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**.

No GPUs!
CPU only!

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

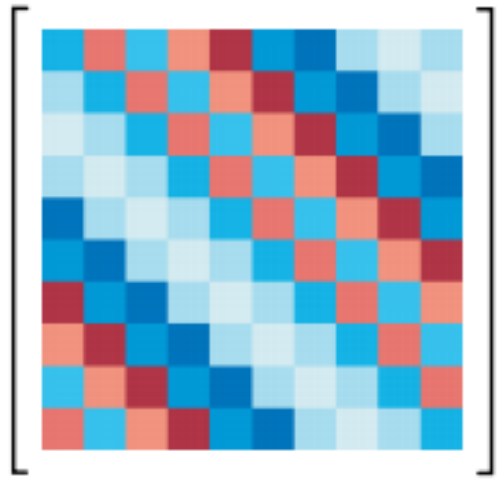
Run-time performance drops dramatically on low-cost embedded-systems (10-20 times less speed)

>300 fps → >30 fps

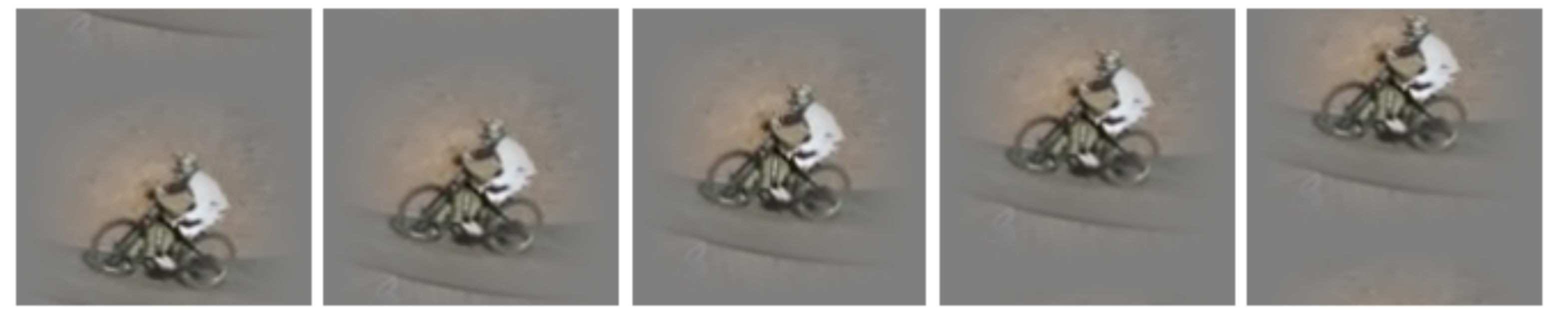
KCF, DCF, CSK, MOSSE

Correlation Filter Trackers

Kernelized Correlation Filter Tracking

$X = C(\mathbf{x}) =$ 

Base sample
x



1.Ridge Regression -> $\min_{\mathbf{w}} \sum_i \left(f(\mathbf{x}_i) - y_i\right)^2 + \lambda \|\mathbf{w}\|^2$

2.Analytical Solution -> $\mathbf{w} = \left(X^T X + \lambda I\right)^{-1} X^T \mathbf{y}$ ***expensive!!! O(n³)***

3.Circulant Matrix -> $X = F \operatorname{diag}(\hat{\mathbf{x}}) \, F^H$

4.Solution in Frequency Domain (Primal) -> $\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda}$ $\longrightarrow$ ***requires O(nlog(n))***

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

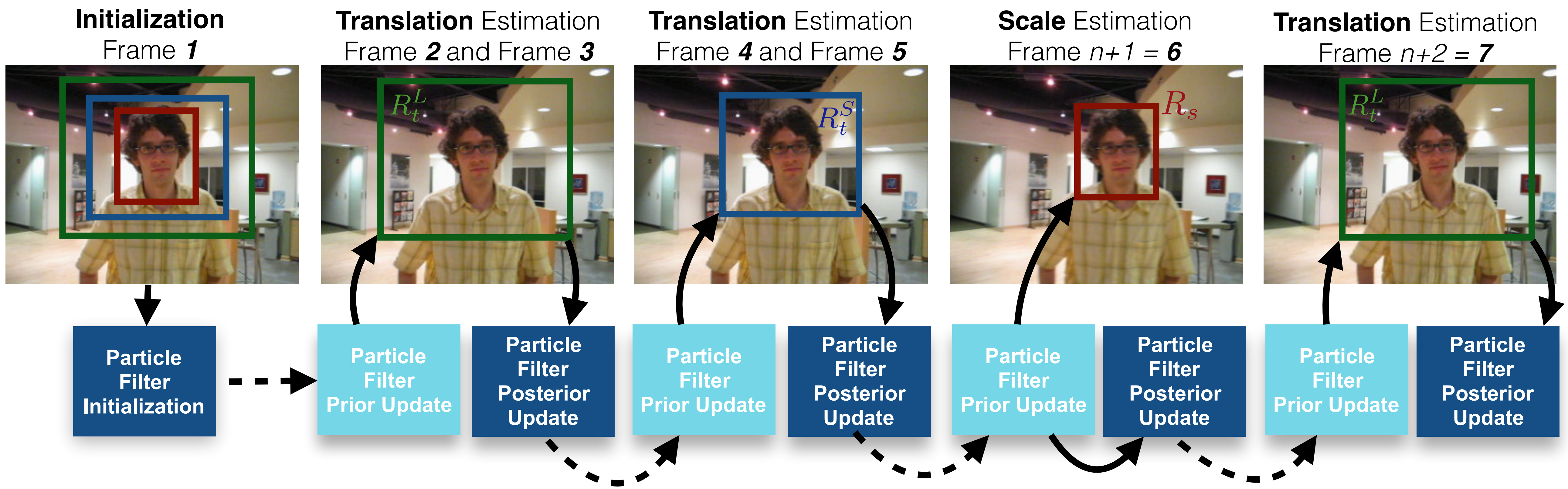
6.Detection -> $\hat{\mathbf{f}}(\mathbf{z}) = \hat{\mathbf{k}}^{\mathbf{xz}} \odot \hat{\alpha}$

runs at >300 fps!!! 🙂

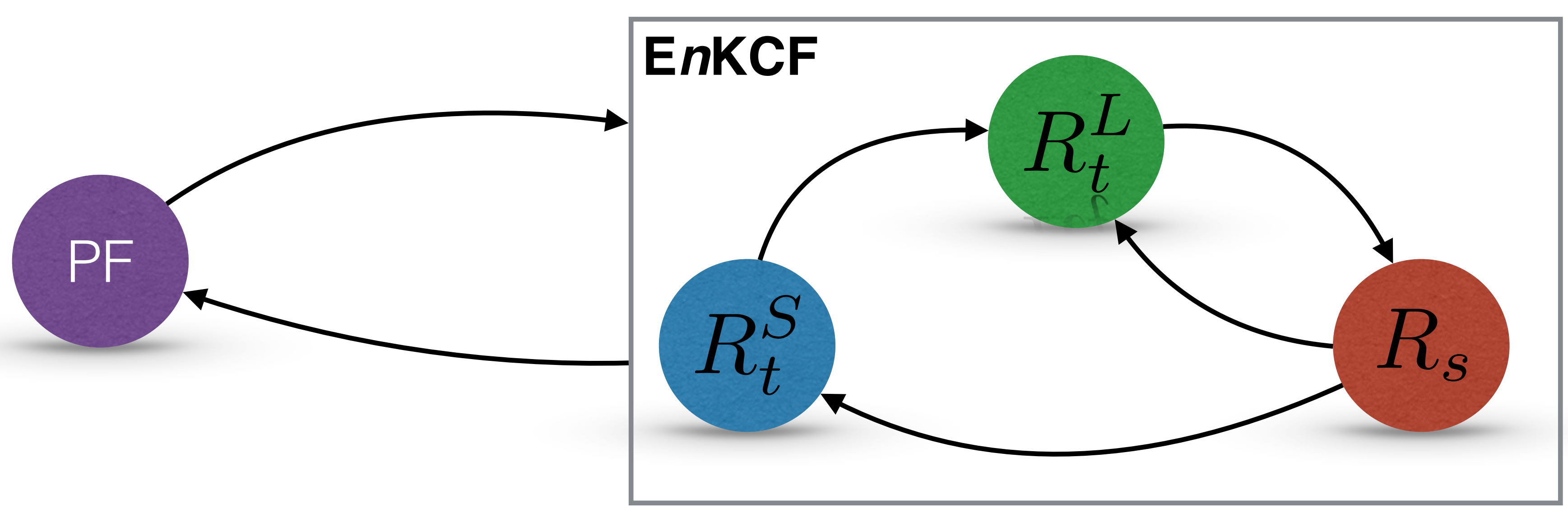
not scale-adaptive 🙁

Henriques, J.F., Caseiro, R., Martins, P. and Batista, J., 2015. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), pp.583-596.

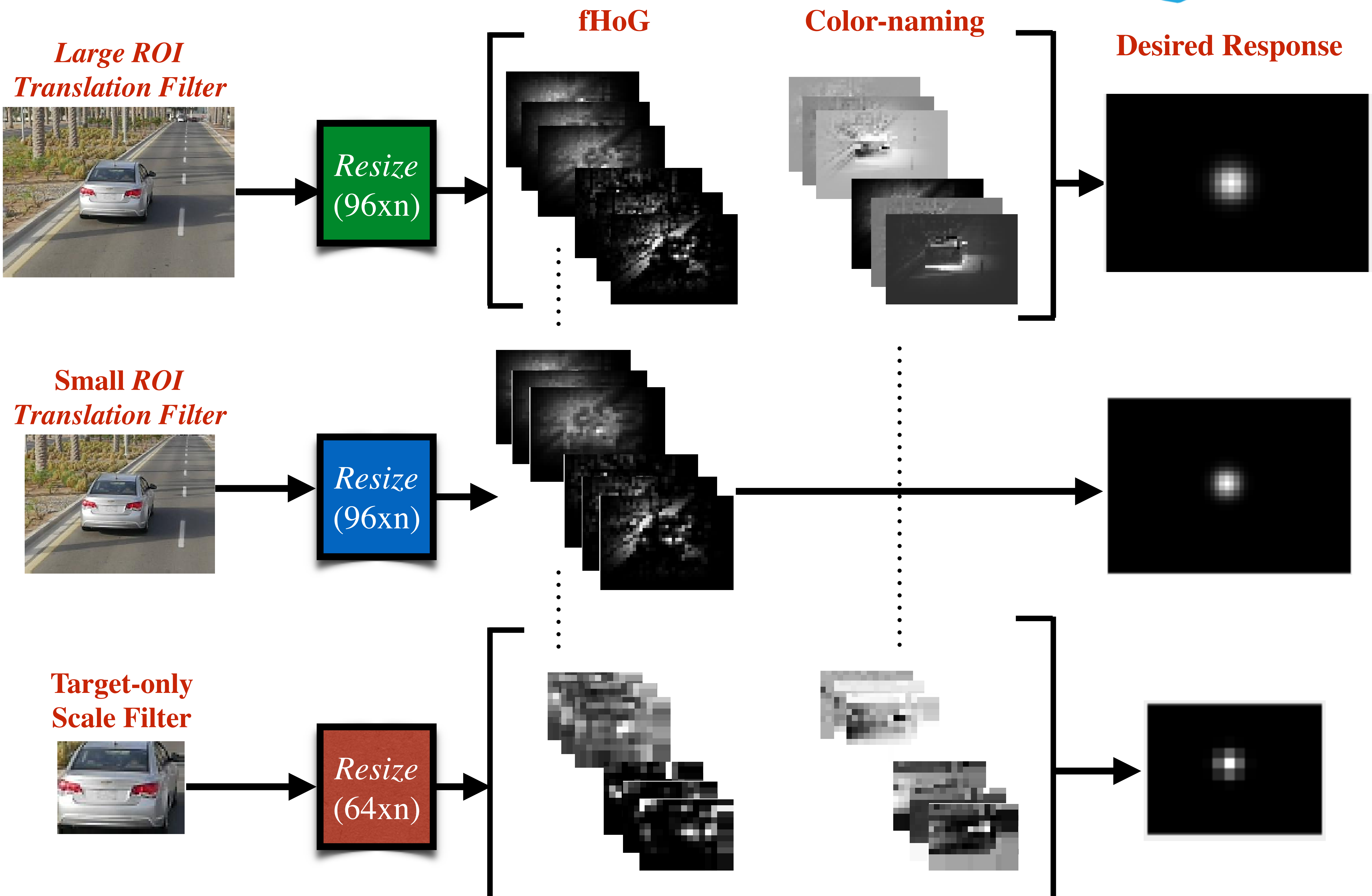
E*n*KCF (Scale Adaptive Tracking at ***>300*** fps)



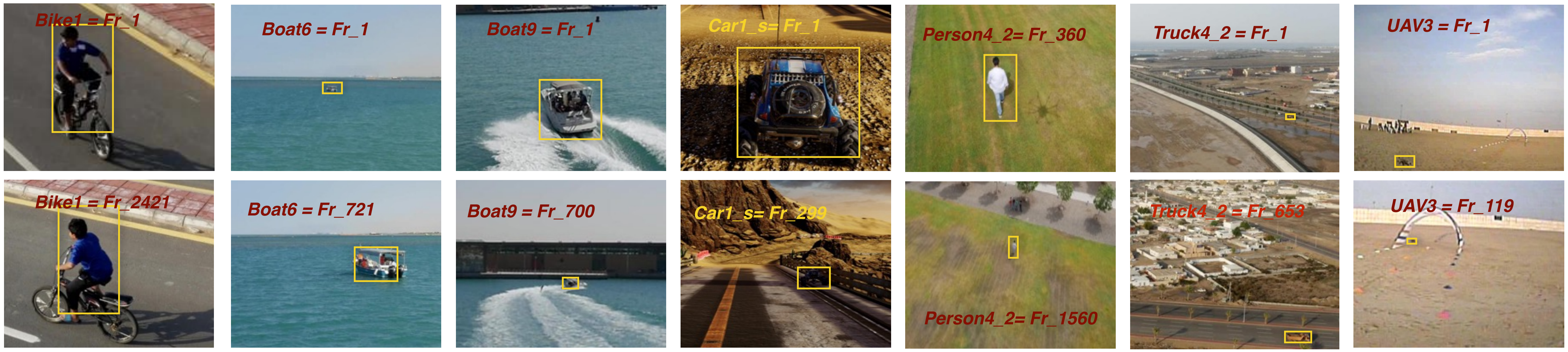
The proposed EnKCF Framework with Particle Filter



Object Representation



Results on ***UAV123*** Dataset



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

>300fps Trackers	EnKCF	KCF	DCF	CSK	MOSSE	STC
Precision (20 px, %)	54.5	52.3	52.6	48.7	46.6	50.7
Success Rate (AUC, %)	40.2	33.6	33.7	31.4	30.1	32.9
FPS	416	296	457	400	512	340

Embedded systems compatible

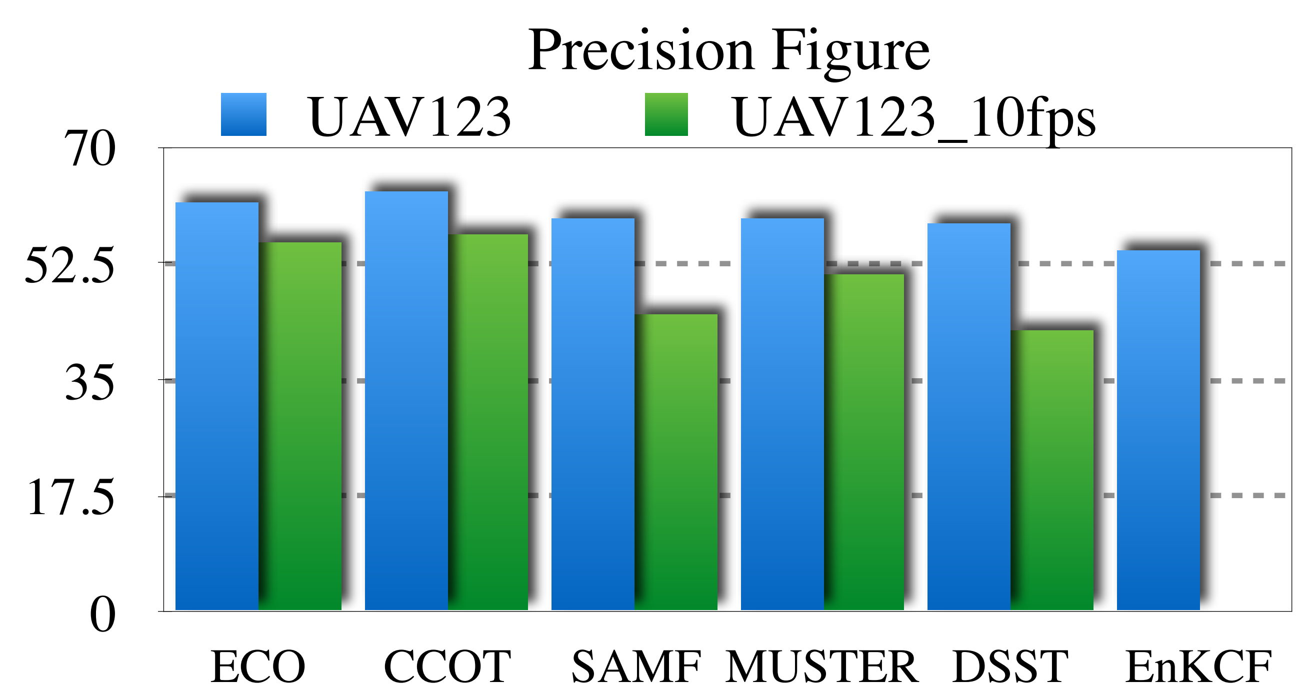
<50fps Trackers	EnKCF	ECO	CCOT	SAMF	MUSTER	DSST
Precision (20 px, %)	54.5	61.6	63.3	59.2	59.3	58.6
Success Rate (AUC, %)	40.2	49.1	49.8	40.3	39.9	36.1
FPS	416	53	12	5	1	35

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**.

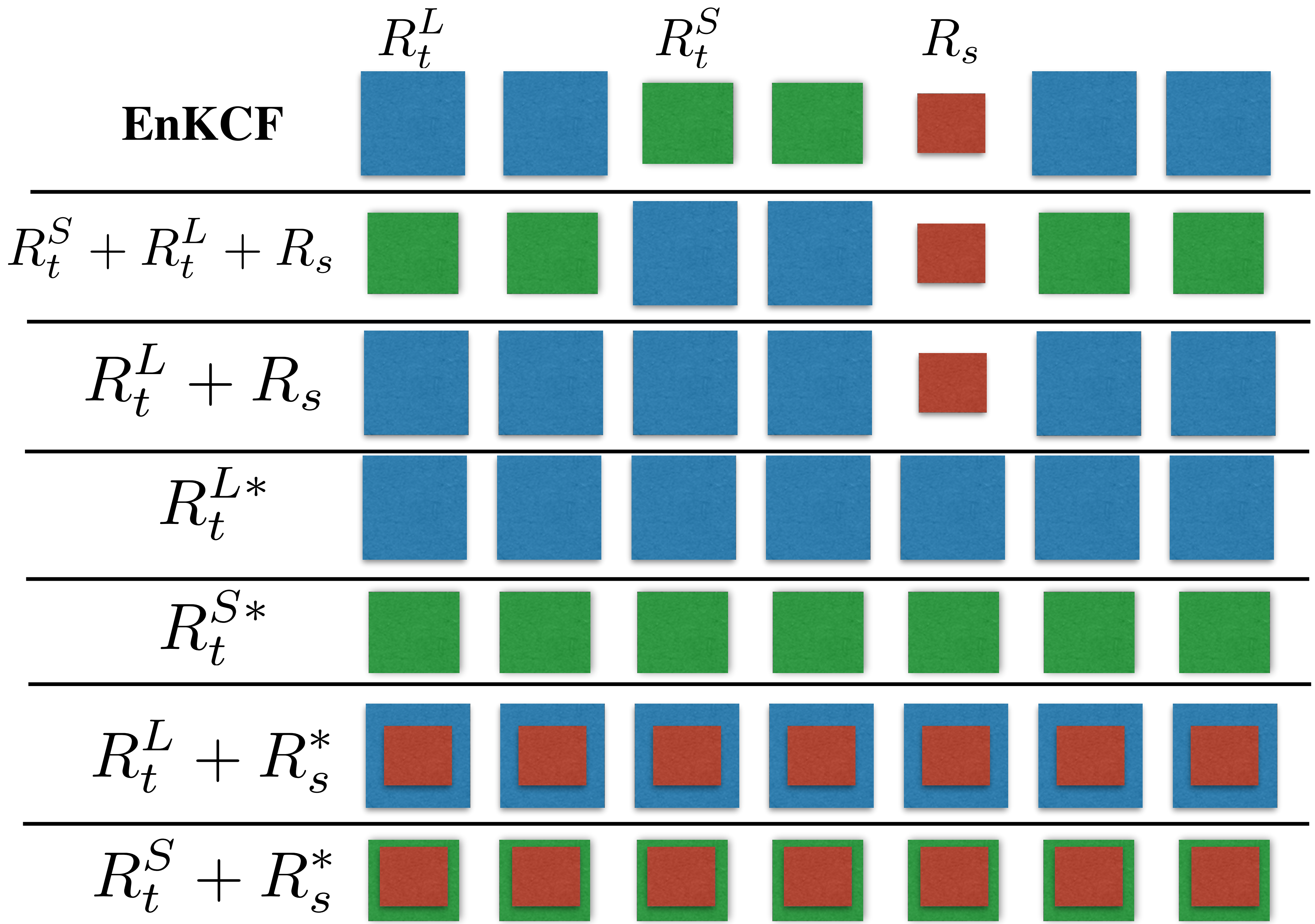
<50fps **Trackers**	**ECO**	**CCOT**	**SAMF**	**MUSTER**	**DSST**
Precision (20 px, %)	55.8	56.8	44.7	50.9	42.6
Success Rate (AUC, %)	46.1	47.1	32.7	37.2	28.5
FPS	53	12	5	1	35



UAV123 — EnKCF > **UAV123_10fps** — ECO, CCOT, DSST, MUSTER, SAMF

***EnKCF can outperform low-speed state-of-the-art tracker on low-cost embedded system.**

Optimal Combination and Order of Deployment



— Best (green) — 2^{nd} Best (red) — 3^{th} Best (blue)

Method	EnKCF	R_t^S+R_t^L+R_s	R_t^L+R_s	R_t^S+R_s	R_t^L*	R_t^S*	R_t^L+R_s*	R_t^S+R_s*
Pr. (20px)	53.9	48.93	52.41	48.10	51.88	51.29	55.85	52.14
SR (50%)	40.2	36.75	38.23	36.04	35.12	34.43	39.89	38.51
FPS	416	412	370	425	365	384	135	151

Results on Different Order of Deployment of Correlation Filters on the UAV123 dataset.

C++ Code

https://github.com/buzkent86/EnKCF_Tracking_WACV18

RunTracking	readme update	a month ago
detector	Camera Motion Model Removal Step Added	9 months ago
main	Datasets Updated	3 months ago
tracker	Fixed Template Size Added	9 months ago
CMakeLists.txt	More typos fixed, and grammar mistakes corrected	3 months ago
README.md	readme update	a month ago

README.md

Description

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

To Compile

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