EnKCF : An Ensemble of Kernelized Correlation Filters for High Speed Object Tracking

Burak Uzkent and YoungWoo Seo

Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology, NY
Motivation

• The goal of this work is to develop an online and single-target tracking algorithm that can run at a typical embedded system at real-time in $>30$ fps.

Trackers are called real-time when operating at $>30$fps on powerful machines (i5, i7)

Run-time performance drops dramatically on low-cost embedded-systems (10-20 times less speed)

Correlation Filter Trackers

KCF, DCF, CSK, MOSSE

No GPUs! CPU only!

$>300$ fps

$>30$ fps
Kernelized Correlation Filter Tracking

1. Ridge Regression $\rightarrow \min_w \sum_i (f(x_i) - y_i)^2 + \lambda \|w\|^2$

2. Analytical Solution $\rightarrow w = (X^T X + \lambda I)^{-1} X^T y$ expensive!!! $O(n^3)$

3. Circulant Matrix $\rightarrow X = F \text{diag}(\hat{x}) F^H$

4. Solution in Frequency Domain (Primal) $\rightarrow \hat{w} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}$ requires $O(n \log(n))$

5. Solution in Dual Domain (Training) $\rightarrow \hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}$

6. Detection $\rightarrow \hat{f}(z) = \hat{k}^{xz} \odot \hat{\alpha}$ runs at >300 fps!!! not scale-adaptive

EnKCF (Scale Adaptive Tracking at >300 fps)

The proposed EnKCF Framework with Particle Filter
Object Representation

Large ROI Translation Filter

Reshape (96xn)

fHoG

Color-naming

Desired Response

Small ROI Translation Filter

Reshape (96xn)

Target-only Scale Filter

Reshape (64xn)

Resize (96xn)
Results on **UAV123** Dataset

Some results on the UAV123 dataset highlighting EnKCF’s scale adaptiveness capability.

### >300fps Trackers

<table>
<thead>
<tr>
<th></th>
<th>EnKCF</th>
<th>KCF</th>
<th>DCF</th>
<th>CSK</th>
<th>MOSSE</th>
<th>STC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision (20 px, %)</td>
<td>54.5</td>
<td>52.3</td>
<td>52.6</td>
<td>48.7</td>
<td>46.6</td>
<td>50.7</td>
</tr>
<tr>
<td>Success Rate (AUC, %)</td>
<td>40.2</td>
<td>33.6</td>
<td>33.7</td>
<td>31.4</td>
<td>30.1</td>
<td>32.9</td>
</tr>
<tr>
<td>FPS</td>
<td>416</td>
<td>296</td>
<td>457</td>
<td>400</td>
<td>512</td>
<td>340</td>
</tr>
</tbody>
</table>

### <50fps Trackers

<table>
<thead>
<tr>
<th></th>
<th>EnKCF</th>
<th>ECO</th>
<th>CCOT</th>
<th>SAMF</th>
<th>MUSTER</th>
<th>DSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision (20 px, %)</td>
<td>54.5</td>
<td>61.6</td>
<td>63.3</td>
<td>59.2</td>
<td>59.3</td>
<td>58.6</td>
</tr>
<tr>
<td>Success Rate (AUC, %)</td>
<td>40.2</td>
<td>49.1</td>
<td>49.8</td>
<td>40.3</td>
<td>39.9</td>
<td>36.1</td>
</tr>
<tr>
<td>FPS</td>
<td>416</td>
<td>53</td>
<td>12</td>
<td>5</td>
<td>1</td>
<td>35</td>
</tr>
</tbody>
</table>

**Embedded systems compatible** 😊

**Computation-intensive** 😞
Results on the **UAV123_10fps** dataset

- State-of-the-art trackers (<50fps) is likely to run on low-cost embedded system at <10fps.

<table>
<thead>
<tr>
<th>&lt;50fps Trackers</th>
<th>ECO</th>
<th>CCOT</th>
<th>SAMF</th>
<th>MUSTER</th>
<th>DSST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision (20 px, %)</td>
<td>55.8</td>
<td>56.8</td>
<td>44.7</td>
<td>50.9</td>
<td>42.6</td>
</tr>
<tr>
<td>Success Rate (AUC, %)</td>
<td>46.1</td>
<td>47.1</td>
<td>32.7</td>
<td>37.2</td>
<td>28.5</td>
</tr>
<tr>
<td>FPS</td>
<td>53</td>
<td>12</td>
<td>5</td>
<td>1</td>
<td>35</td>
</tr>
</tbody>
</table>

*EnKCF can outperform low-speed state-of-the-art tracker on low-cost embedded system.*
Optimal Combination and Order of Deployment

Results on Different Order of Deployment of Correlation Filters on the UAV123 dataset.
# C++ Code

https://github.com/buzkent86/EnKCF_Tracking_WACV18

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunTracking</td>
<td>readme update</td>
<td>a month ago</td>
</tr>
<tr>
<td>detector</td>
<td>Camera Motion Model Removal Step Added</td>
<td>9 months ago</td>
</tr>
<tr>
<td>main</td>
<td>Datasets Updated</td>
<td>3 months ago</td>
</tr>
<tr>
<td>tracker</td>
<td>Fixed Template Size Added</td>
<td>9 months ago</td>
</tr>
<tr>
<td>CMakeLists.txt</td>
<td>More typos fixed, and grammar mistakes corrected</td>
<td>3 months ago</td>
</tr>
<tr>
<td>README.md</td>
<td>readme update</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

**Description**

This is the C++ implementation of the proposed EnKCF tracker. It includes implementation of a bootstrap particle filter and ensemble of kernelized correlation filters. We suggest the user to disable the particle filter in the case of uncompensated platform motion. You can find the information to compile and run the tracker below.

**To Compile**

```
cd C++_Implementation
mkdir build
cd build
exile
```