Tracking with Correlation Filters

Lecture for AE4M33MVP

Acknowledgement to João F. Henriques from Institute of Systems and Robotics University of Coimbra for providing materials for this presentation
Lecture Overview

- Discriminative tracking
- Connection of correlation and the discriminative tracking
- Brief history of correlation filters
- Breakthrough by MOSSE tracker

- Why MOSSE works?
  (connection of correlation filters and machine learning)
  - Circulant matrices
  - Ridge Regression

- Kernelized Correlation Filters
Discriminative Tracking

$t=0$

+1 +1 +1 −1 −1 −1

t > 0

samples

labels

Classifier

Classify subwindows to find target

AE4M33MVP– Tracking with Correlation Filters

slides material by João F. Henriques
Discriminative Tracking

- How to get training samples for the classifier?
- Standard approach:
  - bboxes with high overlap with the GT $\rightarrow$ Pos. samples
  - bboxes far from the GT $\rightarrow$ Neg. samples

$t=0$

- What with the samples in the unspecified area?
Connection to Correlation

- Let’s have a linear classifier with weights $w$
  \[ y = w^T x \]

- During tracking we want to evaluate the classifier at subwindows $x_i$:
  \[ y_i = w^T x_i \]

- Then we can concatenate $y_i$ into a vector $y$ (i.e. response map)

- This is equivalent to **cross-correlation** formulation which can be computed **efficiently** in Fourier domain
  \[ y = x \ast w \]

- Note: Convolution is related; it is the same as cross-correlation, but with the flipped image of $w$ ($P \rightarrow d$).
The Convolution Theorem

“Cross-correlation is equivalent to an element-wise product in Fourier domain”

\[ y = x \odot w \iff \hat{y} = \hat{x}^* \times \hat{w} \]

where:
- \( \hat{y} = \mathcal{F}(v) \) is the Discrete Fourier Transform (DFT) of \( y \).
  (likewise for \( \hat{x} \) and \( \hat{w} \))
- \( \times \) 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).
Connection to Correlation

The Convolution Theorem

“Cross-correlation is equivalent to an element-wise product in Fourier domain”

\[ y = x \odot w \quad \iff \quad \hat{y} = \hat{x}^* \times \hat{w} \]

- In practice:

\[
\begin{align*}
    x & \rightarrow \mathcal{F} \rightarrow \hat{x}^* \\
    w & \rightarrow \mathcal{F} \rightarrow \hat{w}
\end{align*}
\]

\[
\begin{align*}
    \hat{x}^* \times \hat{w} \rightarrow \mathcal{F}^{-1} \rightarrow y
\end{align*}
\]

- 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 \( \mathcal{O}(n^2 \log n) \).
Connection to Correlation

The Convolution Theorem

“Cross-correlation is equivalent to an element-wise product in Fourier domain”

\[ y = x \odot w \quad \iff \quad \hat{y} = \hat{x}^* \times \hat{w} \]

■ Conclusion:

The evaluation of any linear classifier can be accelerated with the Convolution Theorem. (Not just for tracking.)

■ “linear” can become non-linear using kernel trick in some specific cases (will be discussed later)

■ Q: How the \( w \) for correlation should look like? What about training?
Connection to Correlation

- Q: How the \( \mathbf{w} \) for correlation should look like? What about training?

Objective

\[
\ast \quad \mathbf{w} \quad = \quad \text{High values} \\
\text{Unspecified} \\
\text{Low values}
\]

- Intuition of requirements of cross-correlation of classifier(filter) \( \mathbf{w} \) and a training image \( \mathbf{x} \)
  - A high peak near the true location of the target
  - Low values elsewhere (to minimize false positive)
Minimum Average Correlation Energy (MACE) filters, 1980’s

- Bring average correlation output towards 0:

  \[
  \min_w \| x \odot w \|^2
  \]

  except for target location, keep the peak value fixed:

  subject to: \( w^T x = 1 \)

- This produces a **sharp peak** at target location with closed form solution:

  \[
  \hat{w} = \frac{\hat{x}}{\hat{x}^* \times \hat{x}}
  \]

  - \( \hat{x}^* \times \hat{x} \) is called the **spectrum** and is real-valued.
  - division and product (\( \times \)) are element-wise.

- **Sharp peak = good localization!** Are we done?
Brief History of Correlation Filters

The MACE filter suffer from 2 main issues:

1. **Hard constraints** easily lead to overfitting.
   - **UMACE** (“Unconstrained MACE”) addresses this by removing the hard constraints and require to produce a high average correlation response on positive samples. However, it still suffer from the 2\textsuperscript{nd} problem.

2. **Enforcing a sharp peak** is too strong condition; lead to overfitting
   - **Gaussian-MACE / MSE-MACE** – peak to follow a 2D Gaussian shape
     \[
     \min_w \| x \odot w - g \|^2, \\
     \text{subject to: } w^T x = 1
     \]
   - In the original method (1990’s), the minimization was *still* subject to the MACE hard constraint.
     (*It later turned out to be unnecessary!*)
Brief History of Correlation Filters

Sharp vs. Gaussian peaks

Training image: \( x = \)

Naïve filter
\((w = x)\)

Classifier
\((w)\)

Output
\((w * x)\)

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

Sharp vs. Gaussian peaks

Training image: \( \mathbf{x} = \)

Naïve filter \((w = x)\)

Sharp peak \((\text{UMACE})\)

Classifier \((w)\)

Output \((w \ast x)\)

- A very sharp peak is obtained by emphasizing small image details (like the fish’s scales here).
- **generalizes poorly**: fine scale details that are usually not robust.
Brief History of Correlation Filters

Sharp vs. Gaussian peaks

Training image: \( \mathbf{x} = \)

Naïve filter \((\mathbf{w} = \mathbf{x})\)
Sharp peak (UMACE)
Gaussian peak (GMACE)

Classifier \((\mathbf{w})\)

Output \((\mathbf{w} \ast \mathbf{x})\)

- A good compromise.
- Tiny details are ignored.
- Focuses on larger, more robust structures.
Breakthrough by MOSSE tracker

Min. Output Sum of Sq. Errors (MOSSE)

- Presented by David Bolme and colleagues at CVPR 2010

- Tracker run at speed over a 600 frames per second

- very simple to implement
  - no complex features only raw pixel values
  - only FFT and element-wise operation

- performance similar to the most sophisticated tracker (at that time)
Breakthrough by MOSSE tracker

How does it work?

- Use only the “Gaussian peak” objective (no hard constraints)

\[
\min_w \|x \otimes w - g\|^2,
\]

- Found the following solution using the Convolution Theorem:

\[
\hat{w} = \frac{\hat{g} \times \hat{x}}{\hat{x}^* \times \hat{x} + \lambda}
\]

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

- No expensive matrix operations! \(\Rightarrow\) only FFT and element-wise op.
Breakthrough by MOSSE tracker

Implementation aspects

- Cosine (or sine) window preprocessing

  ![Image of cosine window preprocessing]

  - image edges smooth to zero
  - the filter sees an image as a “cyclic” (important for the FFT)
  - gives more importance to the target center.

- Simple update

  \[
  \hat{w}_{new} = \frac{\hat{g}^* \times \hat{x}}{\hat{x}^* \times \hat{x} + \lambda} \\
  \]

  \[
  \hat{w}_t = (1 - \eta)\hat{w}_{t-1} + \eta \hat{w}_{new} \\
  \]

  Train a MOSSE filter \( \hat{w}_{new} \) using the new image \( \hat{x} \).

  Update previous solution \( \hat{w}_{t-1} \) with \( \hat{w}_{new} \) by linear interpolation.
Breakthrough by MOSSE tracker

Implementation aspects

- Scale adaptation

- Extract patches with different scales and normalize them to the same size
- Run classification; use bounding box with the highest response
Why MOSSE works?

Circulant matrices

is a tool that connects correlation filters with machine learning

\[
\min_{\mathbf{w}} \| \mathbf{x} \odot \mathbf{w} - \mathbf{g} \|^2 \quad \text{replace correlation with a special matrix } C(\mathbf{x}) \quad \min_{\mathbf{w}} \| C(\mathbf{x}) \mathbf{w} - \mathbf{g} \|^2
\]

- \( C(\mathbf{x}) \) is a circulant matrix:
Why MOSSE works?

Circulant matrices

is a tool that connects **correlation filters** with **machine learning**

- We can see \( X = C(x) \) as a **dataset** with **cyclically shifted** versions of the image \( x \)

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

- \( P \) is a permutation matrix that shifts the pixels in vertical/horizontal direction by 1 element.
- Arbitrary shift \( i \) obtained with power \( P^i x \).
- Cyclic: \( P^n x = P^0 x = x \).
Why MOSSE works?

Circulant matrices

is a tool that connects correlation filters with machine learning

- Similar role to the Convolution Theorem

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

\[ \mathcal{F}(X) = \begin{bmatrix}
\hat{x}_1 & 0 & \cdots & 0 \\
0 & \hat{x}_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \hat{x}_n
\end{bmatrix} \]

- Data matrix is circulant

\[ \Rightarrow \]

- Becomes diagonal in Fourier domain

- Most of the “data” is 0 and can be ignored! ⇒ Massive speed-up
Why MOSSE works?

Ridge Regression Formulation

= Least-Squares with regularization (avoids overfitting!)

- Consider simple Ridge Regression (RR) problem:

\[
\min_w \|Xw - y\|^2 + \lambda \|w\|^2
\]

has closed-form solution: \( w = (X^TX + \lambda I)^{-1}X^Ty \)

We can replace \( X = C(x) \) (circulant data), and \( y = g \) (Gaussian targets).

- Diagonalizing the involved circulant matrices with the DFT yields:

\[
\hat{w} = \frac{\hat{x}^* \times \hat{y}}{\hat{x}^* \times \hat{x} + \lambda}
\]

\[\Rightarrow\]

- Exactly the MOSSE solution!

- good learning algorithm (RR) with lots of data (circulant/shifted samples).
Kernelized Correlation Filters

- Circulant matrices are a **very general tool** which allows to replace standard operations with fast Fourier operations.

- The same idea can by applied e.g. to the **Kernel Ridge Regression**:
  
  with $K$ kernel matrix $K_{ij} = \kappa(x_i, x_j)$ and dual space representation

  $$\alpha = (K + \lambda I)^{-1} y$$

- For many kernels, circulant data $\Rightarrow$ circulant $K$ matrix

  $$K = C(k^{xx})^{-1}$$
  
  where $k^{xx}$ is kernel auto-correlation and the first row of $K$ (small, and easy to compute)

- Diagonalizing with the DFT for learning the classifier yields:

  $$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \quad \Rightarrow \quad \text{Fast solution in } O(n \log n).$$
  
  Typical kernel algorithms are $O(n^2)$ or higher!
Kernelized Correlation Filters

- The $k^{xx'}$ is kernel correlation of two vectors $x$ and $x'$

$$k_{i}^{xx'} = \kappa(x', P^{i-1}x)$$

- For Gaussian kernel it yields:

$$k^{xx'} = \exp\left(-\frac{1}{\sigma^2}(\|x\|^2 + \|x'\|^2 - 2\mathcal{F}^{-1}(\hat{x}^* \odot \hat{x}'))\right)$$

- Evaluation on subwindows of image $z$ with classifier $\alpha$ and model $x$:
  1. $K^z = C(k^{xz})$
  2. $f(z) = \mathcal{F}^{-1}(\hat{k}^{xz} \odot \hat{\alpha})$

- Update classifier $\alpha$ and model $x$ by linear interpolation from the location of maximum response $f(z)$

- Kernel allows integration of more complex and multi-channel features
Kernelized Correlation Filters

KCF Tracker

- very few hyperparameters
- code fits on one slide of the presentation!
- Use HoG features (32 channels)
- ~300 FPS
- Open-Source (Matlab/Python/Java/C)

Training and detection (Matlab)

```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
```

Sum over channel dimension in kernel computation
Variations of KCF trackers

Basic
- Henriques et al. – CSK
  - raw grayscale pixel values as features
- Henriques et al. – KCF
  - HoG multi-channel features

Further work
- Danelljan et al. – DSST:
  - PCA-HoG + grayscale pixels features
  - filters for translation and for scale (in the scale-space pyramid)
- Li et al. – SAMF:
  - HoG, color-naming and grayscale pixels features
  - quantize scale space and normalize each scale to one size by bilinear inter. → only one filter on normalized size
Variations of KCF trackers

Further work

- Danelljan et al. – SRDCF:
  - spatial regularization in the learning process
    - limits boundary effect
    - penalize filter coefficients depending on their spatial location
  - allows to use much larger search region
  - more discriminative to background (more training data)

CNN-based Correlation Trackers

- Ma et al.
  - features: VGG-Net pretrained on ImageNet dataset extracted from third, fourth and fifth convolution layer
  - for each feature learn a linear correlation filter
  - coarse-to-fine approach from 5→3 layer

- Nam et al. – MDNet:
  - CNN classification (3 convolution layers and 2 fully connected layers)
  - learn on tracking sequences with bbox regression
Results of KCF-based trackers

Result on recent standard evaluation benchmarks

<table>
<thead>
<tr>
<th>Tracker</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDNet*</td>
<td>CNN learned on video sequences</td>
</tr>
<tr>
<td>DeepSRDCF</td>
<td>Corr. Filter + CNN feats</td>
</tr>
<tr>
<td>EBT</td>
<td>Edgebox features+SSVM+color hist.</td>
</tr>
<tr>
<td>SRDCF</td>
<td>Corr. Filter + color names + HoG</td>
</tr>
<tr>
<td>LDP</td>
<td>Part-based Corr. Filter</td>
</tr>
<tr>
<td>sPST</td>
<td>Flow + Edgebox feats + SVM</td>
</tr>
</tbody>
</table>