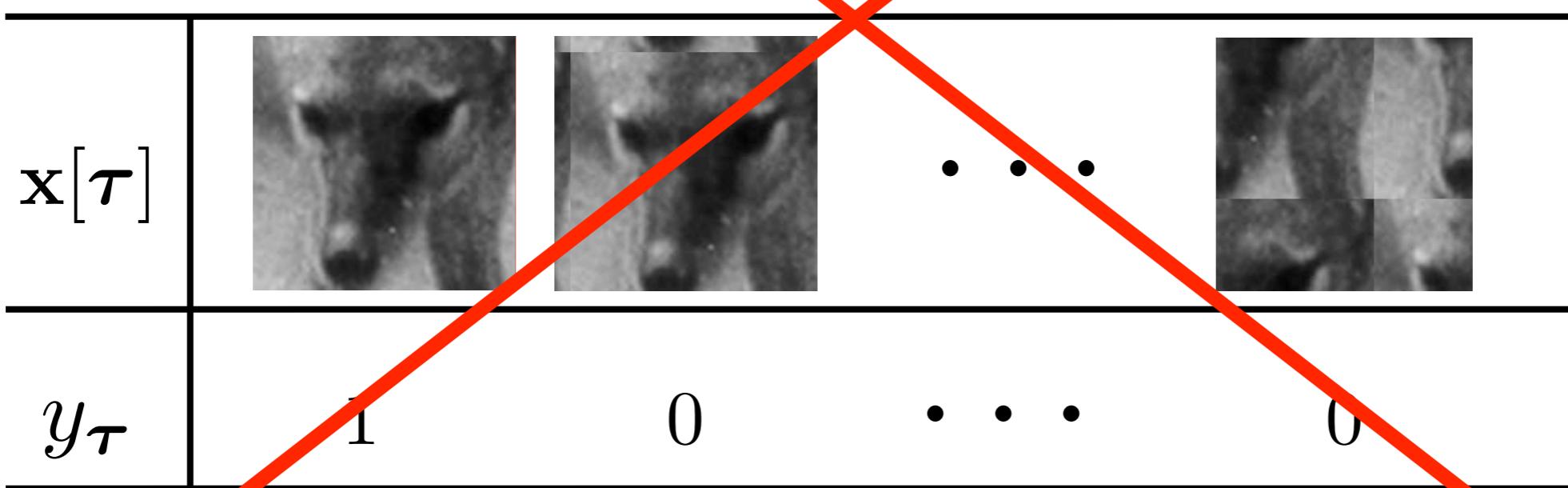
$$\mathbf{X} = [\mathbf{x}[\tau_1], \dots, \mathbf{x}[\tau_D]]$$

$$\mathbf{y} = [y_{\tau_1}, \dots, y_{\tau_D}]^T$$

$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

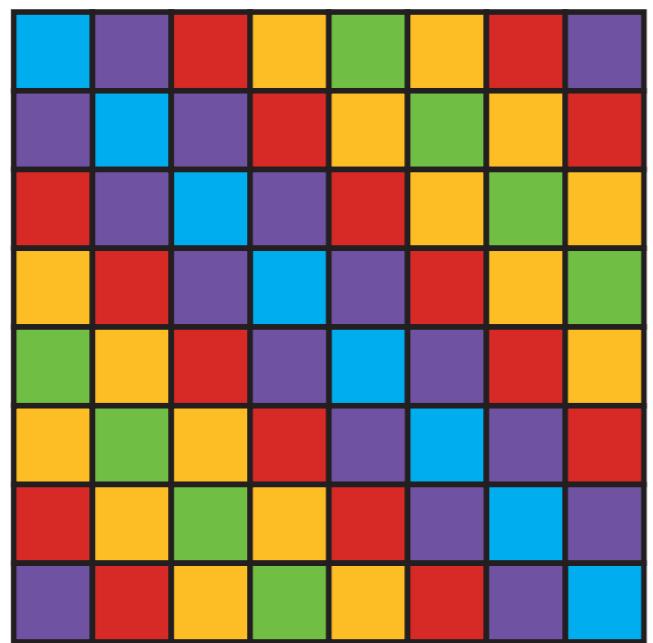
$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \rightarrow \mathcal{O}(D^3)$$

$D = \text{number of samples in } \mathbf{x}$



$$\mathbf{X} = [\mathbf{x}[\tau_1], \dots, \mathbf{x}[\tau_D]]$$

$$\mathbf{y} = [y_{\tau_1}, \dots, y_{\tau_D}]^T$$



X

F⁻¹

$$\begin{array}{c}
 \mathbf{F} \\
 \mathbf{X}
 \end{array}
 \quad \mathbf{F}^{-1} = \quad
 \begin{array}{c}
 \text{---} \\
 D \cdot \text{diag}(\hat{\mathbf{x}})
 \end{array}$$

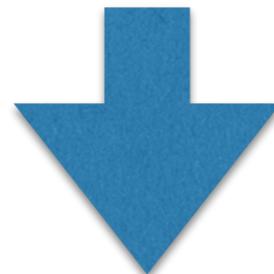
A diagram illustrating the inverse Fourier transform. On the left, there are two matrices: \mathbf{F} (top) and \mathbf{X} (bottom). Matrix \mathbf{F} is a 10x10 grid with various colored squares (blue, purple, red, yellow, green) representing non-zero elements. Matrix \mathbf{X} is a 10x10 grid where every square is either blue or red. In the center, the equation $\mathbf{F}^{-1} =$ is followed by a horizontal line. To the right of the line, the expression $D \cdot \text{diag}(\hat{\mathbf{x}})$ is shown. Below this, another 10x10 matrix is displayed, consisting entirely of blue squares except for a diagonal line of red squares, representing the scaled and diagonalized version of \mathbf{X} .

$\mathbf{F} \leftarrow$ Fourier Transform

■ Not Always Zero ■ Always Zero

$$(\mathbf{X}^T\mathbf{X}+\lambda \mathbf{I})\mathbf{h}=\mathbf{X}^T\mathbf{y}$$

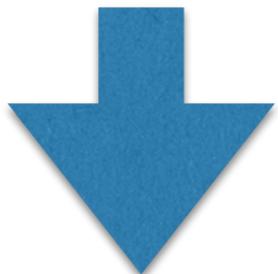
$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{h} = \mathbf{X}^T \mathbf{y}$$



“express as convolution”

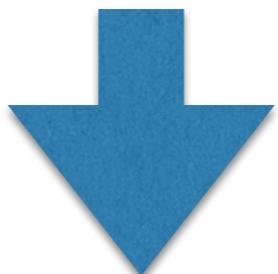
$$(\mathbf{x} \otimes \mathbf{x} + \lambda \boldsymbol{\delta}) * \mathbf{h} = \mathbf{x} \otimes \mathbf{y}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{h} = \mathbf{X}^T \mathbf{y}$$



“express as convolution”

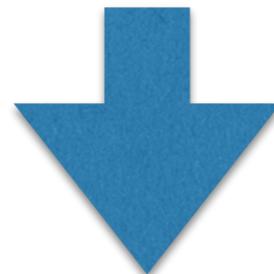
$$(\mathbf{x} \otimes \mathbf{x} + \lambda \delta) * \mathbf{h} = \mathbf{x} \otimes \mathbf{y}$$



“diagonalise with DFT”

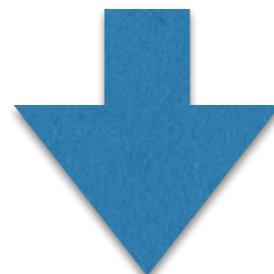
$$(\hat{\mathbf{x}}^* \circ \hat{\mathbf{x}} + \lambda \mathbf{1}) \circ \hat{\mathbf{h}} = \hat{\mathbf{x}}^* \circ \hat{\mathbf{y}}$$

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{h} = \mathbf{X}^T \mathbf{y}$$



“express as convolution”

$$(\mathbf{x} \otimes \mathbf{x} + \lambda \boldsymbol{\delta}) * \mathbf{h} = \mathbf{x} \otimes \mathbf{y}$$



“diagonalise with DFT”

$$(\hat{\mathbf{x}}^* \circ \hat{\mathbf{x}} + \lambda \mathbf{1}) \circ \hat{\mathbf{h}} = \hat{\mathbf{x}}^* \circ \hat{\mathbf{y}}$$

$$\hat{\mathbf{h}} = \hat{\mathbf{S}}_{xy}^{-1} (\hat{\mathbf{S}}_{xx} + \lambda \mathbf{1})$$

$$\hat{\mathbf{h}} \; = \; \hat{\mathbf{S}}_{xy}^{-1}\left(\hat{\mathbf{S}}_{xx} + \lambda\mathbf{1}\right)$$

$$\hat{\mathbf{h}} = \hat{\mathbf{S}}_{xy}^{-1} (\hat{\mathbf{S}}_{xx} + \lambda \mathbf{1})$$

```
>>> xf = fft2(x)
```

$$\hat{\mathbf{h}} = \hat{\mathbf{S}}_{xy}^{-1} (\hat{\mathbf{S}}_{xx} + \lambda \mathbf{1})$$

```
>>> xf = fft2(x)
>>> yf = fft2(y)
```

$$\hat{\mathbf{h}} = \hat{\mathbf{S}}_{xy}^{-1} (\hat{\mathbf{S}}_{xx} + \lambda \mathbf{1})$$

```
>>> xf = fft2(x)
>>> yf = fft2(y)
>>> sxx = xf*conj(xf)
```

$$\hat{\mathbf{h}} = \hat{\mathbf{S}}_{xy}^{-1} (\hat{\mathbf{S}}_{xx} + \lambda \mathbf{1})$$

```
>>> xf = fft2(x)
>>> yf = fft2(y)
>>> sxx = xf*conj(xf)
>>> sxy = xf*conj(yf)
```

$$\hat{\mathbf{h}} = \hat{\mathbf{S}}_{xy}^{-1} (\hat{\mathbf{S}}_{xx} + \lambda \mathbf{1})$$

```
>>> xf = fft2(x)
>>> yf = fft2(y)
>>> sxx = xf*conj(xf)
>>> sxy = xf*conj(yf)
>>> hf = sxy/(sxx + 1e-3)
```

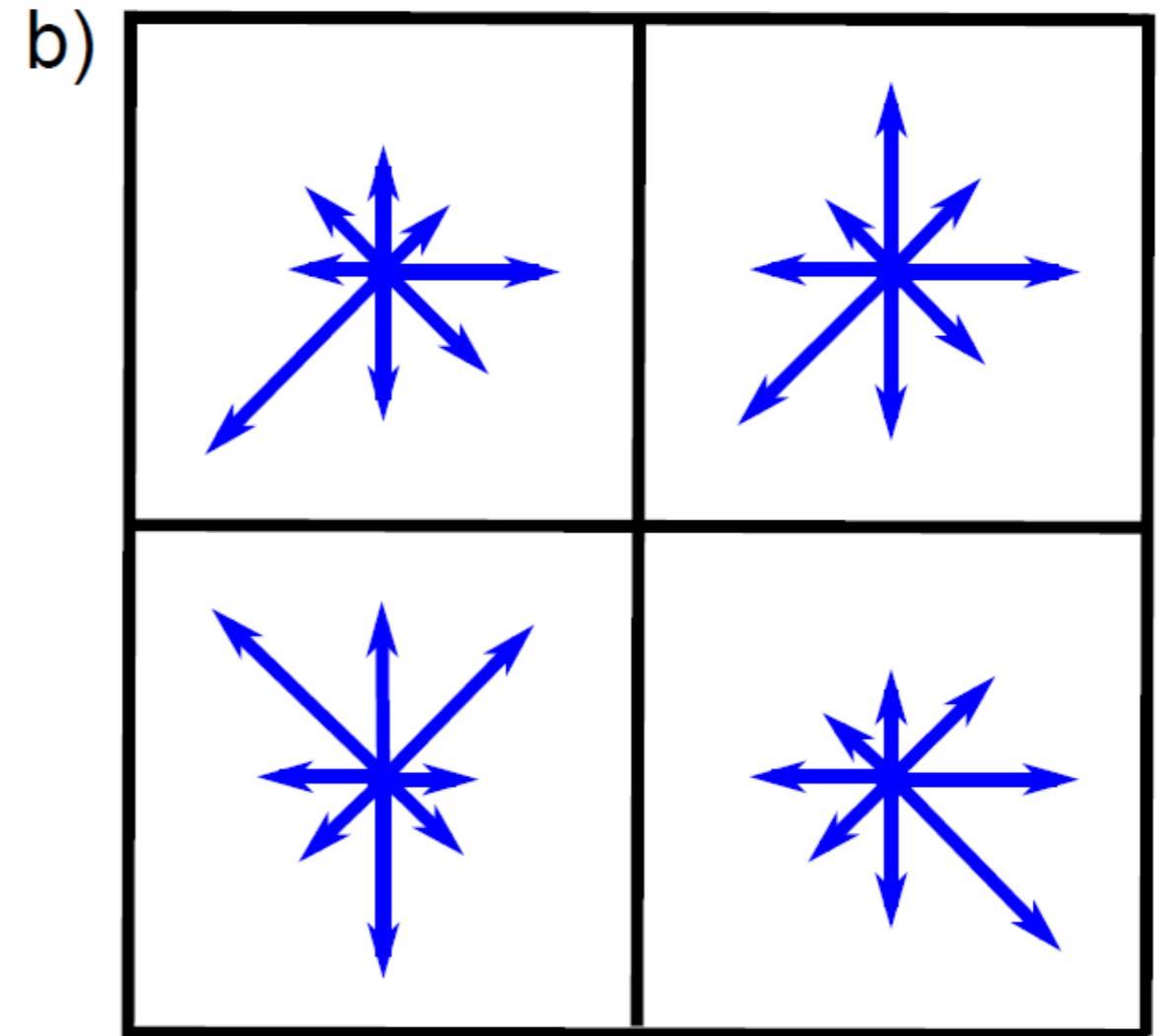
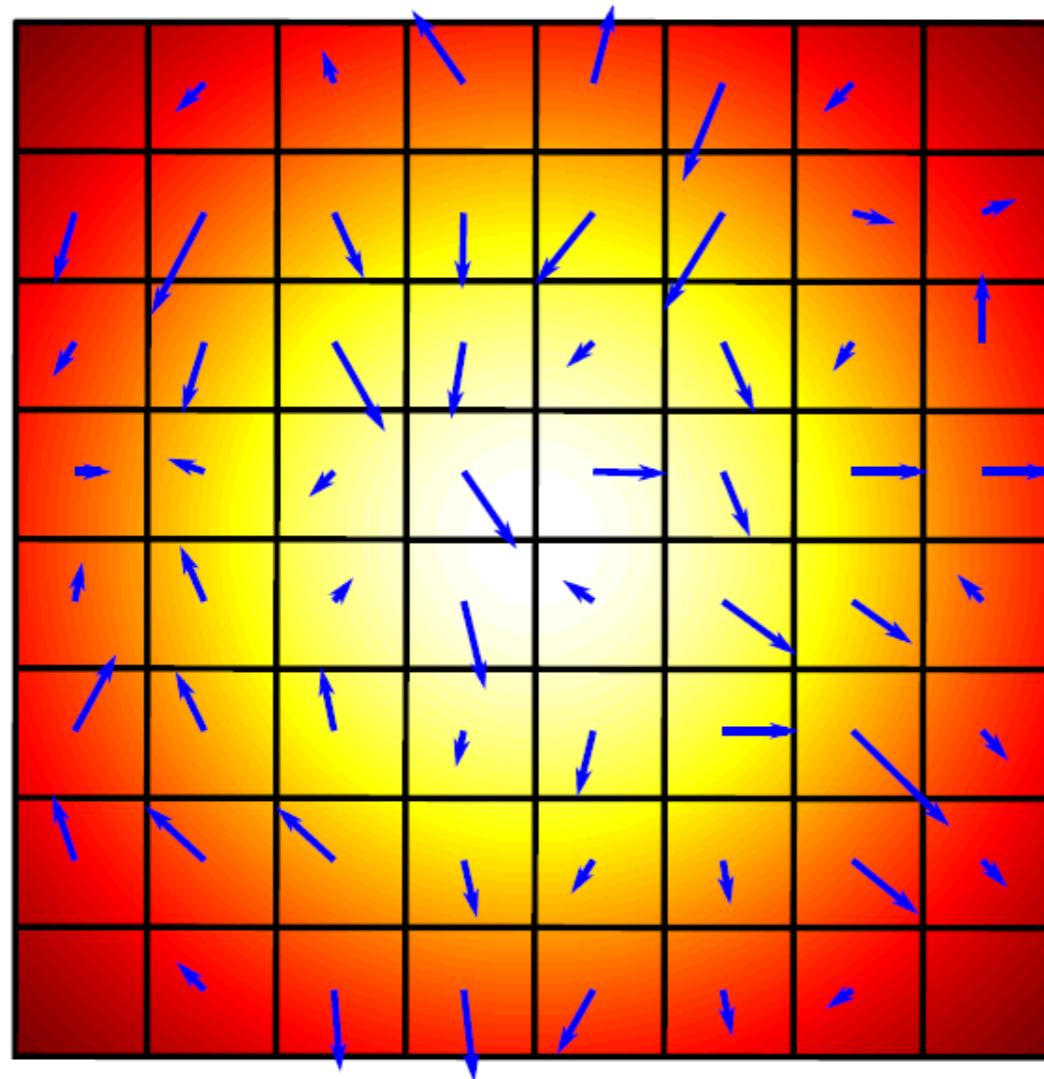


Algorithm	Frame Rate	CPU
FragTrack[1]	realtime	Unknown
GBDL[19]	realtime	3.4 Ghz Pent. 4
IVT [17]	7.5fps	2.8Ghz CPU
MILTrack[2]	25 fps	Core 2 Quad
MOSSE Filters	669fps	2.4Ghz Core 2 Duo



Algorithm	Frame Rate	CPU
FragTrack[1]	realtime	Unknown
GBDL[19]	realtime	3.4 Ghz Pent. 4
IVT [17]	7.5fps	2.8Ghz CPU
MILTrack[2]	25 fps	Core 2 Quad
MOSSE Filters	669fps	2.4Ghz Core 2 Duo

Robust Representations



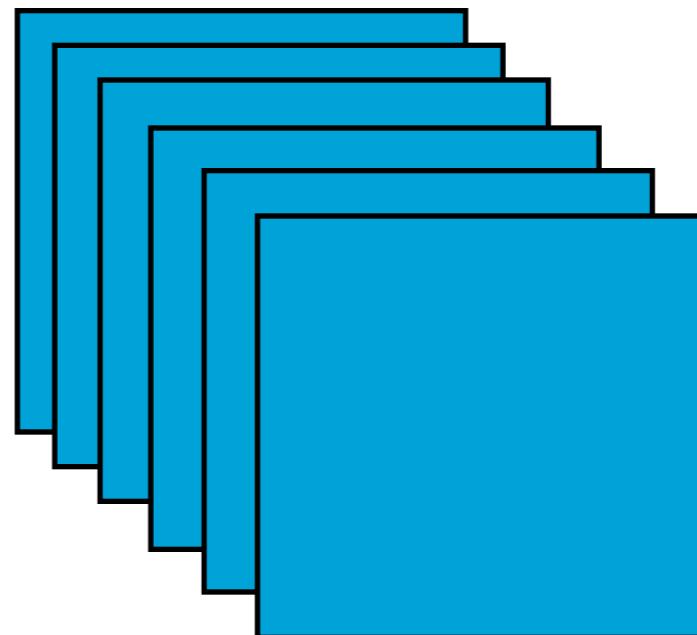
1. Compute image gradients
2. Pool into local histograms
3. Concatenate histograms
4. Normalize histograms

$$\psi \rightarrow \mathbb{R}^N : \mathbb{R}^{N \times K}$$

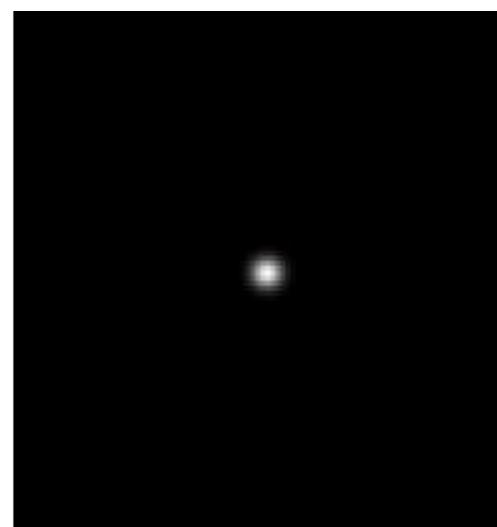


“known signal” \mathbf{X}

*



“unknown
filter”
 \mathbf{h}



“known response” \mathbf{y}

$$E(\mathbf{h}) = \|\mathbf{y} - \sum_{k=1}^K \mathbf{x}^{(k)} * \mathbf{h}^{(k)}\|_2^2 + \frac{\lambda}{2} \sum_{k=1}^K \|\mathbf{h}^{(k)}\|_2^2$$

K = no. of channels

$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

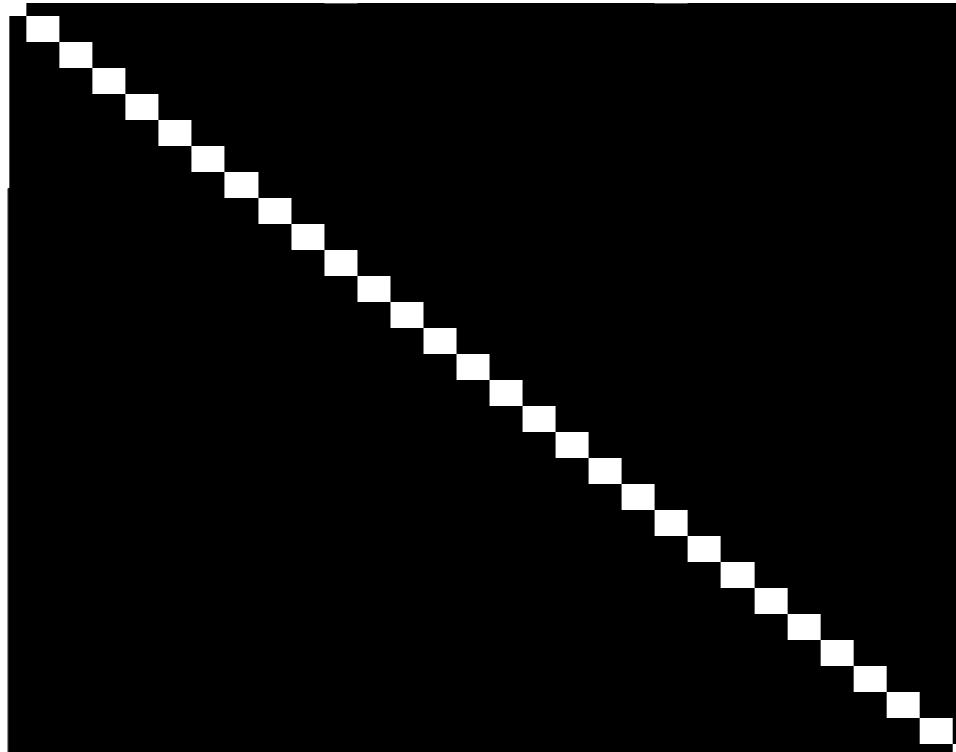
$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

$$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1} \quad \xrightarrow{\text{orange arrow}} \quad \mathcal{O}(K^3 D^3)$$

K = no. of channels
 D = number of samples in \mathbf{x}

$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

$$\hat{\mathbf{X}}\hat{\mathbf{X}}^T + \lambda \mathbf{I} =$$



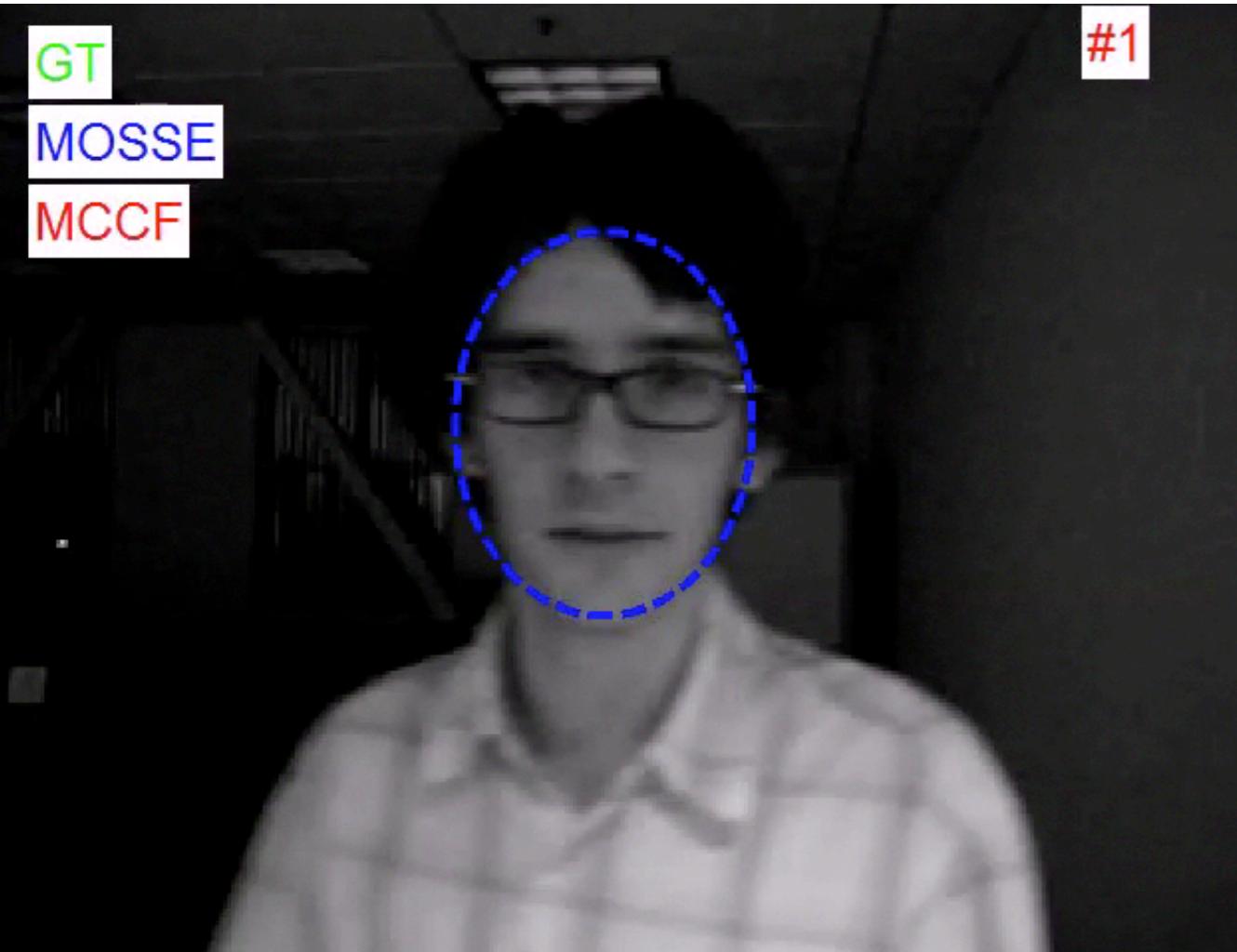
$$E(\mathbf{h}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|_2^2 + \frac{\lambda}{2} \|\mathbf{h}\|_2^2$$

$$(\hat{\mathbf{X}}\hat{\mathbf{X}}^T + \lambda \mathbf{I})^{-1} \xrightarrow{\text{orange arrow}} \mathcal{O}(K^3 D)$$

K = no. of channels
 D = number of samples in \mathbf{x}

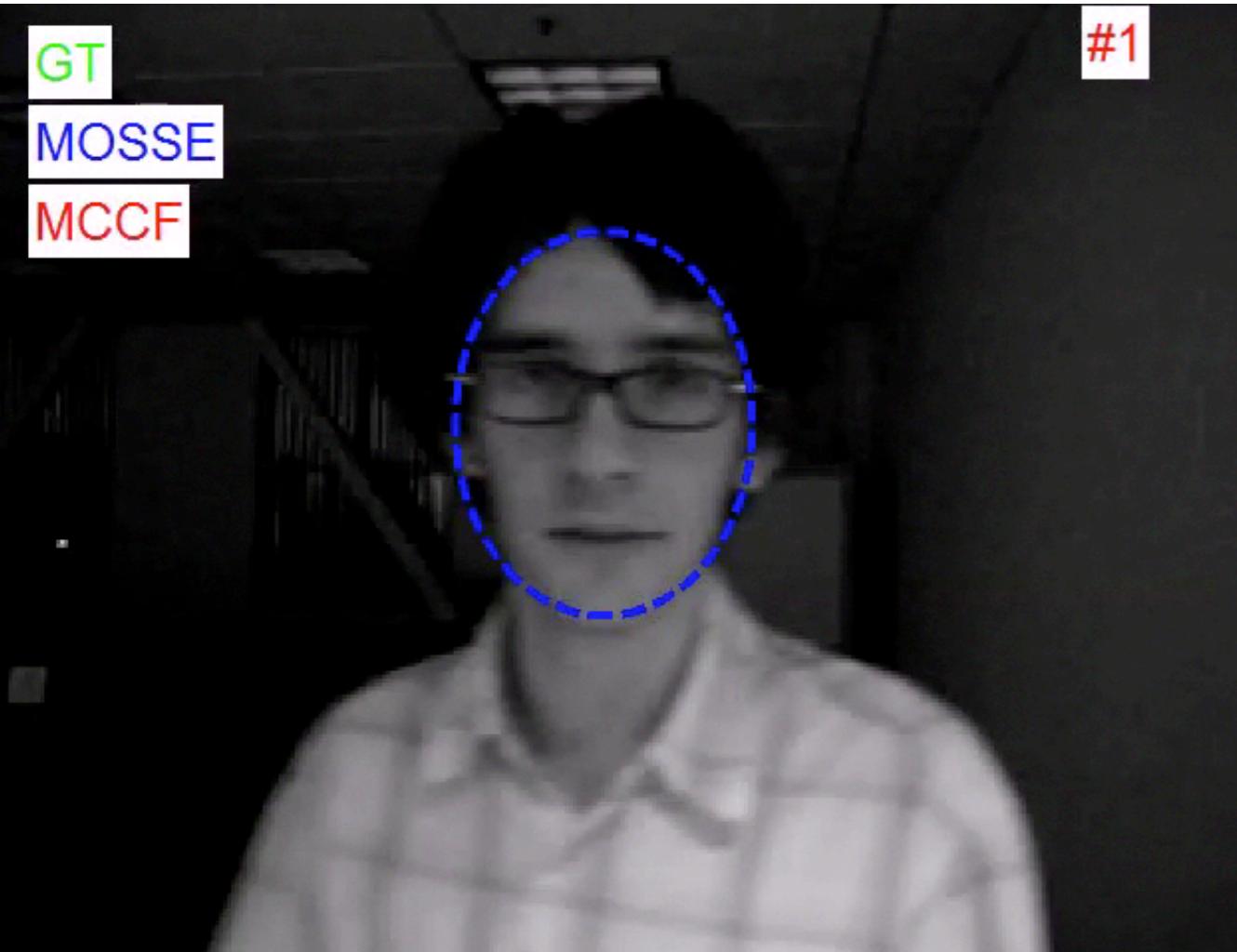
where,

$$\hat{\mathbf{X}} = [\text{diag}(\hat{\mathbf{x}}_1), \dots, \text{diag}(\hat{\mathbf{x}}_K)]$$



MOSSE = Single-Channel CF

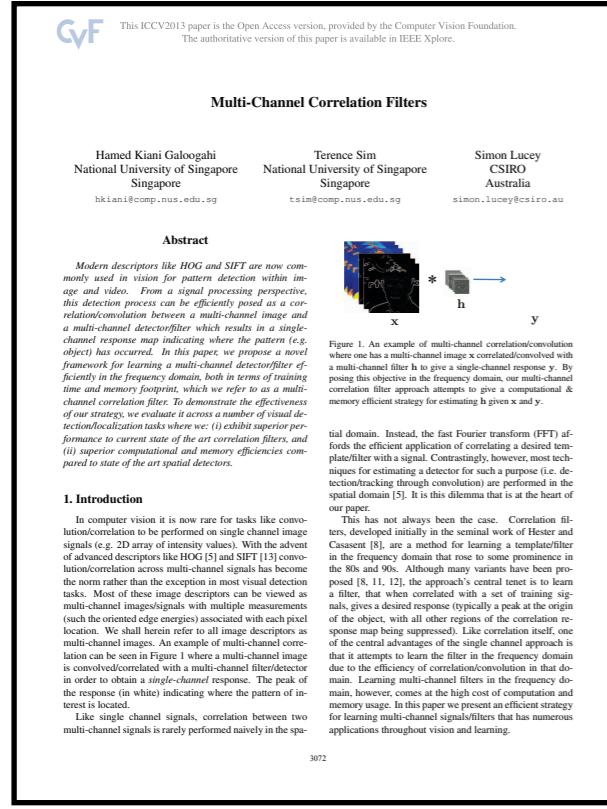
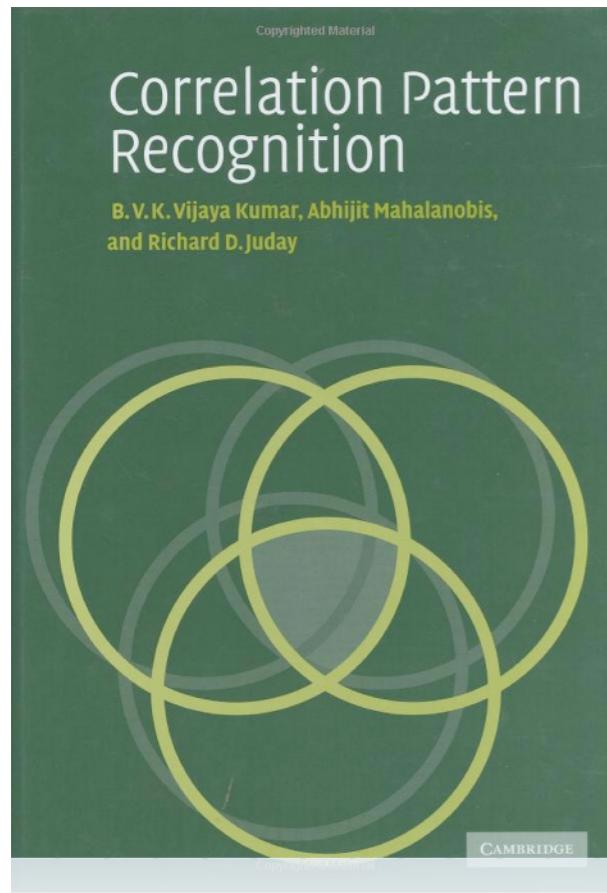
MCCF = Multi-Channel CF



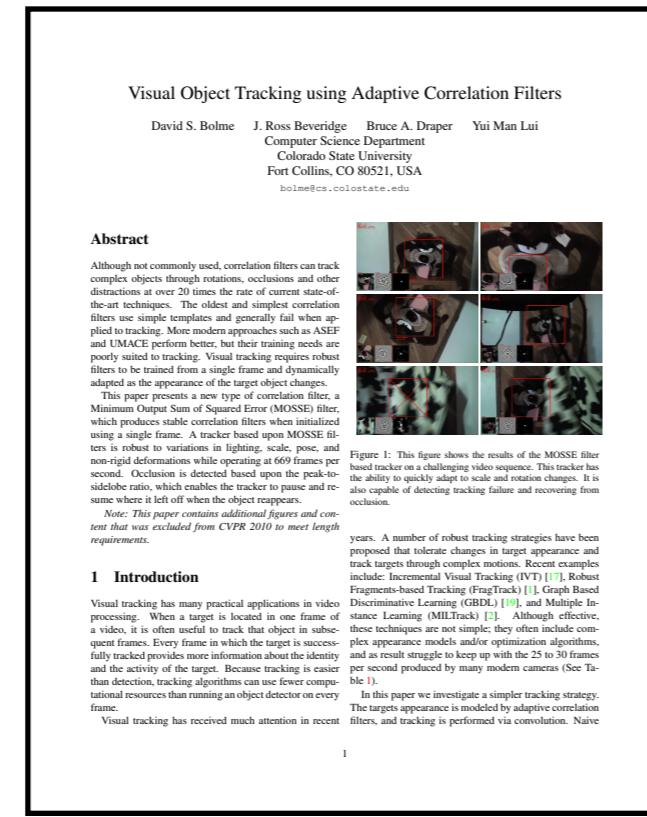
MOSSE = Single-Channel CF

MCCF = Multi-Channel CF

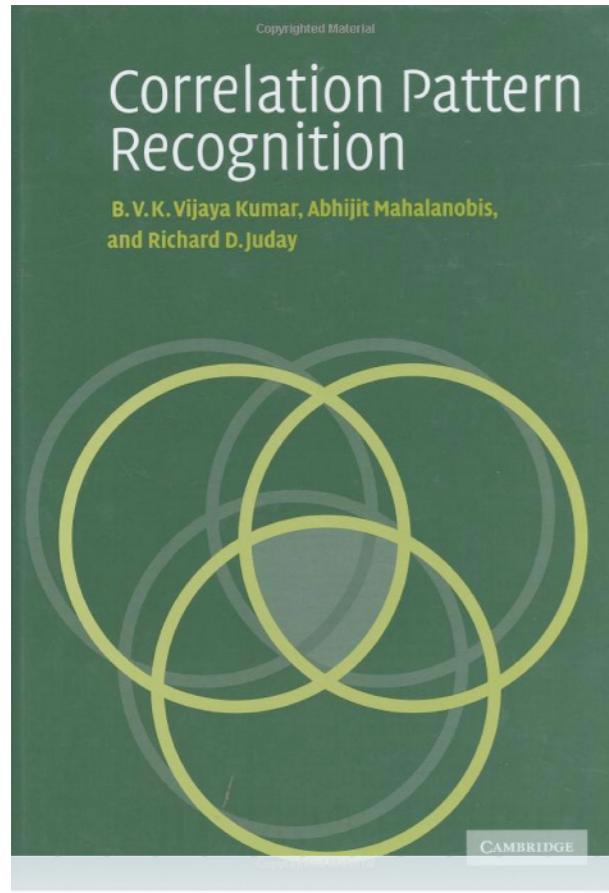
More to read...



- Vijaya Kumar, Mahalanobis, & Juday "Correlation Pattern Recognition", 2010.
- Bolme, Beveridge, Draper & Lui, "Visual Object Tracking using Adaptive Correlation Filters", CVPR 2010.
- Galoogahi, Sim & Lucey "Multi-Channel Correlation Filters", ICCV 2013.
- Henriques, Caseiro, Martins, & Batista, "High-Speed Tracking with Kernelized Correlation Filters", PAMI 2014.



More to read...



GvF This ICCV2013 paper is the Open Access version, provided by the Computer Vision Foundation. The authoritative version of this paper is available in IEEE Xplore.

Multi-Channel Correlation Filters

Hamed Kiani Galoogahi
National University of Singapore
Singapore
hkiani@comp.nus.edu.sg

Terence Sim
National University of Singapore
Singapore
tsim@comp.nus.edu.sg

Simon Lucey
CSIRO
Australia
simon.lucey@csiro.au

Abstract

Modern descriptors like HOG and SIFT are now commonly used in vision for pattern detection within image and video. From a signal processing perspective, this is done via cross-correlation or convolution/correlation between a multi-channel image and a multi-channel detector/filter which results in a single-channel response map indicating where the pattern (e.g. object) has occurred. In this paper, we propose a novel framework for learning a multi-channel detector/filter efficiently in the frequency domain, both in terms of training time and memory footprint. Such a filter refers to as a multi-channel footprint. To demonstrate the effectiveness of our strategy, we evaluate it across a number of visual detection/localization tasks where we: (i) exhibit superior performance to current state of the art correlation filters, and (ii) superior computational and memory efficiencies compared to state of the art spatial detectors.

1. Introduction

In computer vision it is now rare for tasks like convolution/correlation to be performed on single channel image signals (e.g. 2D array of intensity values). With the advent of advanced descriptors like HOG [5] and SIFT [1], convolution/correlation across multi-channel signals has become the norm rather than the exception in most vision detection tasks. Most of these image descriptors can be viewed as multi-channel images/signals with multiple measurements (such as oriented edge energies) associated with each pixel location. We shall herein refer to all image descriptors as multi-channel images. An example of multi-channel correlation in the spatial domain is that the response of a filter is convolved/convolved with a multi-channel filter/detector/filter in order to obtain a single-channel response. The peak of the response (in white) indicating where the pattern of interest is located.

Like single channel signals, correlation between two multi-channel signals is rarely performed naively in the spa-

Copyrighted Material

- Vijaya Kumar, Mahalanobis, & Juday "Correlation Pattern Recognition", 2010.
- Bolme, Beveridge, Draper & Lui, "Visual Object Tracking using Adaptive Correlation Filters", CVPR 2010.
- Galoogahi, Sim & Lucey "Multi-Channel Correlation Filters", ICCV 2013.
- Henriques, Caseiro, Martins, & Batista, "High-Speed Tracking with Kernelized Correlation Filters", PAMI 2014.

Visual Object Tracking using Adaptive Correlation Filters

David S. Bolme J. Ross Beveridge Bruce A. Draper Yui Man Lui
Computer Science Department
Colorado State University
Fort Collins, CO 80521, USA
bolme@cs.colostate.edu

Abstract

Although not commonly used, correlation filters can track complex objects through rotations, occlusions and other distractions at over 20 times the rate of current state-of-the-art techniques. The oldest and simplest correlation filters use simple templates and generally fail when applied to tracking. More modern approaches such as ASEF and UMACE perform better, but their training needs are poorly suited to tracking. Visual tracking requires robust filters to be trained from a single frame and dynamically adapted as the appearance of the target object changes.

This paper proposes a new type of correlation filter, a Minimum Output Sum of Squared Errors (MOSE) filter, which produces stable correlation filters when initialized using a single frame. A tracker based upon MOSE filters is robust to variations in lighting, scale, pose, and non-rigid deformations while operating at 669 frames per second. Occlusion is detected based upon the peak-to-sidelobe ratio, which enables the tracker to pause and resume when it left off when the object reappears.

Note: This paper contains additional figures and content that was excluded from CVPR 2010 to meet length requirements.

1 Introduction

Visual tracking has many practical applications in video processing. When a target is located in one frame of a video, it is often useful to track that object in subsequent frames. Every frame in which the target is successfully tracked provides more information about the identity and the activity of the target. Because tracking is easier than detection, tracking algorithms can use fewer computational resources than running an object detector on every frame.

Visual tracking has received much attention in recent

years. A number of robust tracking strategies have been proposed that tolerate changes in target appearance and track targets through complex motions. Recent examples include Incremental Visual Tracking (IVT) [1], Robust Feature-based Tracking (Fractrak) [1], Gradient-Based Discriminative Learning (GBDL) [19], and Multiple Instance Learning (MILITrack) [2]. Although effective, these techniques are not simple; they often include complex appearance models and/or optimization algorithms, and as result struggle to keep up with the 25 to 30 frames per second produced by many modern cameras (See Table 1).

In this paper we investigate a simpler tracking strategy. The targets appearance is modeled by adaptive correlation filters, and tracking is performed via convolution. Naive

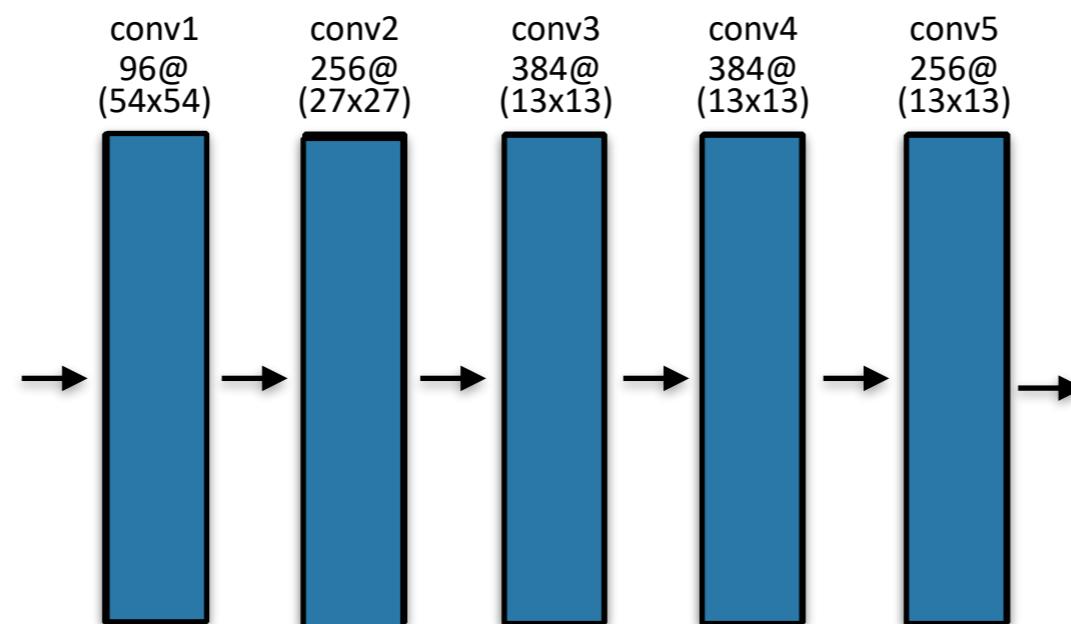
Question: Potential drawbacks of training with circular shifts?

Today

- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks

Deep Tracking Methods

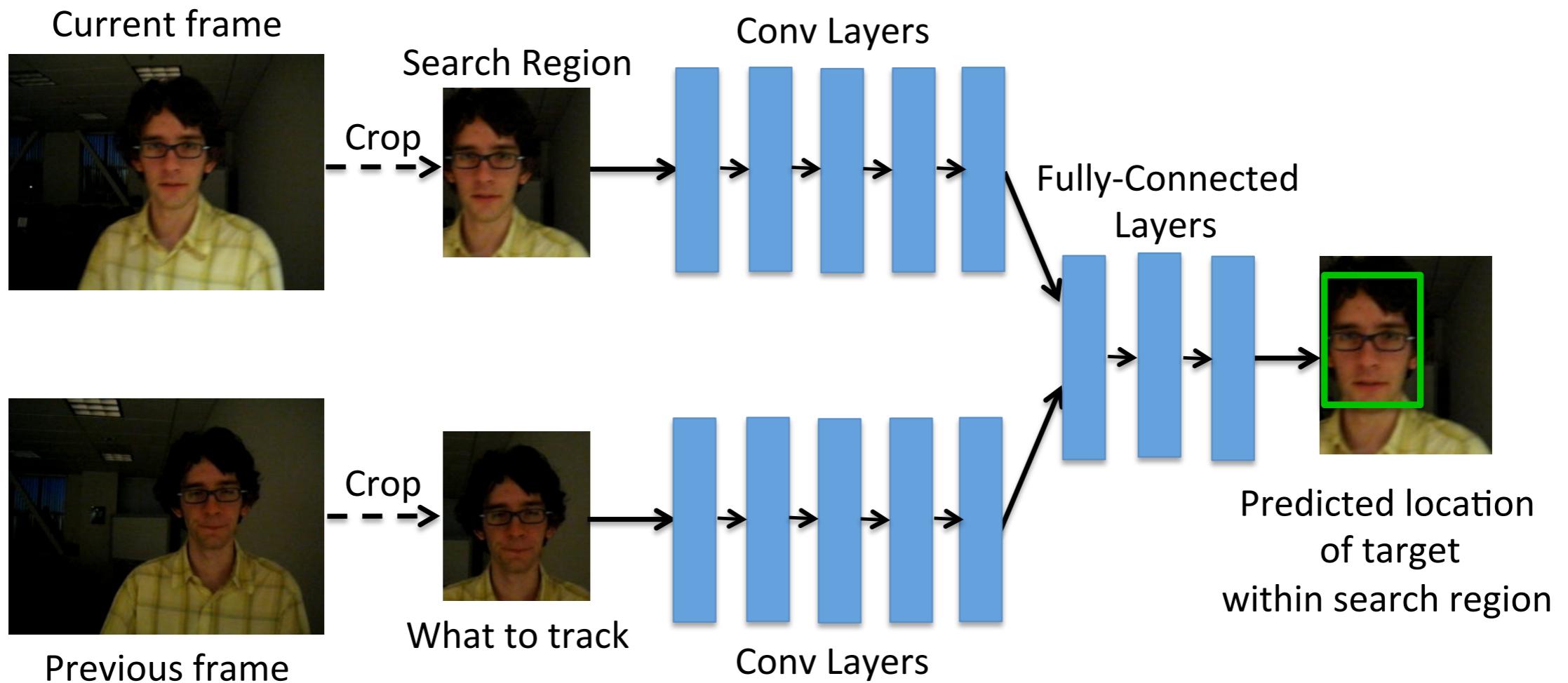
- Recently, there have been works that have tried to explore the employment of tracking using deep learning features.
- As efficiency is key, strategy is to learn from a large ensemble of labeled offline videos.
- Of particular interest are two papers,
 1. Held, Thrun, Savarese “Learning to Track at 100 FPS with Deep Regression Networks”, ECCV 2016.
 2. Bertinetto, Valmadre, Henriques, Vedaldi, Torr “Fully-Convolutional Siamese Networks for Object Tracking”, arXiv 2016.



Today

- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks

Deep Regression Networks



Deep Regression Networks

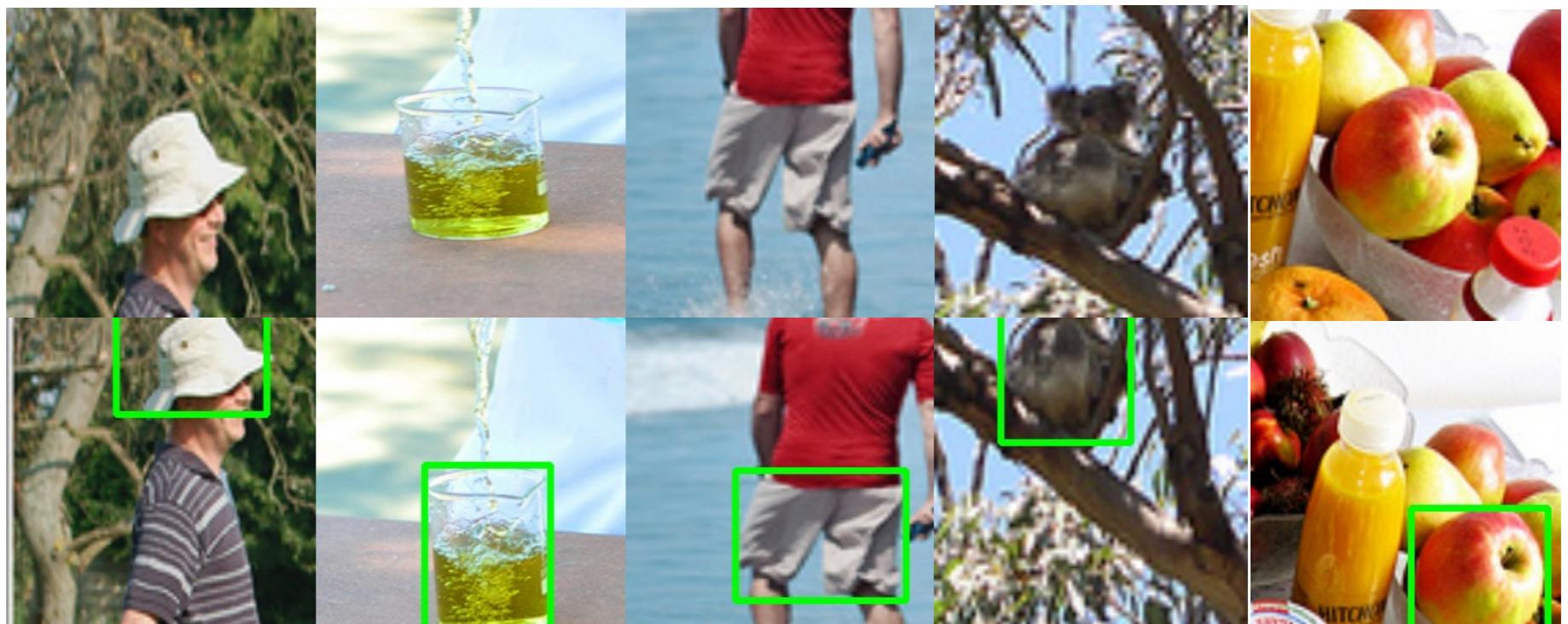
Previous
video frame
centered on
object



Current video frame,
shifted, with
ground-truth
bounding box

Deep Regression Networks

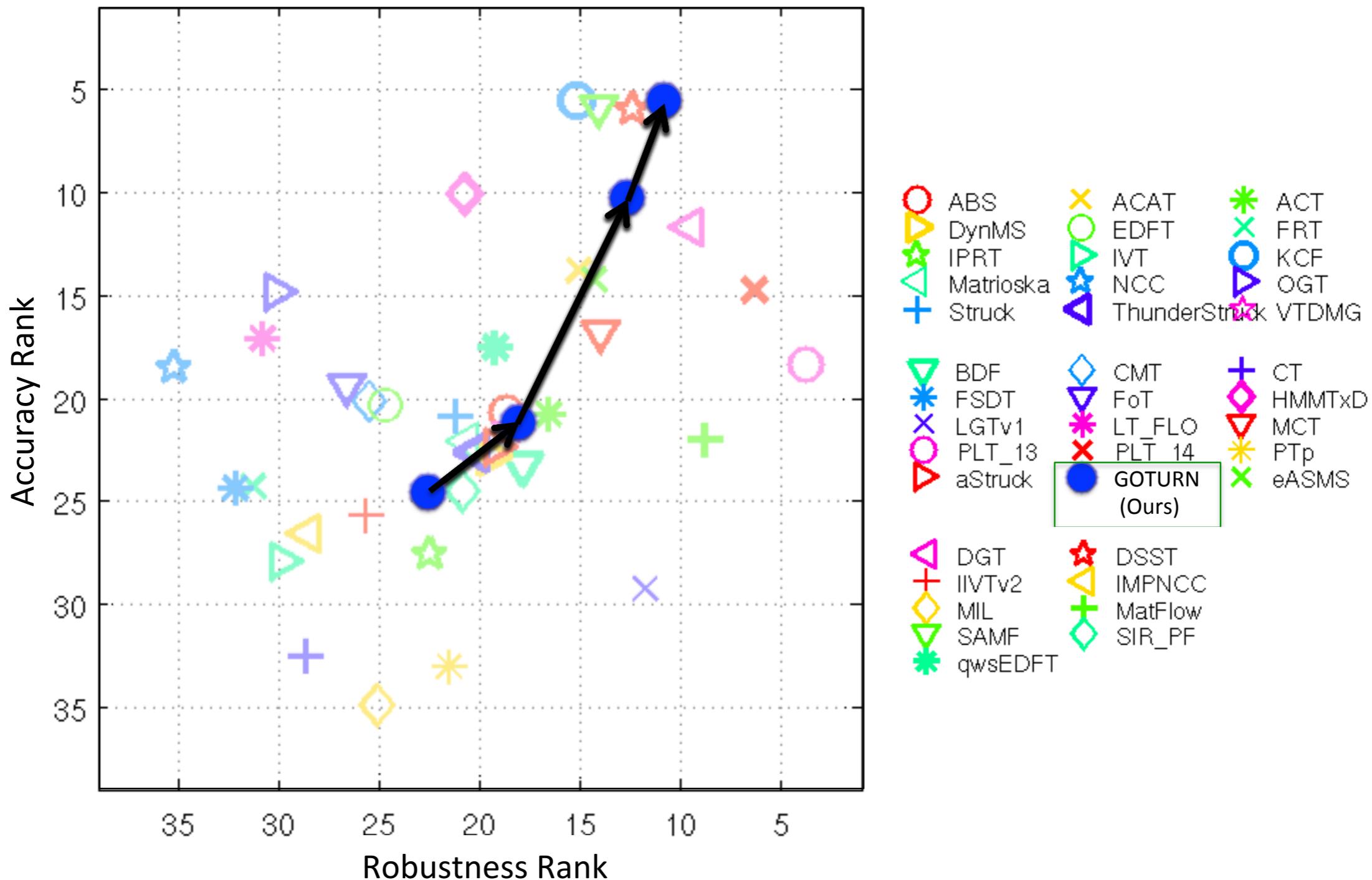
Image
centered on
object



Shifted image
with ground-truth
bounding box



Deep Regression Networks

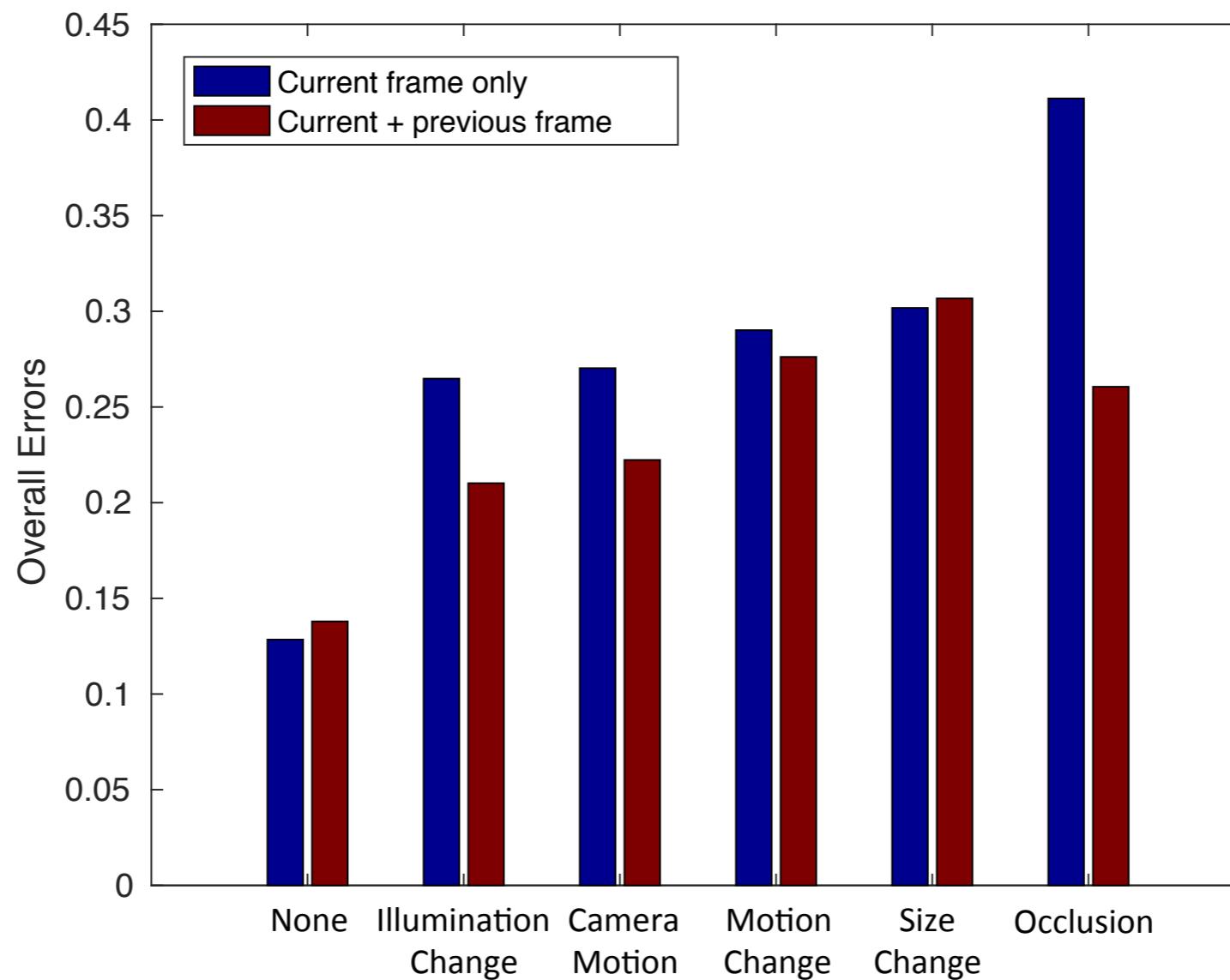


How does it work?

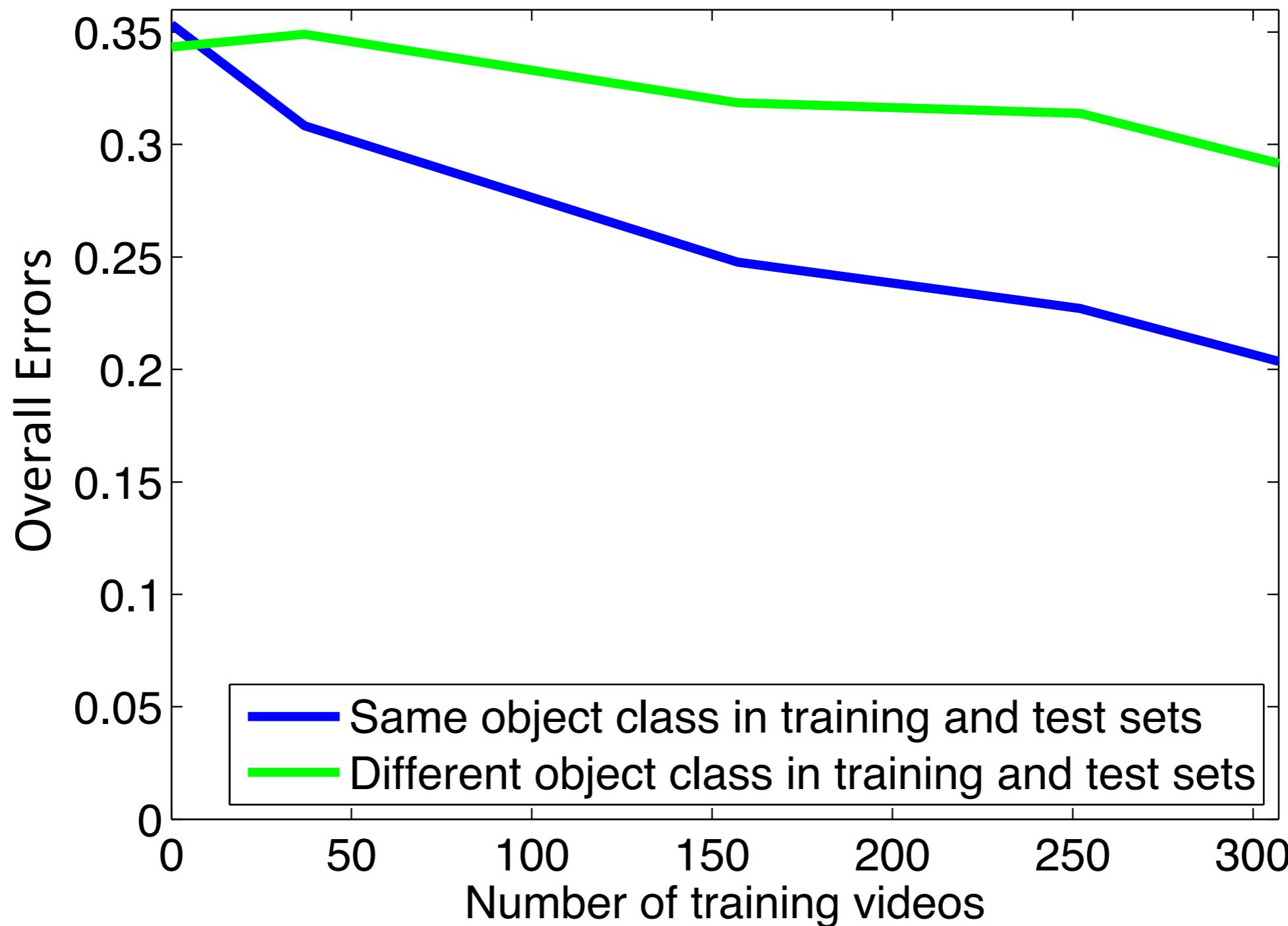
- Two hypotheses,
 1. The network compares the previous frame to the current frame to find the target object in the current frame.
 2. The network acts as a local generic “object detector” and simply locates the nearest “object.”

How does it work?

- Two hypotheses,
 1. The network compares the previous frame to the current frame to find the target object in the current frame.
 2. The network acts as a local generic “object detector” and simply locates the nearest “object.”



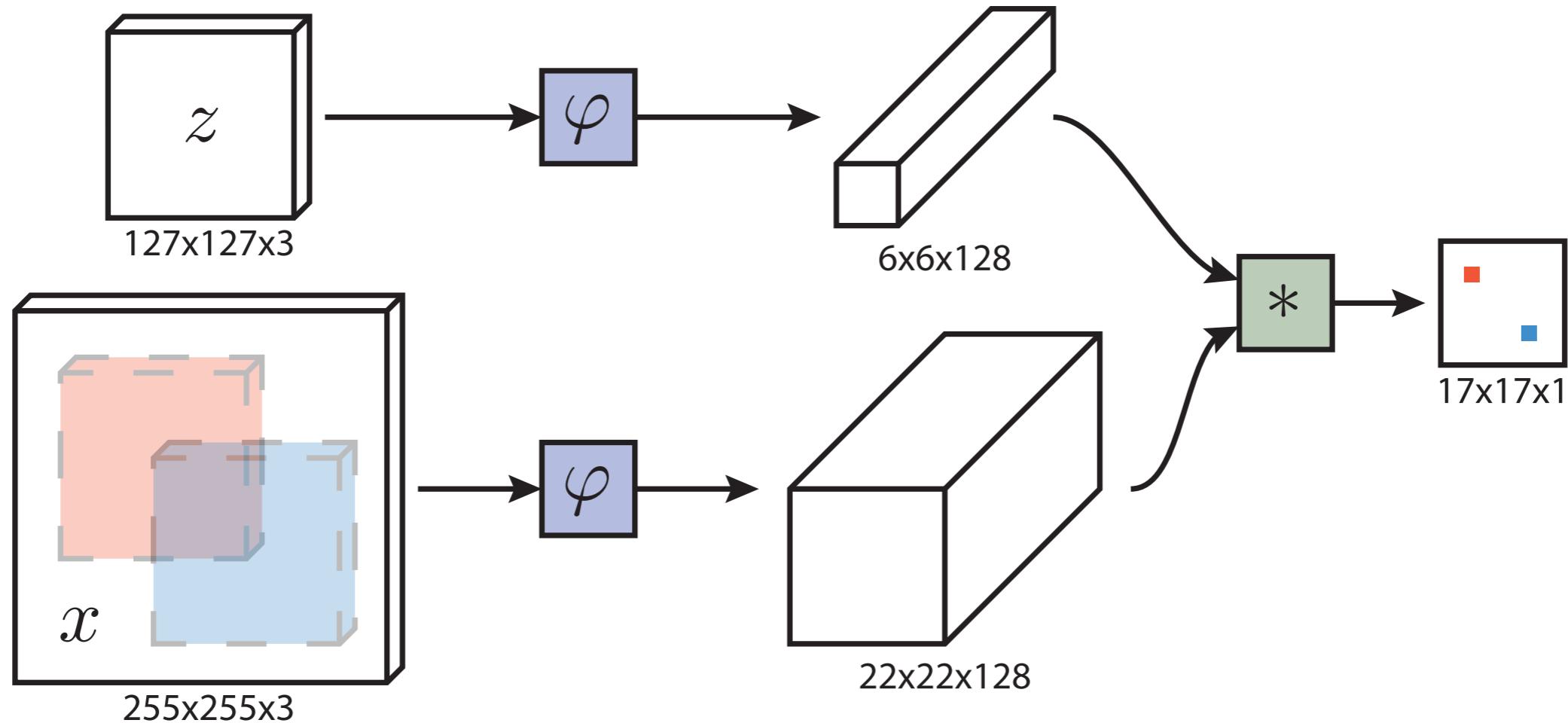
Out-of-distribution generalisation



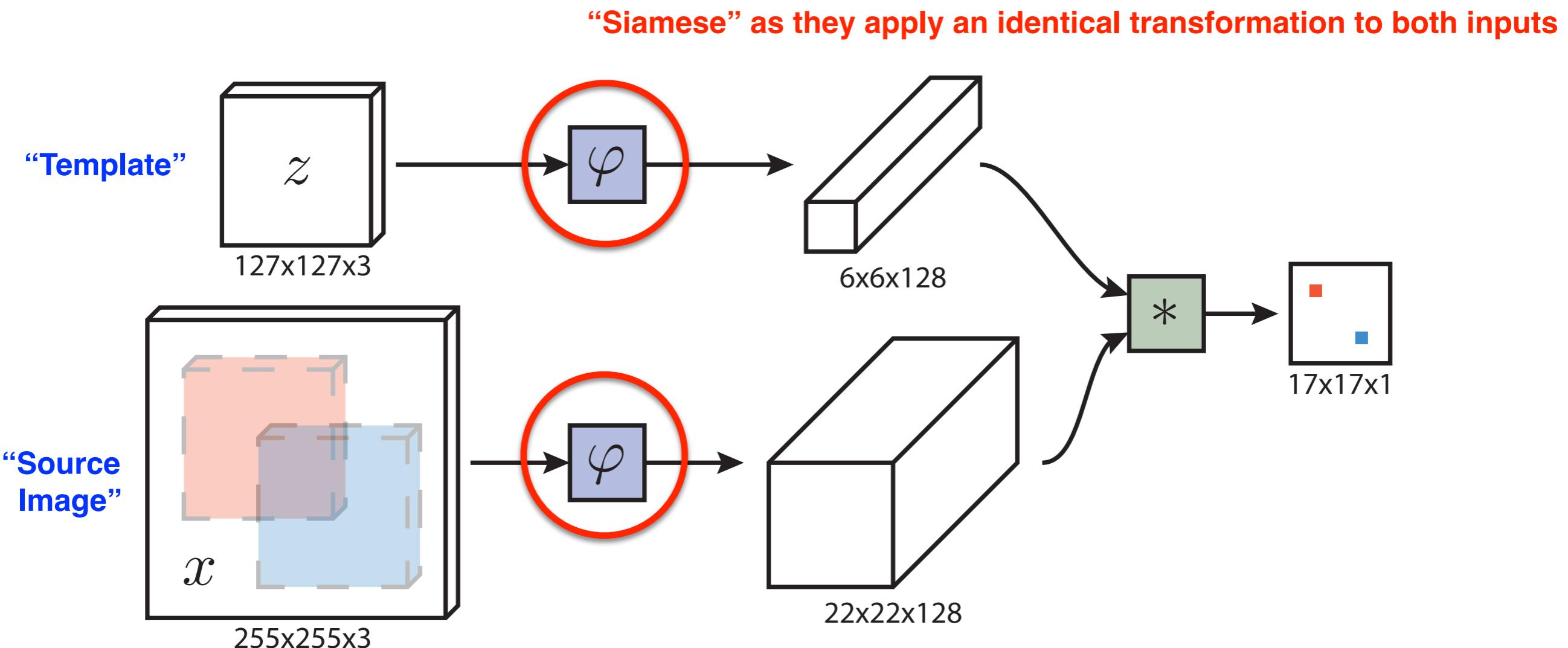
Today

- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks

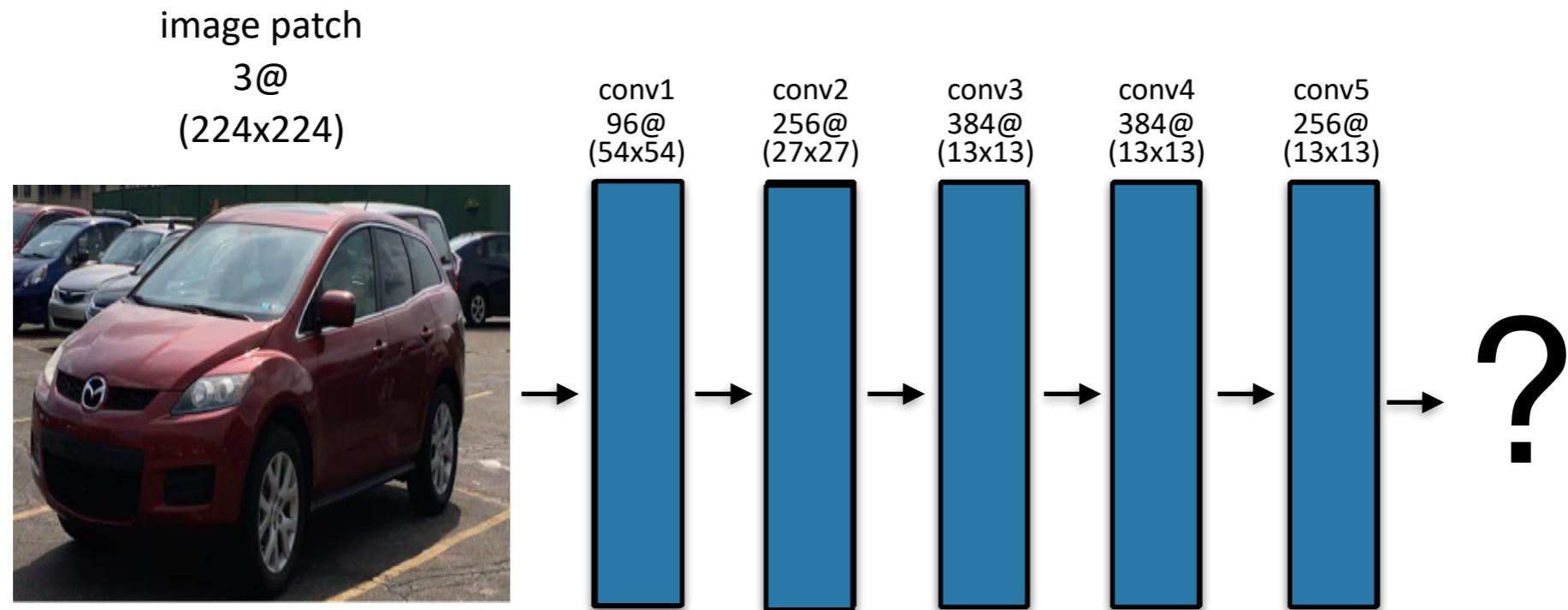
Fully Convolutional Siamese Networks



Fully Convolutional Siamese Networks



Fully Convolutional

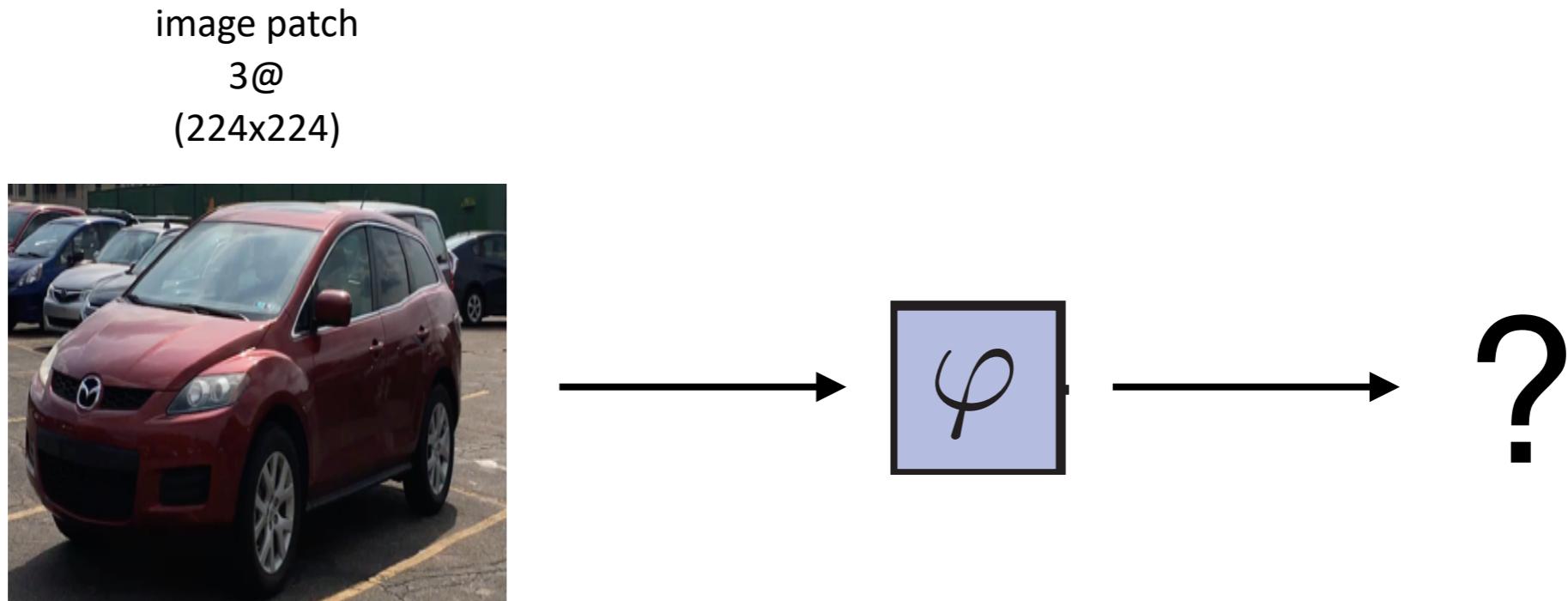


Feature transform commutes with displacement operator

$$L_\tau x[t] = x[t - \tau] \quad (\text{possibly with stride } s \neq 1)$$

$$L_\tau \varphi(x) = \varphi(L_{s\tau} x)$$

Fully Convolutional



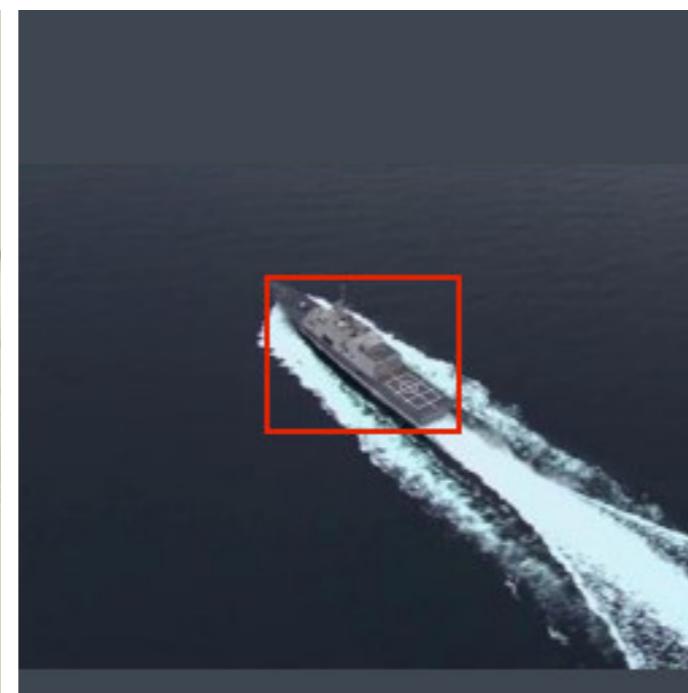
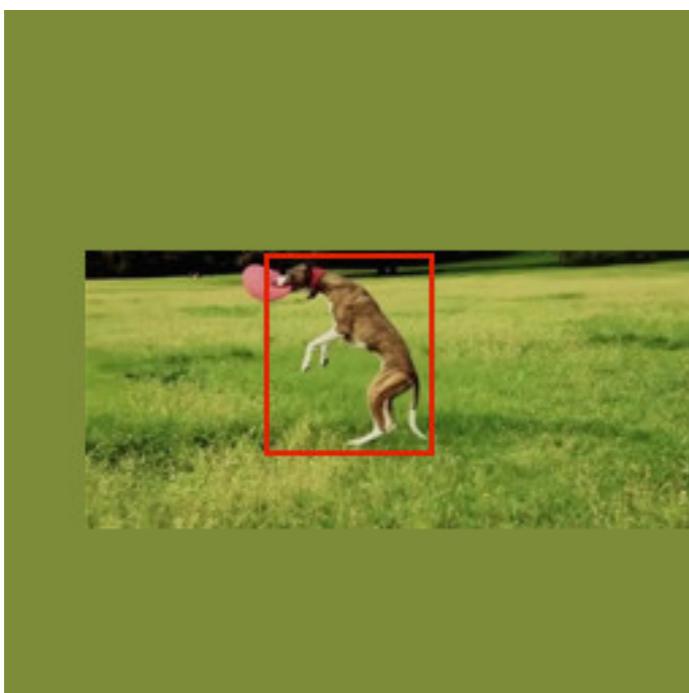
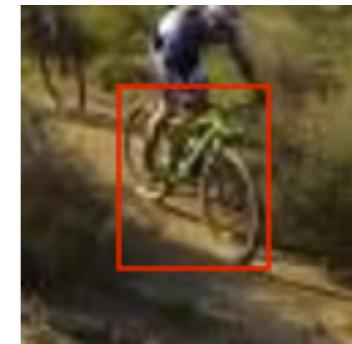
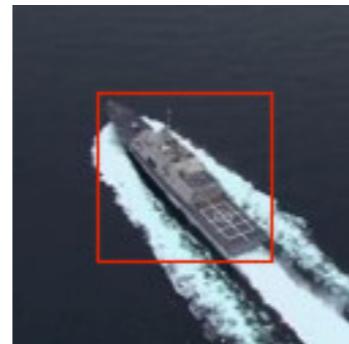
Feature transform commutes with displacement operator

$$L_\tau x[t] = x[t - \tau] \quad (\text{possibly with stride } s \neq 1)$$

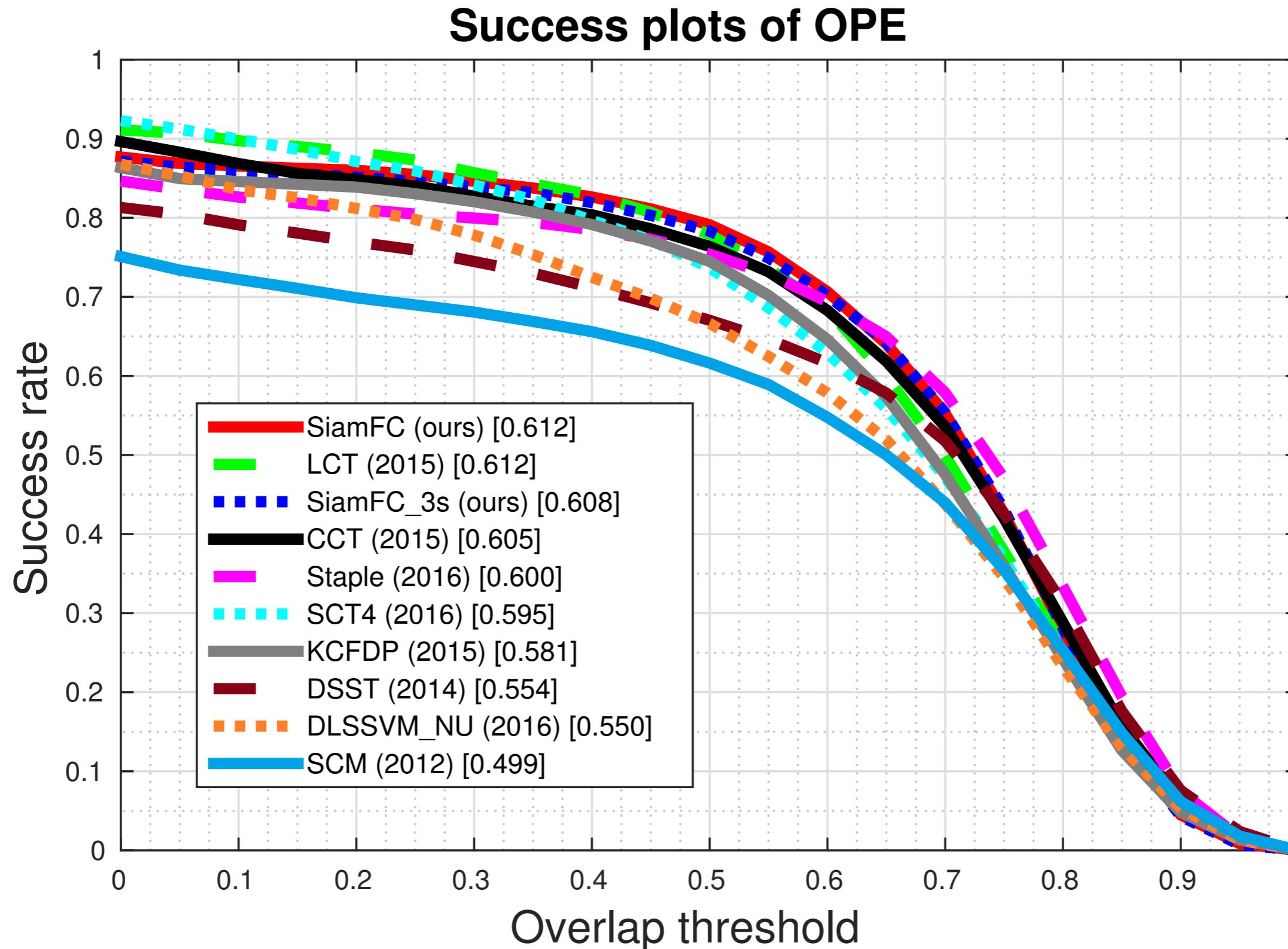
$$L_\tau \varphi(x) = \varphi(L_{s\tau} x)$$

Fully Convolutional Siamese Networks

- Example pairs from training sequences

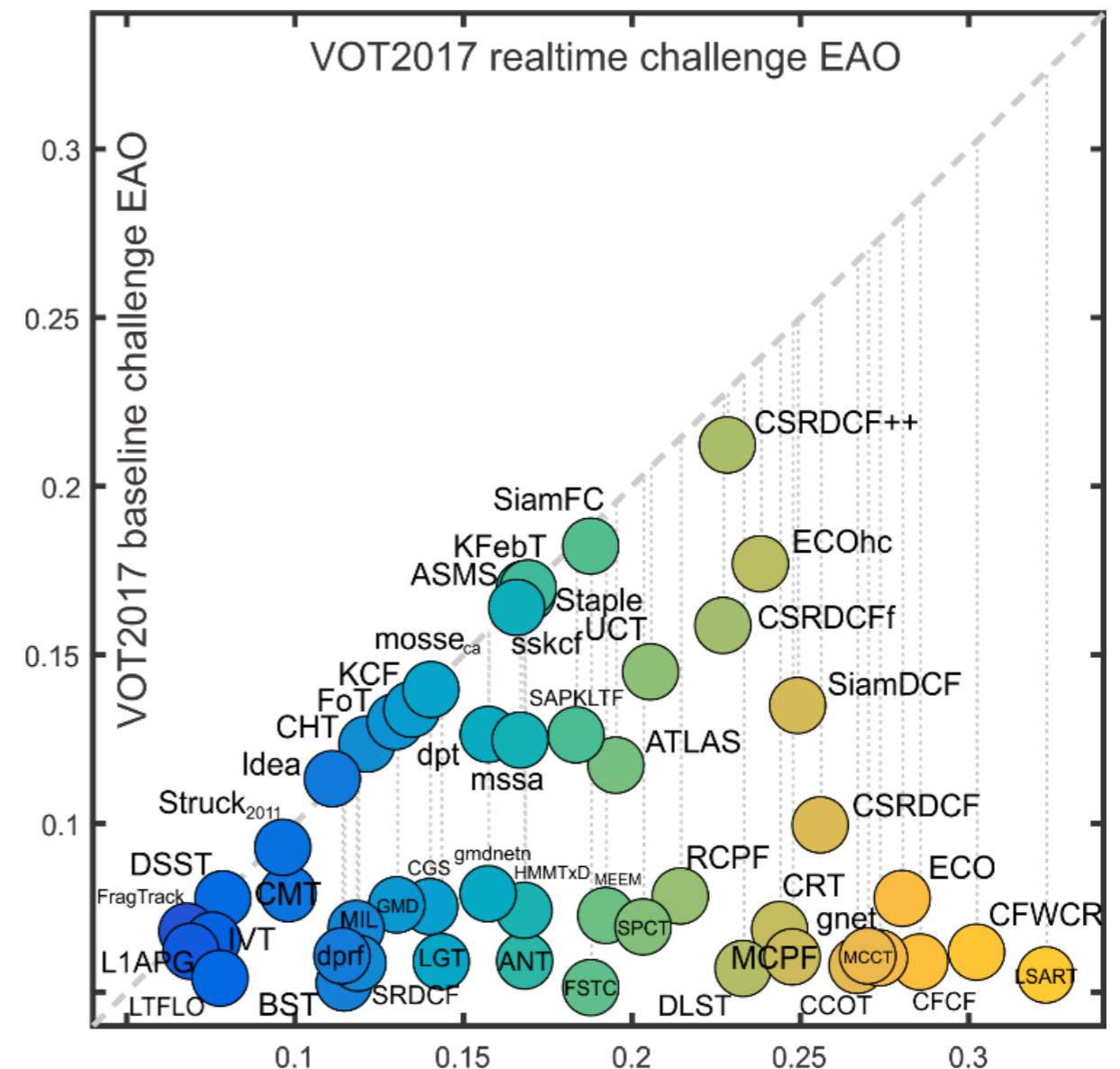


Fully Convolutional Siamese Networks

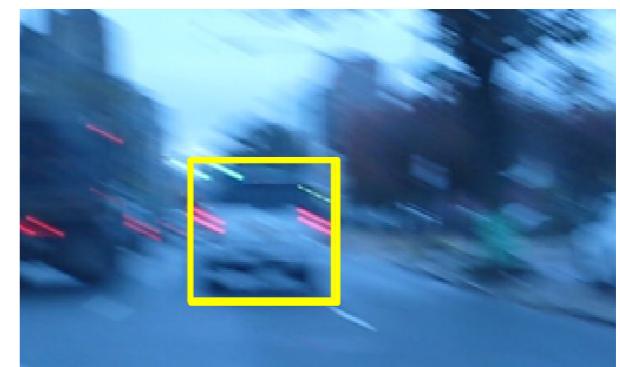
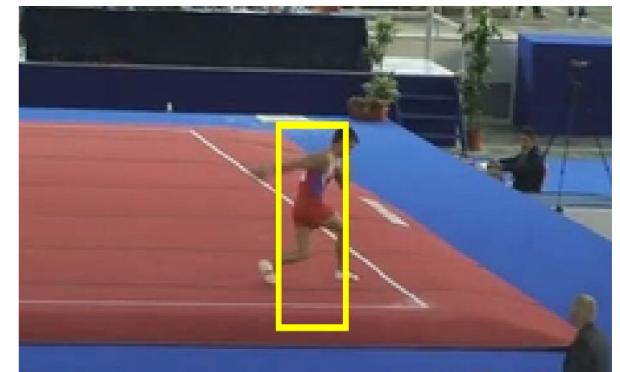
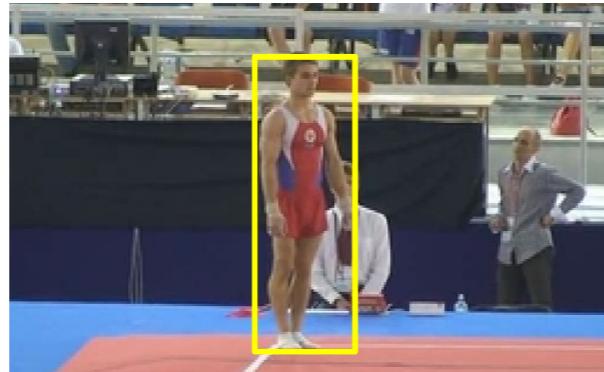


Fully Convolutional Siamese Networks

- Real-time evaluation:
frames dropped if
tracker is not ready
 - 22nd overall
 - 2nd in real-time



Fully Convolutional Siamese Networks



Frame 1 (init.)

Frame 50

Frame 100

Frame 200

Fully Convolutional Siamese Networks



Frame 1 (init.)

Frame 50

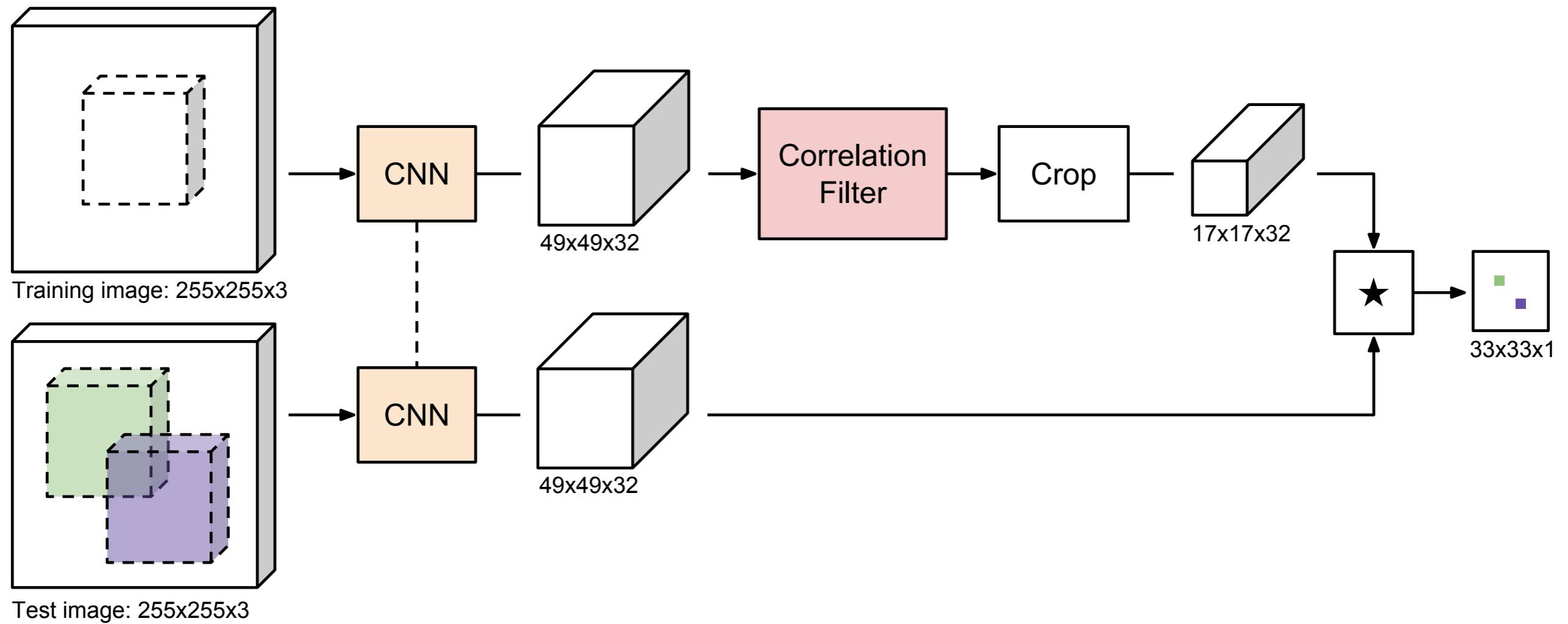
Frame 100

Frame 200

Today

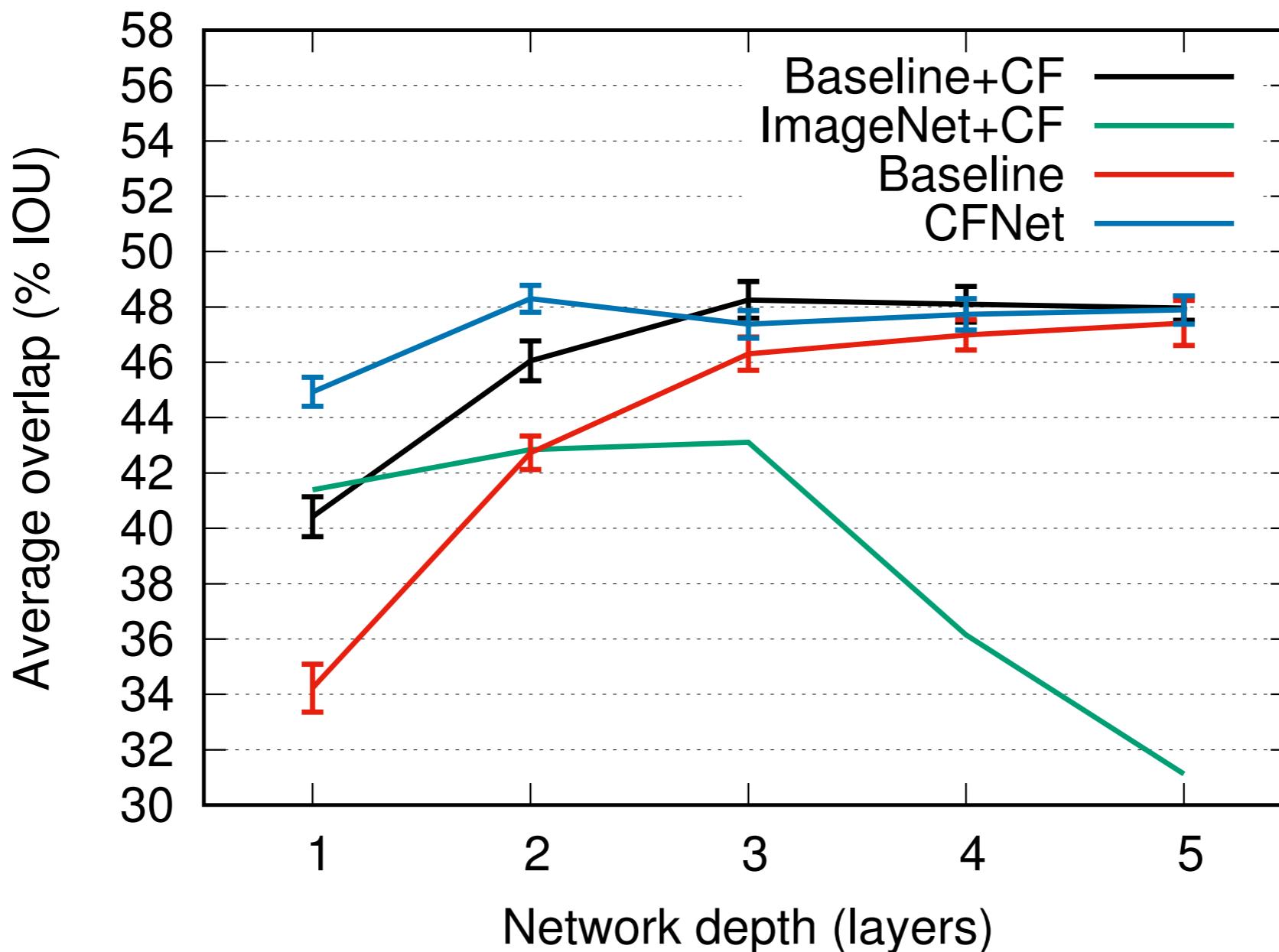
- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks

Correlation Filter networks



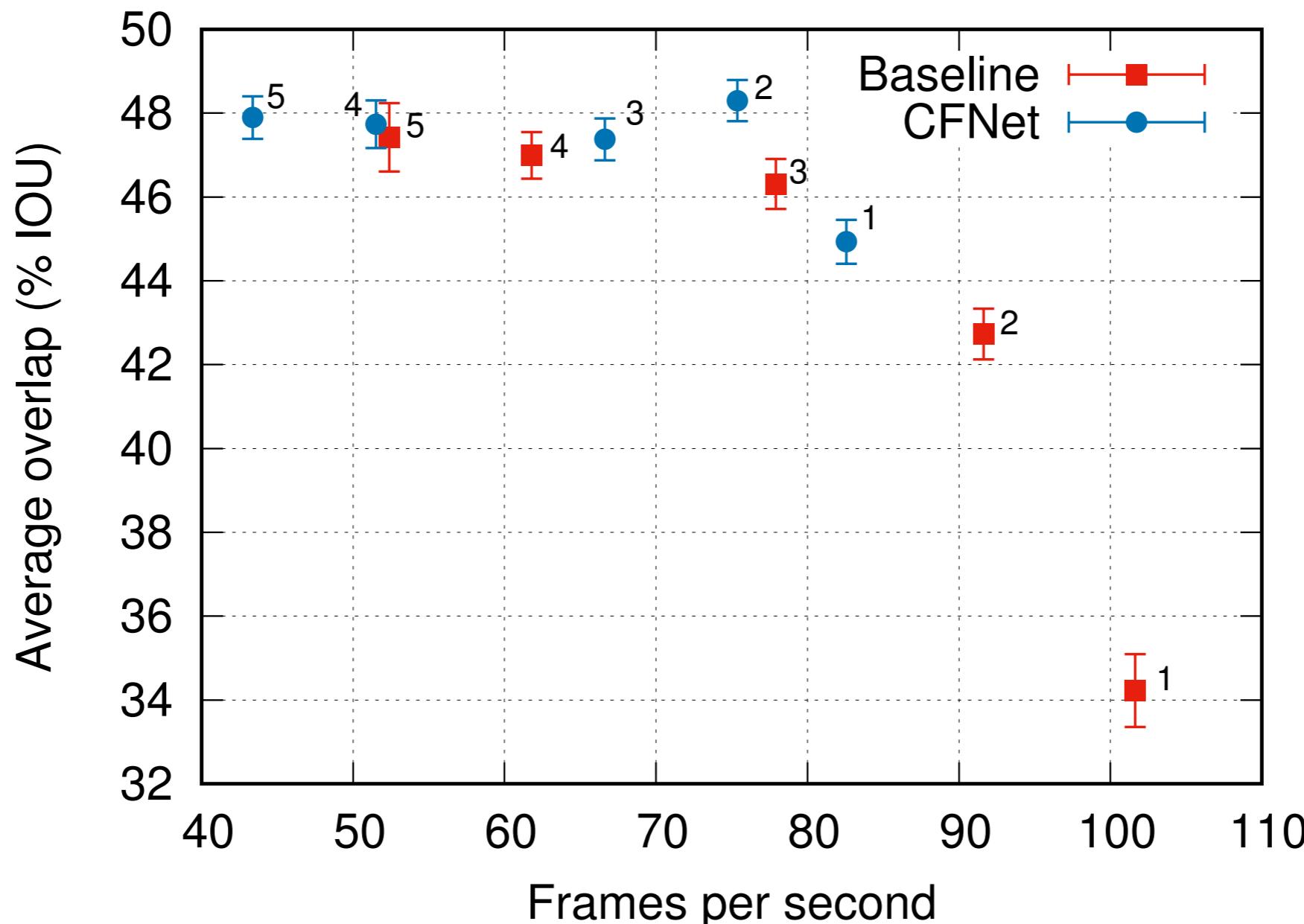
Correlation Filter networks

- Does it help to include Correlation Filter during training?



Correlation Filter networks

- Does it help to include Correlation Filter during training?



Today

- Convolution, circulant Toeplitz matrices, FFT
- Correlation filter
- Deep object tracking
 - Deep regression networks
 - Fully-convolutional Siamese networks
 - Correlation filter networks