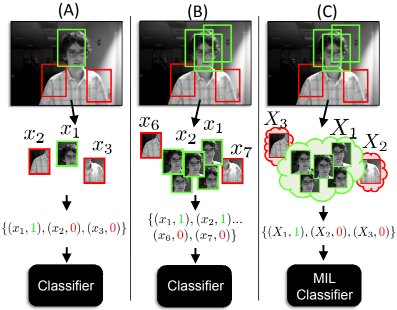
Visual Tracking
Part II: Case Studies

CSE P576 Autumn 2021
Vitaly Ablavsky

Tracking as Learning before Deep Learning

Visual Tracking with Online Multiple Instance Learning



[B. Babenko, M.-H. Yang, and S. Belongie,
"Visual Tracking with Online Multiple Instance Learning," CVPR 2009]

Visual Tracking with Online Multiple Instance Learning

$$y_i = \max_j (y_{ij})$$

$$\log \mathcal{L} = \sum_i \Big(\log p(y_i | X_i) \Big)$$

$$p(y_i | X_i) = 1 - \prod_j \Big(1 - p(y_i | x_{ij}) \Big)$$

[B. Babenko, M.-H. Yang, and S. Belongie,
"Visual Tracking with Online Multiple Instance Learning," CVPR 2009]

Tracking-Learning-Detection

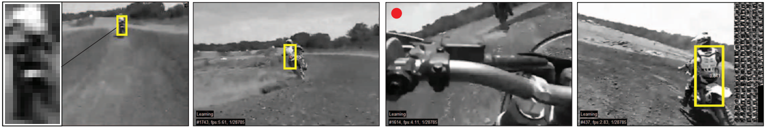
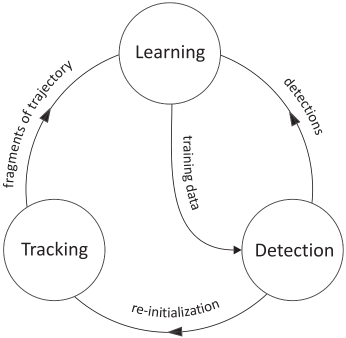
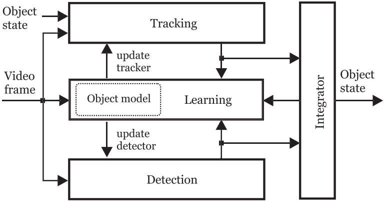


Fig. 1. Given a single bounding box defining the object location and extent in the initial frame (LEFT), our system tracks, learns, and detects the object in real time. The red dot indicates that the object is not visible.

[Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," PAMI 2012]

Tracking-Learning-Detection





[Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," PAMI 2012]

Tracking-Learning-Detection

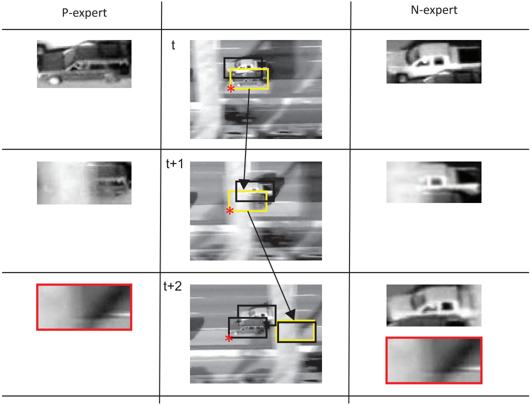


Fig. 7. Illustration of the examples output by the P-N experts. The third row shows error compensation.

[Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-Learning-Detection,” PAMI 2012]

Tracking by Correlation before Deep Learning

Fast and Precise Object Localization with Correlation Filters

Motivation

Given a query (left) find its match in the test image (right)

Motivation

Given a query (left) find its match in the test image (right)

Problem Definition

Learning stage:

- Input: example signal(s) $\mathbf{x}$ in 1D or 2D
- Output: a scoring function $f : \mathbf{x}' \to \mathbb{R}$

Detection stage:

- Input: $\mathbf{Z}$, a “long” 1D signal or a “large” 2D image
- Output: sub-signal $\mathbf{z}$ of $\mathbf{Z}$ for which f is highest
  (size of $\mathbf{z}$ is the same as the size of training signal $\mathbf{x}$)

Approach I: Let the Filter be Idnetical to Query

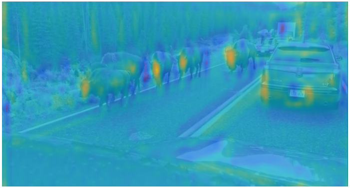
$$f(\mathbf{x}'; \mathbf{x}) \doteq \mathbf{x}^T \mathbf{x}'$$

$\mathbf{x}$

Approach I: Let the Filter be Idnetical to Query

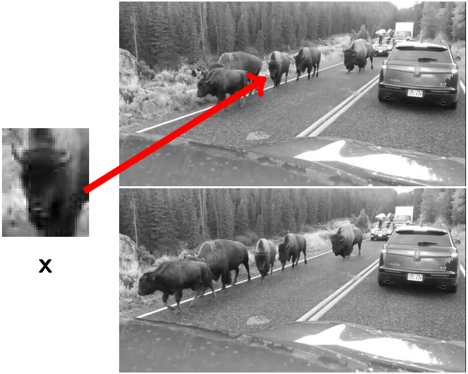
$$f(\mathbf{x}'; \mathbf{x}) \doteq \mathbf{x}^T \mathbf{x}'$$

$\mathbf{x}$

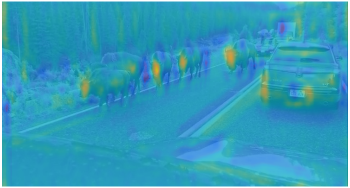


Approach I: Let the Filter be Idnetical to Query

$$f(\mathbf{x}'; \mathbf{x}) \doteq \mathbf{x}^T \mathbf{x}'$$



$\mathbf{x}$



Approach I: Let the Filter be Idnetical to Query

$$f(\mathbf{x}';\mathbf{x}) \doteq \mathbf{x}^T\mathbf{x}'$$

Approach I: Let the Filter be Identical to Query

$$f(\mathbf{x}'; \mathbf{x}) \doteq \mathbf{x}^T \mathbf{x}'$$

- correlation peak's shape depends on the template structure
- robust to deformation: "picks up" other bison in the scene
- not discriminative: highlights car parts

Details of Template-Matching

Template-matching can be faster in the frequency domain:

1. compute Discrete Fourier Transform (DFT) of the "big" image: $\mathbf{Z} \stackrel{\mathrm{DFT}}{\rightarrow} \hat{\mathbf{Z}}$
2. compute DFT of template: $\mathbf{x} \stackrel{\mathrm{DFT}}{\rightarrow} \hat{\mathbf{x}}$ (same size as $\hat{\mathbf{Z}}$)
3. point-wise multiply (taking Hermitian transpose) $\hat{\mathbf{y}} = \hat{\mathbf{Z}} \odot \hat{\mathbf{x}}^H$
4. bring back to spatial domain: $\mathbf{y} = \mathrm{inverse\text{-}DFT}(\hat{\mathbf{y}})$

Approach II: Optimize the Filter

- use “many” templates during training
- shape the desired filter response

Approach II: Optimize the Filter

- use "many" templates during training
- shape the desired filter response

1. "Multivariant Technique for Multiclass Pattern Recognition," C. F. Hester et al., J. Applied Optics, 1980
2. "Average of Synthetic Exact Filters," Bolme et al., CVPR 2009
3. "Accurate Scale Estimation for Robust Visual Tracking," Danelljan et al., BMVC 2014
4. "High-Speed Tracking with Kernelized Correlation Filters," Henriques et al., PAMI 2015
5. "Zero-Aliasing Correlation Filters for Object Recognition," Fernandez et al., PAMI 2015 (one of co-authors in Dayton)

Approach II: Optimize the Filter

- use “many” templates during training
- shape the desired filter response

1. “Multivariant Technique for Multiclass Pattern Recognition,” C. F. Hester et al., J. Applied Optics, 1980
2. “Average of Synthetic Exact Filters,” Bolme et al., CVPR 2009
3. “Accurate Scale Estimation for Robust Visual Tracking,” Danelljan et al., BMVC 2014
4. “High-Speed Tracking with Kernelized Correlation Filters,” Henriques et al., PAMI 2015
5. “Zero-Aliasing Correlation Filters for Object Recognition,” Fernandez et al., PAMI 2015 (one of co-authors in Dayton)

Formulation as a Regression Problem

$$f(\mathbf{x}; \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w}^{\text{opt}} = \min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

Formulation as a Regression Problem

$$f(\mathbf{x};\mathbf{w}) \doteq \mathbf{w}^T\mathbf{x}$$

$$\mathbf{w}^{\mathrm{opt}} = \min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i) - y_i)^2 + \lambda\|\mathbf{w}\|^2$$

Define X such that its i-th row is $\mathbf{x}_i$, then :

$$\text{for real-valued inputs : } \mathbf{w}^{\mathrm{opt}} = (X^T X + \lambda \mathrm{I})^{-1} X^T \mathbf{y}$$

$$\text{for complex-valued inputs : } \mathbf{w}^{\mathrm{opt}} = (X^H X + \lambda \mathrm{I})^{-1} X^H \mathbf{y}$$

Formulation as a Regression Problem

$$f(\mathbf{x}; \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w}^{\mathrm{opt}} = \min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2$$

Define X such that its i-th row is $\mathbf{x}_i$, then :

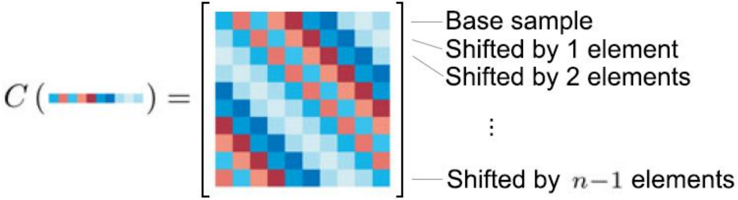
$$\text{for real-valued inputs : } \mathbf{w}^{\mathrm{opt}} = (X^T X + \lambda \mathrm{I})^{-1} X^T \mathbf{y}$$

$$\text{for complex-valued inputs : } \mathbf{w}^{\mathrm{opt}} = (X^H X + \lambda \mathrm{I})^{-1} X^H \mathbf{y}$$

Remarks:

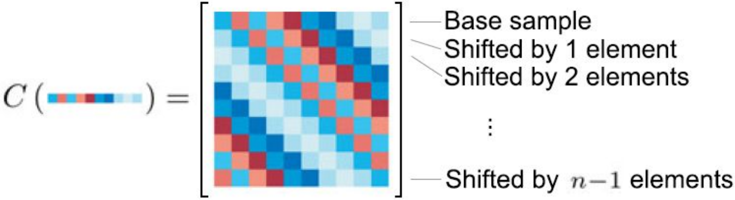
- Brute-force solution of the linear system is expensive: a 50x50 image patch yields a 2500 dim vector; if we have 2500 examples, matrix inversion becomes impractical in real time
- Need a recipe for getting “good” training samples

Obtaining Training Examples via Circular Shifts



Let $X = C(\mathbf{x})$
then $X = F\mathrm{diag}(\hat{\mathbf{x}})F^H$ and F is the DFT matrix

Obtaining Training Examples via Circular Shifts



Let $X = C(\mathbf{x})$
then $X = F\mathrm{diag}(\hat{\mathbf{x}})F^H$ and F is the DFT matrix

Computationally-Efficient Solution

Recall, the problem we are solving:
find $\mathbf{w}$ such that for each example $\mathbf{x}_i$, the score $f(\mathbf{x}_i; \mathbf{w}) = y_i$ The

closed-form solution takes the form of

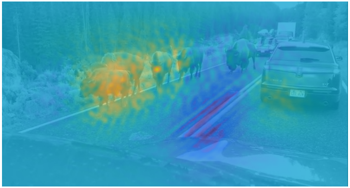
$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}} \odot \hat{\mathbf{x}}^* + \lambda}$$

Notes:

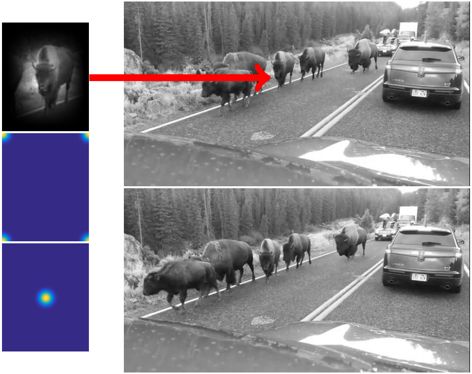
- only point-wise multiplications required
- time complexity bound by the cost of DFT, i.e., $O(n \log n)$
- compare to kernel ridge regression: $O(n^3)$

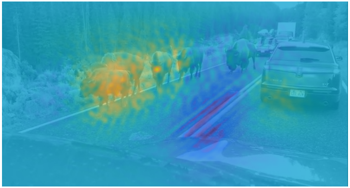
Learned Correlation Filter in Action

Learned Correlation Filter in Action

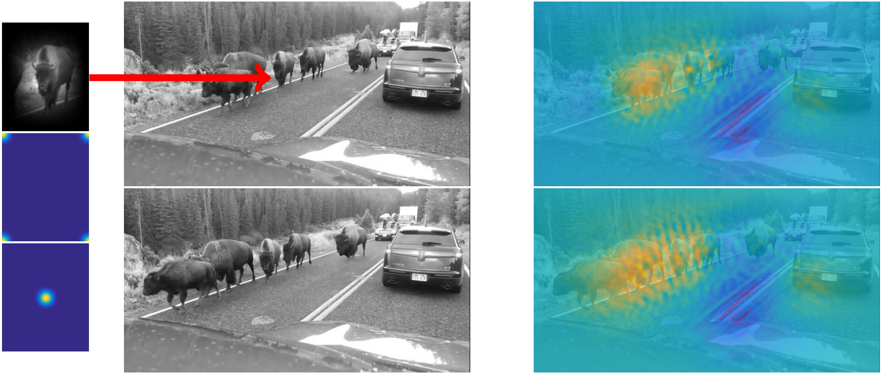


Learned Correlation Filter in Action

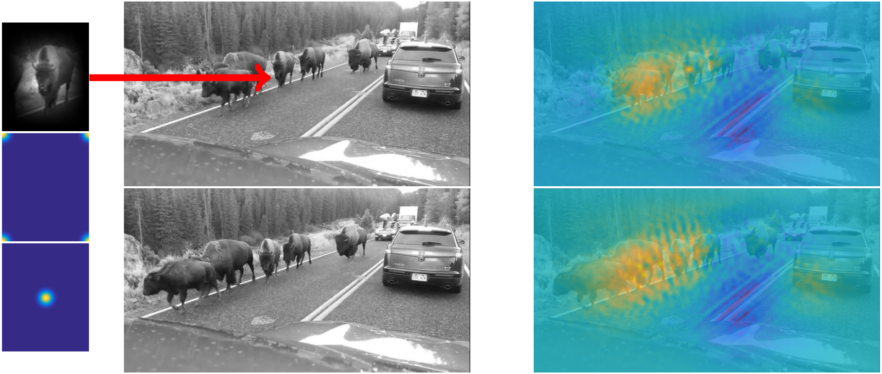




Learned Correlation Filter in Action



Learned Correlation Filter in Action



- now, correlation peak's shape follows the prescribed pattern
- specialized to "our" bison, low response for the rest
- low response on the vehicle

Correlation Filter Extensions

- multiple channels, e.g., FHOG
- kernelized: linear, Gaussian, etc.
- multiple spatial scales
- control aliasing (due to finite signal extent)

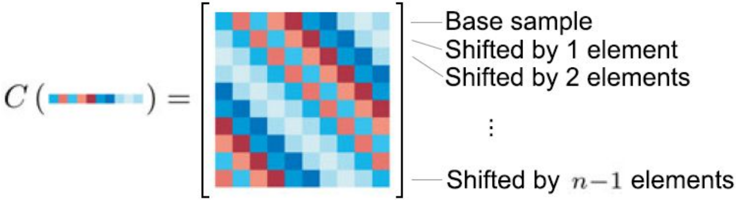
Correlation Filter Extensions

- multiple channels, e.g., FHOG
- kernelized: linear, Gaussian, etc.
- multiple spatial scales
- control aliasing (due to finite signal extent)

From Inner Products to Kernels

- Thus far, similarity between vectors $\mathbf{x}'$ and $\mathbf{x}$ defined as $\mathbf{x}^T\mathbf{x}'$
- Let $\varphi(\mathbf{x})$ map $\mathbf{x}$ into another space (typically higher-dim)
- Define *kernel* function $k(\mathbf{x}', \mathbf{x}) \doteq \varphi(\mathbf{x})^T\varphi(\mathbf{x}')$
- In practice we want to compute $k(\mathbf{x}', \mathbf{x})$ directly, avoiding φ

Kernelized Correlation Filter



given a kernel $k : (\mathbf{x}, \mathbf{x}') \longrightarrow \mathbb{R}$,
define *kernel correlation* vector $\mathbf{k}^{\mathbf{x}\mathbf{x}'}$ as

$$k_i^{\mathbf{x}\mathbf{x}'} \doteq k(\mathbf{x}', P^{i-1}\mathbf{x}),$$

where P is a *cyclic shift operator*

Kernelized Correlation Filter

Key equations:

$$\mathbf{w} = \sum_i \alpha_i \varphi(\mathbf{x}_i)$$

$$f(\mathbf{z}) = \mathbf{w}^T \mathbf{z} = \sum_{i=1}^{n} \alpha_i k(\mathbf{z}, \mathbf{x}_i)$$

$$\hat{\alpha} = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}$$

$$\hat{f}(\mathbf{z}) = \hat{\mathbf{k}}^{\mathbf{xz}} \odot \hat{\alpha}$$

Notes

- Time complexity is again bound by the DFT, hence $O(n \log n)$

Multi-channel Correlation Filter

Suppose our signal comprises multiple channels $c = 1, \ldots, c_{\max}$
Example: FHOG 31 channels: $3 \times$ 9-bin histograms + 4 texture

- Q: How expensive is kernel correlation?
- A: Time complexity scales linearly with $c_{\max}$
- For linear kernel:

$$\mathbf{k}^{\mathbf{x}\mathbf{x}'} = \text{inverse-DFT}\left(\sum_c \hat{\mathbf{x}}_c^* \odot \hat{\mathbf{x}}_c'\right)$$

FHOG Correlation Filter in Action (linear kernel)

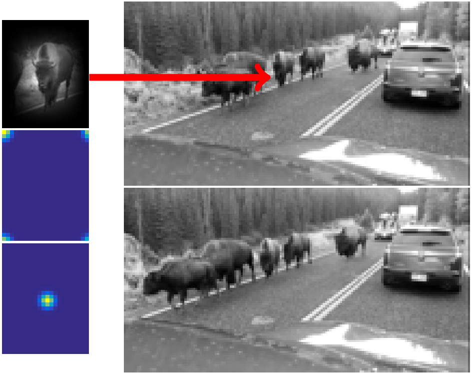
- FHOG cell is 4x4 pixels, thus **x**, **y** are smaller than template

FHOG Correlation Filter in Action (linear kernel)

- FHOG cell is 4x4 pixels, thus **x**, **y** are smaller than template

FHOG Correlation Filter in Action (linear kernel)

- FHOG cell is 4x4 pixels, thus **x**, **y** are smaller than template



FHOG Correlation Filter in Action (linear kernel)

- FHOG cell is 4x4 pixels, thus **x**, **y** are smaller than template

FHOG Correlation Filter in Action (linear kernel)

- FHOG cell is 4x4 pixels, thus **x**, **y** are smaller than template

- correlation surface is strongly peaked in the training and test images
- unlike standard template matching, response is strong only for that particular instance

Practical consideraitons

Note: correlation filters defined thus far only "work" if the test image has the same size as $\mathbf{x}$

Q: how to correlate a filter with $\mathbf{Z}$ of size larger than $\mathbf{x}$?

A:

- Transform the filter back to spatial domain, i.e., transform

$$\hat{f}(\mathbf{z}) = \hat{\mathbf{k}}^{\mathbf{xz}} \odot \hat{\alpha}$$

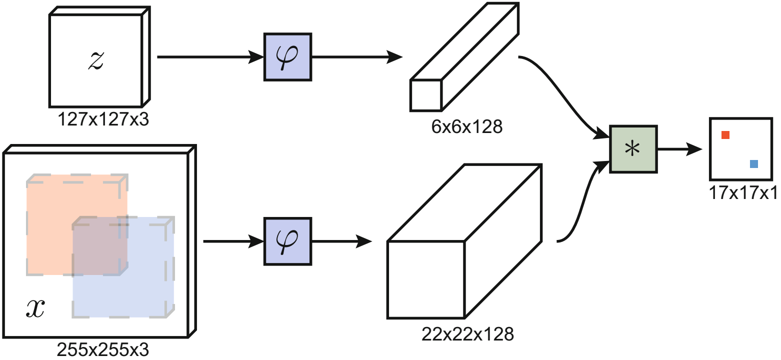
- Modify the learned filter in the DFT domain:
  pad with zeros to match the size of $\mathbf{Z}$

Conclusions

Advantages of correlation filters:

- good accuracy out of the box
- controlled by a handful of "knobs"
- failure modes "easy" to understand
- open-source implementations:
    1. "Accurate Scale Estimation for Robust Visual Tracking," Danelljan et al., BMVC 2014: **MATLAB**, **C++**
    2. "High-Speed Tracking with Kernelized Correlation Filters," Henriques et al., PAMI 2015: **MATLAB**
    3. "Correlation Filters with Limited Boundaries," Galoogahi et al., CVPR 2015: **MATLAB**

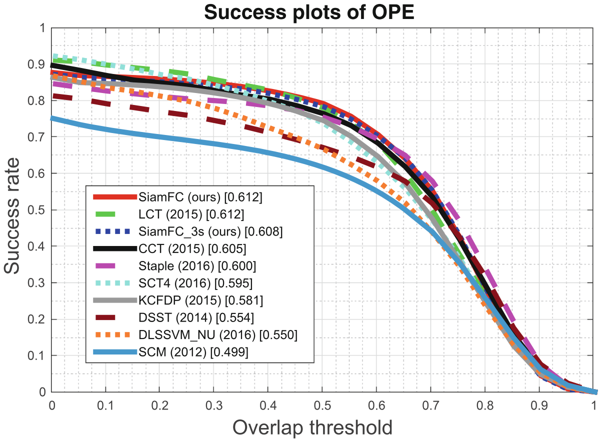
Tracking by Correlation with Deep Learning



[L. Bertinetto et al., “Fully-Convolutional Siamese Networks for Object Tracking,” ECCV 2016]

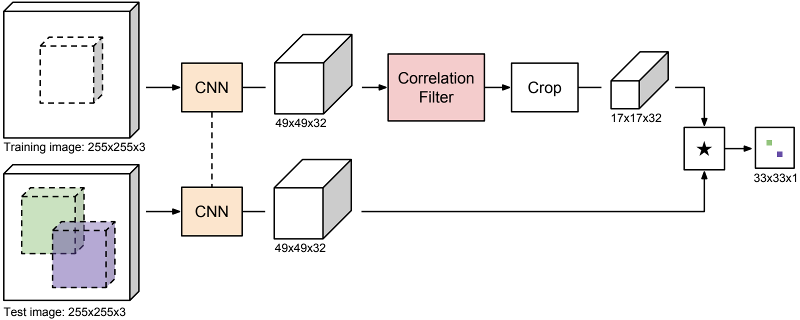
Fig. 2. Training pairs extracted from the same video: exemplar image and corresponding search image from same video. When a sub-window extends beyond the extent of the image, the missing portions are filled with the mean RGB value.

[L. Bertinetto et al., "Fully-Convolutional Siamese Networks for Object Tracking," ECCV 2016]



[L. Bertinetto et al., "Fully-Convolutional Siamese Networks for Object Tracking," ECCV 2016]

End-to-end representation learning for Correlation Filter based tracking



[J. Valmadre et al., "End-to-end representation learning for Correlation Filter based tracking," CVPR 2017]

Multi-Target Tracking via Graph-Theoretic Methods

Global Data Association for Multi-Object Tracking Using Network Flows

Detection input　　Tracking result

[L. Zhang, Y. Li, and R. Nevatia, "Global Data Association for Multi-Object Tracking Using Network Flows," CVPR 2008]

Global Data Association for Multi-Object Tracking Using Network Flows

$$
\begin{aligned}
\mathcal{T}^* &= \operatorname*{\mathbf{argmax}}_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) \\
&= \operatorname*{\mathbf{argmax}}_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T}) \\
&= \operatorname*{\mathbf{argmax}}_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T}) \qquad (1)
\end{aligned}
$$

$$
\mathcal{T}^* = \operatorname*{\mathbf{argmax}}_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k) \qquad (2)
$$

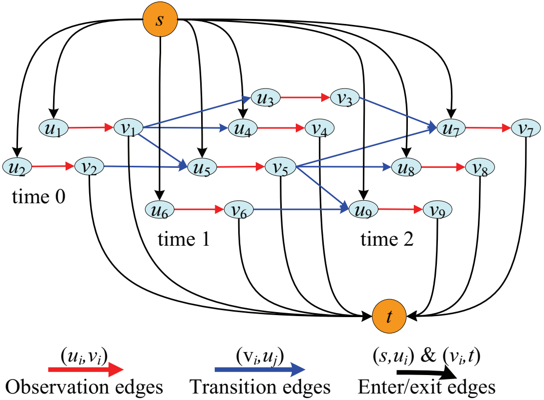
$$
\text{s.t. } \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \forall k \neq l \qquad (3)
$$

$$
P(\mathbf{x}_i|\mathcal{T}) = \begin{cases} 1 - \beta_i & \exists T_k \in \mathcal{T}, \mathbf{x}_i \in \mathcal{T}_k \\ \beta_i & \text{otherwise} \end{cases} \qquad (4)
$$

$$
\begin{aligned}
P(\mathcal{T}_k) &= P(\{\mathbf{x}_{k_0}, \mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_{l_k}}\}) \\
&= P_{entr}(\mathbf{x}_{k_0}) P_{link}(\mathbf{x}_{k_1}|\mathbf{x}_{k_0}) P_{link}(\mathbf{x}_{k_2}|\mathbf{x}_{k_1}) \\
&\quad \ldots P_{link}(\mathbf{x}_{k_{l_k}}|\mathbf{x}_{k_{l_k - 1}}) P_{exit}(\mathbf{x}_{k_{l_k}}) \qquad (5)
\end{aligned}
$$

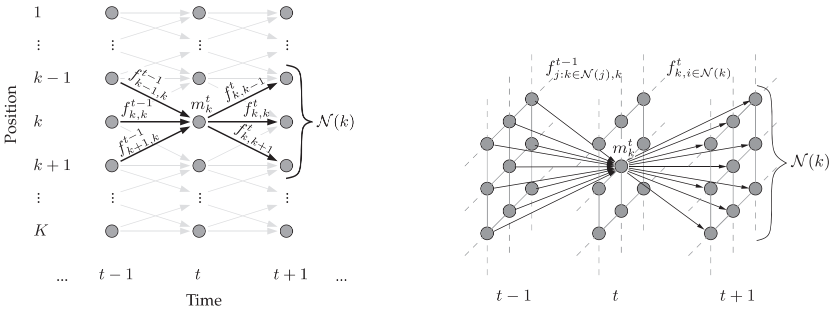
[L. Zhang, Y. Li, and R. Nevatia, “Global Data Association for Multi-Object Tracking Using Network Flows,” CVPR 2008]

Global Data Association for Multi-Object Tracking Using Network Flows



[L. Zhang, Y. Li, and R. Nevatia, "Global Data Association for Multi-Object Tracking Using Network Flows," CVPR 2008]

Multiple Object Tracking Using K-Shortest Paths Optimization



(a) Simplified flow model (b) Grid flow model

[J. Berclaz et al., "Multiple Object Tracking Using K-Shortest Paths Optimization," PAMI 2011]

Multiple Object Tracking Using K-Shortest Paths Optimization

$$\forall t, j, \quad \underbrace{\sum_{i:j \in \mathcal{N}(i)} f_{i,j}^{t-1}}_{\text{Arriving at } j \text{ at } t} = m_j^t = \underbrace{\sum_{k \in \mathcal{N}(j)} f_{j,k}^{t}}_{\text{Leaving from } j \text{ at } t}$$

$$\underbrace{\sum_{j \in \mathcal{N}(\upsilon_{\text{source}})} f_{\upsilon_{\text{source}},j}^{t}}_{\text{Leaving } \upsilon_{\text{source}}} = \underbrace{\sum_{k:\upsilon_{\text{sink}} \in \mathcal{N}(k)} f_{k,\upsilon_{\text{sink}}}^{t}}_{\text{Arriving at } \upsilon_{\text{sink}}}$$

$$\rho_i^t = \hat{P}(M_i^t = 1 \mid \mathbf{I}^t)$$

[J. Berclaz et al., “Multiple Object Tracking Using K-Shortest Paths Optimization,” PAMI 2011]

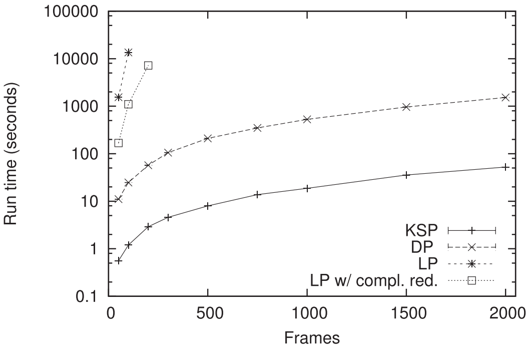
Multiple Object Tracking Using K-Shortest Paths Optimization

$$
\begin{aligned}
\text{Maximize} \quad & \sum_{t,i} \log\left(\frac{\rho_i^t}{1-\rho_i^t}\right) \sum_{j \in \mathcal{N}(i)} f_{i,j}^t \\
\text{subject to} \quad & \forall t, i, j, \quad f_{i,j}^t \geq 0 \\
& \forall t, i, \quad \sum_{j \in \mathcal{N}(i)} f_{i,j}^t \leq 1 \\
& \forall t, i, \quad \sum_{j \in \mathcal{N}(i)} f_{i,j}^t - \sum_{k : i \in \mathcal{N}(k)} f_{k,i}^{t-1} \leq 0 \\
& \sum_{j \in \mathcal{N}(\upsilon_{\text{source}})} f_{\upsilon_{\text{source}},j} - \sum_{k : \upsilon_{\text{sink}} \in \mathcal{N}(k)} f_{k,\upsilon_{\text{sink}}} \leq 0.
\end{aligned}
$$

[J. Berclaz et al., "Multiple Object Tracking Using K-Shortest Paths Optimization," PAMI 2011]

Multiple Object Tracking Using K-Shortest Paths Optimization

$$\mathbf{f}^* = \arg\min_{\mathbf{f}\in\mathcal{H}} \sum_{t,i} c\big(e_{i,j}^t\big) \sum_{j\in\mathcal{N}(i)} f_{i,j}^t$$



"compl. red." means complexity reduction i.e., pruned detections

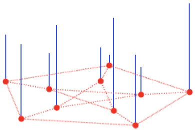
[J. Berclaz et al., "Multiple Object Tracking Using K-Shortest Paths Optimization," PAMI 2011]

Multiple Object Tracking Using K-Shortest Paths Optimization

[J. Berclaz et al., "Multiple Object Tracking Using K-Shortest Paths Optimization," PAMI 2011]

Multi-Target Tracking via Graph Neural Networks

Signal processing on graphs: definitions



We are interested in signals defined on an undirected, connected, weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, with vertices $\mathcal{V}$ (where $|\mathcal{V}| = N$), edges $\mathcal{E}$, and a weighted adjacency matrix $\mathbf{W}$. If vertices i and j are linked by an edge $e = (i, j)$, then $W_{i,j}$ represents the weight of the edge; otherwise, $W_{i,j} = 0$.

A signal or function $f : \mathcal{V} \to \mathbb{R}$ defined on the vertices of the graph may be represented as a vector $\mathbf{f}$ in $\mathbb{R}^N$, where the i^{th} component of the vector $\mathbf{f}$ represents the function value at the i^{th} vertex in $\mathcal{V}$.

[19] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98, 2013.

The non-normalized graph Laplacian

- The *non-normalized graph Laplacian*, also called the *combinatorial graph Laplacian*, is defined as $Ł := \mathbf{D} - \mathbf{W}$, where the degree matrix $\mathbf{D}$ is a diagonal matrix whose i^{th} diagonal element d_i is equal to the sum of the weights of all of its incident edges.
- The graph Laplacian is a difference operator: for any $\mathbf{f} \in \mathbb{R}^N$

$$(Ł\mathbf{f})(i) = \sum_{j \in \mathcal{N}_i} W_{i,j}[f(i) - f(j)],$$

where $\mathcal{N}_i$ is the set of vertices connected v_i by an edge.
- Since Ł is real and symmetric, it has a complete set of orthonormal eigenvectors, $\{\mathbf{u}_l\}_{l=0,\dots,N-1}$. These eigenvectors have associated real, non-negative eigenvalues $\{\lambda_l\}_{l=0,\dots,N-1}$ satisfying $Ł\mathbf{u}_l = \lambda\mathbf{u}_l$, for $l = 0, \dots, N-1$.

A graph Fourier transform

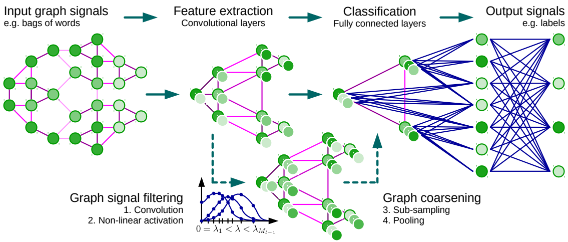
We can define the graph Fourier transform $\hat{\mathbf{f}}$ of any function $\mathbf{f} \in \mathbb{R}^N$ on the vertices of $\mathcal{G}$ as the expansion of $\mathbf{f}$ in terms of the eigenvectors of the graph Laplacian:

$$\hat{f}(\lambda_l) := \langle \mathbf{f}, \mathbf{u}_l \rangle = \sum_{i=1}^{\mathcal{N}} f(i) u_l^*(i).$$

The *inverse graph Fourier transform* is then given by

$$f(i) = \sum_{i=1}^{\mathcal{N}} \hat{f}(\lambda_l) u_l(i).$$

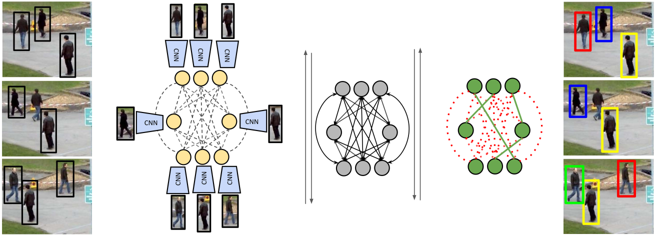
Modern GNNs: desiderata



- strictly localized filters
- permutation invariance
- low computational complexity

Figure credit: [6] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc.

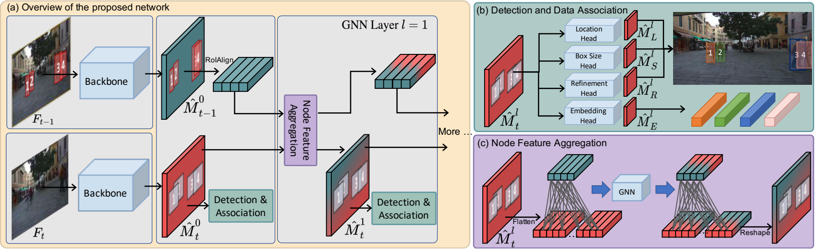
GNNs for Solving Network Flow (approximately)



Approximating the *network max-flow* algorithm with a GNN [2]

[2] G. Brasó and L. Leal-Taixé. Learning a neural solver for multiple object tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

GNNs for Joint Detect and Data Association



1. GNN nodes: possible detections at time t and tracklets from time $t-1$
2. GNN edges between possible detections and tracklets
3. possible detections are defined at *every* pixel of $\hat{M}^\ell_t$

[23] Y. Wang, K. Kitani, and X. Weng. Joint Object Detection and Multi-Object Tracking with Graph Neural Networks. arXiv:2006.13164, 2020.

Next Time: Guest Lecture

“Visual Object Tracking and Retrieval with Natural Language Description”
Guest Lecturer: Qi Feng, Boston University

[Q. Feng et al., “Real-time Visual Object Tracking with Natural Language Description,” WACV 2020]