Discriminative Correlation Filters for Visual Tracking

Martin Danelljan
Overview – Part I

Part I: Basics of Discriminative Correlation Filters

1. The Visual Tracking problem
2. DCF – the simple case
3. Multi-channel, multi-sample DCF
4. Special cases and approximative inference
5. Tracking pipeline and practical considerations
6. Kernels
7. Scale estimation
8. Periodic assumption: problem and solutions
Overview – Part II

Part II: Advanced topics in DCF tracking

1. Training set management
2. Deep image representations for tracking
3. Continuous-space formulation
4. Efficient Convolution Operators (ECO)
5. End-to-end Learning with DCF
6. Empowering deep features
Visual Tracking
Visual Tracking
Visual Tracking

- Only initial target location in known
- Challenges
  - **Environmental**: occlusions, blur, clutter, illumination
  - **Motion/transformations**: rotations, fast motion, scale change
  - **Appearance changes**: deformations
Applications

Robotics, AR/VR, autonomous driving, video analysis ...
Discriminative Correlation Filters (DCF) - The Basics
Discriminative Correlation Filters

What is it?
• Discriminatively learn a correlation filter
• Utilize the Fourier transform for efficiency

Why use it?
• Translation invariance ⇒ Correlation
• State-of-the-art since 2014
• Accuracy (even sub-pixel)
• Generic and customizable
DCF Popularity and Performance

• **Hundreds** of papers since 2014

• **Winner** of Visual Object Tracking (VOT) Challenge 2014, 2016, 2017 and 2018

• In **VOT 2018**: all top-5 trackers are based on DCF
DCF – the Simple Case

DFT: \( \hat{x}[k_1, k_2] = \sum_{n_1, n_2} x[n_1, n_2] e^{-i2\pi \left( \frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right)} \)

\[ \rightarrow x \ast f = y \]

\[ \hat{x} \cdot \hat{f} = \hat{y} \]
DCF – the Simple Case

\[ \hat{f} = \frac{\hat{y}}{\hat{x}} \quad \rightarrow \quad f = \mathcal{F}^{-1} \left\{ \frac{\hat{y}}{\hat{x}} \right\} \]

\[ \hat{x} \cdot \hat{f} = \hat{y} \]
DCF – the Simple Case

\[ s := z \ast f = \mathcal{F}^{-1} \{ \hat{z} \cdot \hat{f} \} \]

Target prediction: \((n_1^*, n_2^*) = \arg \max s[n_1, n_2]\)
Standard DCF Formulation

1. Multiple training samples
   \[ \{(x_j, y_j)\}_{j=1}^m \]

2. Multidimensional sophisticated features
   \[ x_j[n] \in \mathbb{R}^D \]
Standard DCF Formulation

\[ S_f \{x\} = \sum_{d=1}^{D} f^d \ast x^d \]
Standard DCF Formulation

\[ L(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{ x_j \} - y_j \|^2 + \lambda \sum_{d=1}^{D} \| f^d \|^2 \]
Inference

- DFT and Parseval’s theorem:

\[
L(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{x}_j^d \hat{f}^d - \hat{y}_j \right\|^2 + \lambda \sum_{d=1}^{D} \left\| \hat{f}^d \right\|^2
\]

\[
L(f) = \sum_{j=1}^{m} \alpha_j \left\| \begin{pmatrix} \hat{x}_j^0 \end{pmatrix}^T \hat{f}^0 \begin{pmatrix} \hat{\hat{y}}_j^0 \end{pmatrix} - \begin{pmatrix} \hat{\hat{y}}_j^0 \end{pmatrix} \right\|^2 + \lambda \sum_{d=1}^{D} \left\| \hat{f}^d \right\|^2
\]
Inference

\[ A[k] = \begin{pmatrix} \hat{x}_1[k]^T \\ \vdots \\ \hat{x}_m[k]^T \end{pmatrix}, \quad \hat{y}[k] = \begin{pmatrix} \hat{y}_1[k] \\ \vdots \\ \hat{y}_m[k] \end{pmatrix}, \quad B = \begin{pmatrix} \sqrt{\alpha_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sqrt{\alpha_m} \end{pmatrix} \]

\[
L(f) = \left\| \begin{pmatrix} BA[0]\hat{f}[0] \\ \vdots \\ BA[K]\hat{f}[K] \end{pmatrix} - \begin{pmatrix} B\hat{y}[0] \\ \vdots \\ B\hat{y}[K] \end{pmatrix} \right\|^2 + \lambda \sum_{d=1}^{D} \left\| \hat{f}^d \right\|^2
\]
Inference

\[ A = \begin{pmatrix} A[0] & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A[K] \end{pmatrix}, \quad \hat{f} = \begin{pmatrix} \hat{f}[0] \\ \vdots \\ \hat{f}[K] \end{pmatrix}, \quad \Gamma = I \otimes B^2 \]

\[ L(f) = \left\| \Gamma^{\frac{1}{2}} A\hat{f} - \Gamma^{\frac{1}{2}} \hat{y} \right\|^2 + \lambda \left\| \hat{f} \right\|^2 \]
Inference

$$\left( A^H \Gamma A + \lambda I \right) \hat{f} = A^H \Gamma \hat{y}$$

$$\nabla L = 0$$

$$L(f) = \left\| \Gamma^{\frac{1}{2}} A \hat{f} - \Gamma^{\frac{1}{2}} \hat{y} \right\|^2 + \lambda \left\| \hat{f} \right\|^2$$
Inference

$$(A^H \Gamma A + \lambda I) \hat{f} = A^H \Gamma \hat{y}$$
Special Case 1: $D = 1$

Only a single **feature channel**:

$$
\hat{f} = \frac{\sum_{j=1}^{m} \alpha_j \hat{x}_j \hat{y}_j}{\sum_{j=1}^{m} \alpha_j \hat{x}_j \hat{x}_j + \lambda}
$$

The original MOSSE filter [Bolme et al., CVPR 2010].
Dual form

\[(A^H\Gamma A + \lambda I)\hat{f} = A^H\Gamma \hat{y}\]

\[\hat{f} = A^H\Gamma (AA^H\Gamma + \lambda I)^{-1}\hat{y}\]

Blocks of \(m \times m\)
Special Case 2: $m = 1$

Only a single **training sample**:

$$
\hat{f}_d = \frac{\bar{x}_d \hat{y}}{\sum_{l=1}^{D} \bar{x}_l \bar{x}_l + \lambda}
$$

[Danelljan et al., BMVC 2014, PAMI 2017]
Approximate inference

1. **Independent samples:**
   - Optimal for $m = 1$

   $$
   
   f^d = \sum_{j=1}^{m} \alpha_j \frac{\hat{x}_{j}^d \hat{y}_j}{\sum_{l=1}^{D} \hat{x}_{j}^l \hat{x}_{j}^l + \lambda}
   $$

2. **Independent channels:**
   - Optimal for $D = 1$

   $$
   f^d = \frac{1}{D} \frac{\sum_{j=1}^{m} \alpha_j \hat{x}_{j}^d \hat{y}_j}{\sum_{j=1}^{m} \alpha_j \hat{x}_{j}^d \hat{x}_{j}^d + \lambda}
   $$

3. **Combination:**
   - Optimal for $m = 1$
   - Optimal for $D = 1$

   $$
   f^d = \frac{\sum_{j=1}^{m} \alpha_j \hat{x}_{j}^d \hat{y}_j}{\sum_{j=1}^{m} \alpha_j \sum_{l=1}^{D} \hat{x}_{j}^l \hat{x}_{j}^l + \lambda}
   $$
General tracking pipeline

1. **Initialize** model in first frame

2. **Track** in the new frame

3. **Update** model and goto 2.
Tracking pipeline: example

1. **Initialize**
   1: \[ \hat{f}^d_{\text{num}} \leftarrow \hat{x}_1^d \hat{y}_1 \]
   2: \[ \hat{f}^d_{\text{den}} \leftarrow \sum_{l=1}^{D} \hat{x}_1^l \hat{x}_1^l \]
   3: \[ \hat{f}^d \leftarrow \frac{\hat{f}^d_{\text{num}}}{\hat{f}^d_{\text{den}} + \lambda} \]

2. **Track**
   1: \[ s \leftarrow \mathcal{F}^{-1}\{\sum_{d=1}^{D} \hat{f}^d \hat{z}^d\} \]
   2: \[(n_1^*, n_2^*) \leftarrow \arg \max_s s[n_1, n_2]\]

3. **Update**
   1: \[ \hat{f}^d_{\text{num}} \leftarrow (1 - \gamma)\hat{f}^d_{\text{num}} + \gamma \hat{x}_j^d \hat{y}_j \]
   2: \[ \hat{f}^d_{\text{den}} \leftarrow (1 - \gamma)\hat{f}^d_{\text{den}} + \gamma \sum_{l=1}^{D} \hat{x}_j^l \hat{x}_j^l \]
   3: \[ \hat{f}^d \leftarrow \frac{\hat{f}^d_{\text{num}}}{\hat{f}^d_{\text{den}} + \lambda} \]

Target location

Learning rate
Practical considerations

1. Multiply samples with **cosine window**

   - Reduces boundary effects
Practical considerations

2. For $y_j$: use **Gaussian function**

\[ y_j [n_1, n_2] = e^{-\frac{1}{2\sigma^2} ||n-n^{*,j}||^2} \]

- Centered at target location $n^{*,j} = (n_1^{*,j}, n_2^{*,j})$
- Peak width parameter $\sigma$
- Motivation: minimizes the uncertainty principle
Kernelized Correlation Filters
Kernelized Correlation Filters (KCF)

- Henriques et al. [ECCV 2012, PAMI 2014]
- Idea: apply the **kernel trick** to the DCF

**Kernel:** \[ \kappa(x, z) = \langle \phi(x), \phi(z) \rangle \]

**Shift invariant:** \[ \kappa(\tau_n x, z) = \kappa(x, \tau_{-n} z) \]

**Shift operator:** \[ \tau_n x[m] = x[m - n] \]

**Example:** \[ \kappa(x, z) = e^{-\frac{1}{2\eta} \| x - z \|^2} \]
Kernelized Correlation Filters (KCF)

- Kernelized correlation:  \( k_{x,z}[n] = \kappa(\tau_n x, z) \)

- Train model: \( \hat{u} = \frac{\hat{y}}{\hat{k}_{x,x} + \lambda} \)

- Target scores: \( s = \mathcal{F}^{-1}\{\hat{k}_{z,x} \hat{u}\} \)

- Approximative update rules [Henriques et al., 2012; Danelljan et al., 2014].
Kernelized Correlation Filters (KCF)

Should you use kernels?

😊 More complicated learning
😊 Harder to generalize
😊 More costly
😊 Similar or poorer performance

Essence of deep learning:
- Learn your feature mapping instead
Scale Estimation

Scale Estimation
Approach 1: Multi-scale detection

1. Extract test samples at multiple scales \( \{z_1, z_2, \ldots, z_L\} \)

2. Compute scores at each scale \( s_l = S_f\{z_l\} \)

3. Find max position and scale

\[
(n_1^*, n_2^*, l^*) = \arg \max_{n_1, n_2, l} s_l[n_1, n_2]
\]
Approach 2: Scale filter

- **Idea:** train a separate 1-dimensional scale DCF
- Directly discriminates between scales
- Discriminative Scale Space Tracker (DSST) [Danelljan et al., BMVC 2014, PAMI 2017]
Discriminative Scale Space Tracker

Scale training sample $x$

$\rightarrow \quad x(1)$

$\rightarrow \quad x(2)$

$\ldots$

$\rightarrow \quad x(N)$

$x^1 \quad x^2 \quad \ldots \quad x^D$

Desired output $y$

Target size

Confidence score
Discriminative Scale Space Tracker
Discriminative Scale Space Tracker
Evaluation Measures

\[ \text{IoU}_j = \frac{\text{Area}(B^P_j \cap B^G_j)}{\text{Area}(B^P_j \cup B^G_j)} \]
Scale Estimation Results

Success plot of OPE

\[ \text{OP}(T) = \frac{\sum_j [\text{IoU}_j > T]}{\sum_j 1} \]

\[ \text{AUC} = \int_0^1 \text{OP}(T) \, dT \]

OTB-2013 dataset
[Wu et al., CVPR 2013]
Scale estimation: Comparison

Approach 2 (scale filter):
• Faster
• Generic (used in many different trackers)
• Often more accurate for simple DCF trackers

Approach 1 (multi-scale detection):
• Slower
• Often more accurate for advanced DCF trackers presented next
The Periodic Assumption: Problem and Solutions
Periodic Assumption in DCF

What we want...  What actually happens...

[Images of what we want and what actually happens]
Larger Samples?
Why?

Learned filter $f^d$
Effects of Periodic Assumption

Forces a \textbf{small sample size} in training/detection

Effects:
\begin{itemize}
  \item Limits training data
  \item Corrupts data
  \item Limits search region
\end{itemize}
Tackling the Periodic Assumption

We need means of controlling the **filter extent**!

- Enables **larger samples**.

Approaches:

1. Constrained optimization
2. Spatial regularization
Constrained Optimization

• **Idea:** Constrain filter coefficients to be zero outside the target bounding box.

\[
\begin{align*}
\min & \quad L(f) \\
\text{subject to} & \quad f^d[n] = 0, \forall d, \forall n \in B
\end{align*}
\]

• Rewrite constraint:

\[
\begin{align*}
f^d[n] = 0, \forall n \in B & \iff 1_B f^d = 0 \\
& \iff 1_B \mathcal{F}^{-1} \hat{f}^d = 0
\end{align*}
\]

In reverse Fourier transform
Constrained Optimization

- Fourier domain formulation:

\[
\begin{align*}
    \min & \quad L(f) \\
    \text{subject to} & \quad 1_B \mathcal{F}^{-1} \hat{f}^d = 0, \forall d
\end{align*}
\]

\[
L(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{x}_j^d \mathcal{F} \mathbf{1}_T \mathcal{F}^{-1} \hat{f}^d - \hat{y}_j \right\|^2 + \lambda \sum_{d=1}^{D} \left\| \hat{f}^d \right\|^2
\]

target pixels
Constrained Optimization

\[ L(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{x}_j \mathcal{F} \mathbf{1}_T \mathcal{F}^{-1} \hat{f}^d - \hat{y}_j \right\|^2 + \lambda \sum_{d=1}^{D} \left\| \hat{f}^d \right\|^2 \]

- Generates **dense** normal equations 😞
- Use iterative solvers:
  - ADMM [H.K. Galoogahi, CVPR 2015]
  - Proximal gradient [J.A. Fernandez, PAMI 2015]
- Requires iterating between spatial and Fourier 😞
Spatially Regularized DCF (SRDCF)

[M. Danelljan, ICCV 2015]

\[
L(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{ x_j \} - y_j \|^2 + \lambda \sum_{d=1}^{D} \| f^d \|^2
\]

\[
L(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{ x_j \} - y_j \|^2 + \sum_{d=1}^{D} \| \omega f^d \|^2
\]
Spatially Regularized DCF (SRDCF)
[M. Danelljan, ICCV 2015]

\[
L(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{ x_j \} - y_j \|^2 + \sum_{d=1}^{D} \| w f^d \|^2
\]

Prior: \( f^d[n] \sim \mathcal{N} \left( 0, \frac{1}{w^2[n]} \right) \)
Spatially Regularized DCF (SRDCF)

\[ L(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} x_j^d \ast f^d - y_j \right\|^2 + \sum_{d=1}^{D} \left\| \omega f^d \right\|^2 \]

DFT

\[ L(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{x}_j^d \hat{f}^d - \hat{y}_j \right\|^2 + \sum_{d=1}^{D} \left\| \frac{\hat{\omega}}{N_1 N_2} \ast \hat{f}^d \right\|^2 \]
Spatially Regularized DCF (SRDCF)

\[ \mathcal{W} \quad \xrightarrow{\text{DFT}} \quad \hat{\mathcal{W}} \]
Spatially Regularized DCF (SRDCF)

\[ L(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{x}_j^d \hat{f}^d - \hat{y}_j \right\|^2 + \sum_{d=1}^{D} \left\| \frac{\hat{w}}{N_1 N_2} \ast \hat{f}^d \right\|^2 \]

\[ (A^H \Gamma A + W^H W) \hat{f} = A^H \Gamma \hat{y} \]

Convolution matrix
Spatially Regularized DCF (SRDCF)

What we had...

What we achieved...
Spatially Regularized DCF
Spatially Regularized DCF
Spatially Regularized DCF

OTB-2015 dataset

Success plot of OPE

Overlap Precision [%]

Overlap threshold

- Ours CN [63.9]
- Ours HOG [63.0]
- Small Period HOG [57.8]
- Large Period CN [48.8]
- Small Period CN [48.4]
- Large Period HOG [42.3]
Adaptive Training Set Management

Model Drift
Adaptive Training Set Management

\[ L(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{x_j\} - y_j \|^2 + \sum_{d=1}^{D} \| \omega f^d \|^2 \]
Discriminative Tracking Methods

\[ J(\theta) = \sum_{k=1}^{t} \alpha_k \sum_{j=1}^{n_k} L(\theta; x_{jk}, y_{jk}) + \lambda R(\theta) \]
Our Approach - Motivation

• **Continuous** weights
  – More control of importance
  – Helps in ambiguous cases (e.g. partial occlusions)

• Re-determination of importance in **each frame**
  – Exploit **later** samples
  – Use all available information

• **Prior** information
  – E.g. how old the sample is
  – Or number of samples in a frame
Our Approach

\[
J(\theta, \alpha) = \sum_{k=1}^{t} \alpha_k \sum_{j=1}^{n_k} L(\theta; x_{jk}, y_{jk}) + \frac{1}{\mu} \sum_{k=1}^{t} \frac{\alpha_k^2}{\rho_k} + \lambda R(\theta)
\]

subject to \( \alpha_k \geq 0 \), \( k = 1, \ldots, t \)

\[
\sum_{k=1}^{t} \alpha_k = 1.
\]
Adaptive Sample Weights
Deep Image Representations
For Tracking
Hand-crafted Features

Color Features
[M. Danelljan, CVPR 2014]

Color Names
[Weijer and Schmid, TIP 2009]

Shape features

Histogram of Oriented Gradients (HOG)
[Dalal and Triggs, 2005]
Deep Convolutional Features
Evaluation of Convolutional Feature Layers

• On OTB-2013 dataset

[Overlapping Precision (%)]

Layer 0  Layer 1  Layer 2  Layer 3  Layer 4  Layer 5

[M. Danelljan, ICCVW 2015]
Learning Continuous Convolution Operators

Discriminative Correlation Filters (DCF)

Limitations:
- Single-resolution feature map
- Coarse output scores
Our Approach: Overview

Multi-resolution features

Continuous filters

Continuous output
DCF Limitations:

1. Single-resolution feature map

• Why a problem?
  – Combine convolutional layers of a CNN
    • Shallow layers: low invariance – high resolution
    • Deep layers: high invariance – low resolution

• How to solve?
  – Explicit resampling?
    • Artefacts, information loss, redundant data
  – Independent DCFs with late fusion?
    • Sub-optimal, correlations between layers
DCF Limitations:

2. Coarse output scores

• Why a problem?
  – Accurate localization
    • Sub-grid (e.g. HOG grid) or sub-pixel accuracy
    • More accurate annotations=> less drift

• How to solve?
  – Interpolation?
    • Which interpolation strategy?
  – Interweaving?
    • Costly
DCF Limitations:

3. Coarse labels

• Why a problem?
  – Accurate learning
    • Sub-grid or sub-pixel supervision

• How to solve?
  – Interweaving?
    • Costly
  – Explicit interpolation of features?
    • Artefacts
Interpolation Operator

\[
J_d \{ x^d \} (t) = \sum_{n=0}^{N_d-1} x^d[n] b_d \left( t - \frac{T}{N_d} n \right)
\]
Convolution Operator

\[ S_f \{x\} = \sum_{d=1}^{D} f^d * J_d \{x^d\} \]

\[ g * h(t) = \frac{1}{T} \int_0^T g(t - s)h(s) \, ds \]
Training Loss

\[ E(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{ x_j \} - y_j \|^2 + \sum_{d=1}^{D} \| w f^d \|^2 \]

\[ \| g \|^2 = \frac{1}{T} \int_{0}^{T} |g(t)|^2 \, dt \]
Training Loss – Fourier Domain

\[ E(f) = \sum_{j=1}^{m} \alpha_j \left\| \sum_{d=1}^{D} \hat{f}^d X_j^d \hat{b}_d - \hat{y}_j \right\|_{\ell^2}^2 + \sum_{d=1}^{D} \left\| \hat{w} * \hat{f}^d \right\|_{\ell^2}^2 \]

\[ \downarrow \]

\[ (A^H \Gamma A + W^H W) \hat{f} = A^H \Gamma \hat{y} \]

\[ \hat{g}[k] = \langle g, e_k \rangle = \frac{1}{T} \int_0^T g(t) e^{-i \frac{2\pi}{T} k t} \, dt \quad X^d[k] = \sum_{n=0}^{N_d-1} x^d[n] e^{-i \frac{2\pi}{N_d} n k} \]
Optimization: Conjugate Gradient

• Solve \((A^H \Gamma A + W^H W) \hat{f} = A^H \Gamma \hat{y}\)

• **Use Conjugate Gradient:**
  – Only need to evaluate \((A^H \Gamma A + W^H W) \hat{f}\)
  – \(\Rightarrow\) No sparse matrix handling!
  – **Warm start** estimate and search direction
  – **Preconditioner** important

How to set $y_j$ and $b_d$?

• Use periodic summation of functions $g : \mathbb{R} \to \mathbb{R}$:

$$g_T(t) = \sum_{n=-\infty}^{\infty} g(t - nT)$$

• Gaussian function for $y_j$
• Cubic spline kernel for $b_d$
• Fourier coefficients $\hat{y}_j, \hat{b}_d$ with Poisson’s summation formula:

$$\hat{g}_T[k] = \frac{1}{T} \hat{g} \left( \frac{k}{T} \right)$$
Results

• Layer fusion on OTB-2015 dataset
Sub-pixel Localization with CCOT

\[ S_f \{ \} = \]
Sub-pixel Localization with CCOT
Feature Point Tracking Framework

- Grayscale pixel features, $D = 1$
- Uniform regularization, $w(t) = \beta$

\[
\hat{f}[k] = \frac{\sum_{j=1}^{m} \alpha_j X_j[k] \hat{b}[k] \hat{y}_j[k]}{\sum_{j=1}^{m} \alpha_j |X_j[k] \hat{b}[k]|^2 + \beta^2}
\]
CCOT Feature Point Tracking
Experiments: Feature Point Tracking

• The Sintel dataset

Precision plot

Fraction of endpoints below threshold

Endpoint error threshold [pixels]

- Ours (0.886)
- Ours-FF (0.871)
- MOSSE (0.879)
- KLT (0.773)
Efficient Convolution Operators (ECO)

Issues With C-COT

1. Slow
   - ~10 FPS with hand-crafted features
   - ~1 FPS with deep features

2. Overfitting
   - ~0.5M parameters updated online
   - Memory focusing on recent samples
Factorized Convolution

\[ S_{Pf}\{x\} = \sum_{c,d} p_{d,c} f_c \ast J_d\{x^d\} = f \ast P^T J\{x\} \]

- Learn filter \( f \) and matrix \( P \) \textbf{jointly}
- Gauss Newton iterations with Conjugate Gradient
- \textbf{80\%} reduction in parameters
Factorized Convolution

C-COT filters

ECO filters
Generative Sample Space Model

- Online Gaussian Mixture Model of training samples
- $\Rightarrow$ 90\% reduction in training samples

ECO: GMM clusters

Previous: Linear memory
Speedup

• 10x speedup compared to C-COT
• Same or better performance

• 60 FPS on CPU with handcrafted features
• 15 FPS on GPU with deep features

Notes:
• Matlab/Mex
• “Slow” network
End-to-end Learning with DCF
End-to-end Learning

• Could we learn the underlying features?

• Use the DCF solution for a single training sample as a layer in a deep network:

\[
\hat{f}_\theta^d = \frac{\hat{x}_\theta^d \hat{y}}{\sum_{l=1}^{D} \hat{x}_\theta^l \hat{x}_\theta^l + \lambda}
\]

Network parameters

• Train in Siamese fashion: \( \ell(f_\theta \ast z_\theta, c) \)
  – On image pairs

Test sample

Desired output
End-to-end Learning: CFNet

[J. Valmadre et al., CVPR 2017]

- Logistic loss
End-to-end Learning: CFCF

[E. Gondogdu and A. Alatan, TIP 2018]

- $L^2$-loss. Finetune VGG-m.
- Integrate learned features in C-COT
Unveiling the Power of Deep Tracking

ECO

Tracking Performance, NFS

AUC in %

Hand-crafted + VGG-M
Motivation

• Challenges: Deformations, In-plane/Out-of-plane rotations

• Can we utilize the invariance of deep features?
Motivation

• How about using deeper networks?
Motivation

• Features unsuitable for tracking?
  – Let's train features for tracking
Causes 1: Training data

- Limited training data in the first frame
- Training data only models translations
Data augmentation

• Can simulate commonly encountered challenges in object tracking, e.g. rotations, motion blur, occlusions
Data augmentation

Impact of data augmentation, OTB-2015

Deep: ResNet-50 (Conv4)
Shallow: HOG+Color Names
Cause 2: Accuracy-Robustness Tradeoff

Image

Shallow Model

Deep Model
Cause 2: Accuracy-Robustness Tradeoff

Let’s revisit training in ECO

• Training data: Shifted versions of the target
• Width of label function determines how the samples are labelled
• Sharp label function ⇒ Enforce Accuracy
• Wide label function ⇒ Prefer Robustness
Cause 2: Accuracy-Robustness Tradeoff

Impact of label width, OTB-2015

Deep: ResNet-50 (Conv4)

Shallow: HOG+Color Names
Accuracy–Robustness tradeoff

Tracking Performance, NFS

<table>
<thead>
<tr>
<th>Method</th>
<th>AUC in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-crafted</td>
<td>40</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>45</td>
</tr>
<tr>
<td>ResNet-50+σ</td>
<td>50</td>
</tr>
<tr>
<td>ResNet-50+σ+Aug</td>
<td>55</td>
</tr>
</tbody>
</table>
Accuracy-Robustness tradeoff

Image

Shallow Model

Deep Model
New framework

Extract Features

Shallow

Deep

Train Separate Filters

Apply filter

Fusion?
Adaptive Model Fusion

We want the score function to have a single, sharp peak
Adaptive Model Fusion

\[ y_\beta(t) = \beta_d y_d(t) + \beta_s y_s(t) \]

- Prediction Quality Measure

\[ \xi_{t^*}\{y\} = \min_t \frac{y(t^*) - y(t)}{\Delta(t - t^*)} \quad \Delta(\tau) = 1 - e^{-\frac{\kappa}{2} |\tau|^2} \]
Results

Need For Speed dataset (100 videos)
Results

Generalization to networks

![Diagram showing AUC comparison between ECO and Ours for different networks: HOG-CN, VGG-M, GoogLeNet, and ResNet-50. The diagram indicates improvements in AUC for each network: +4.4%, +6.2%, +8.4%, and +8.4% respectively.]
State-of-the-Art and Conclusions
Current state-of-the-art

- VOT2018 sequestered dataset

<table>
<thead>
<tr>
<th>Tracker</th>
<th>EAO</th>
<th>A</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. MFT</td>
<td>0.2518</td>
<td>0.5768</td>
<td>0.3105</td>
</tr>
<tr>
<td>2. UPDT</td>
<td>0.2469</td>
<td>0.6033</td>
<td>0.3427</td>
</tr>
<tr>
<td>3. RCO</td>
<td>0.2457</td>
<td>0.5707</td>
<td>0.3154</td>
</tr>
<tr>
<td>4. LADCF</td>
<td>0.2218</td>
<td>0.5499</td>
<td>0.3746</td>
</tr>
<tr>
<td>5. DeepSTRCF</td>
<td>0.2205</td>
<td>0.5998</td>
<td>0.4435</td>
</tr>
<tr>
<td>6. CPT</td>
<td>0.2087</td>
<td>0.5773</td>
<td>0.4238</td>
</tr>
<tr>
<td>7. SiamRPN</td>
<td>0.2054</td>
<td>0.6277</td>
<td>0.5175</td>
</tr>
<tr>
<td>8. DLSTpp</td>
<td>0.1961</td>
<td>0.5833</td>
<td>0.4544</td>
</tr>
</tbody>
</table>

[“The Visual Object Tracking VOT2018 Challenge Results”, M. Kristan et al., 2018]
Conclusions and Future Work

• DCF is a **versatile** framework for tracking
• Highly adaptable for specific applications
• Efficient **online learning**

• Future work:
  – Richer output: towards **segmentation**
  – Long-term tracking robustness
  – Better **end-to-end** integration and learning
Acknowlegements

- EMC2, funded by Vetenskapsrådet
- Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation
Martin Danelljan

www.liu.se