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, Rochester, NY

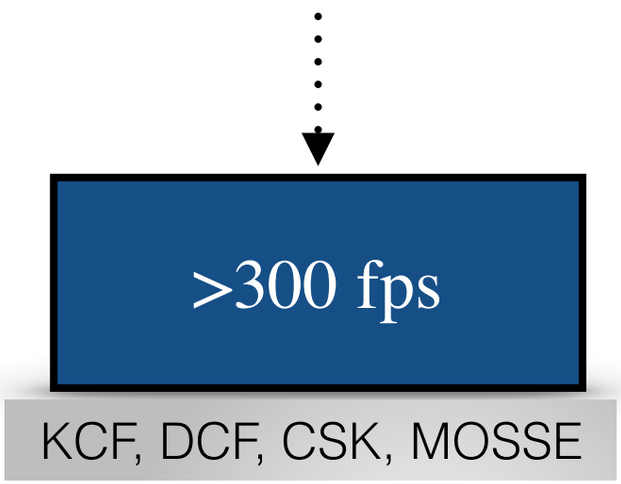
Motivation

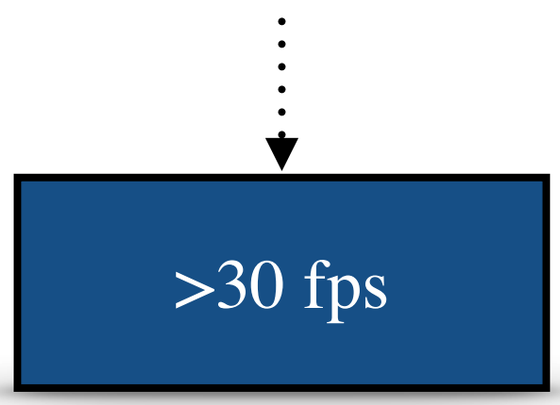
- The goal of this work is to develop an ***online*** and ***single-target*** tracking algorithm that can run at a typical embedded system at real-time in ***>30* fps**.

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)





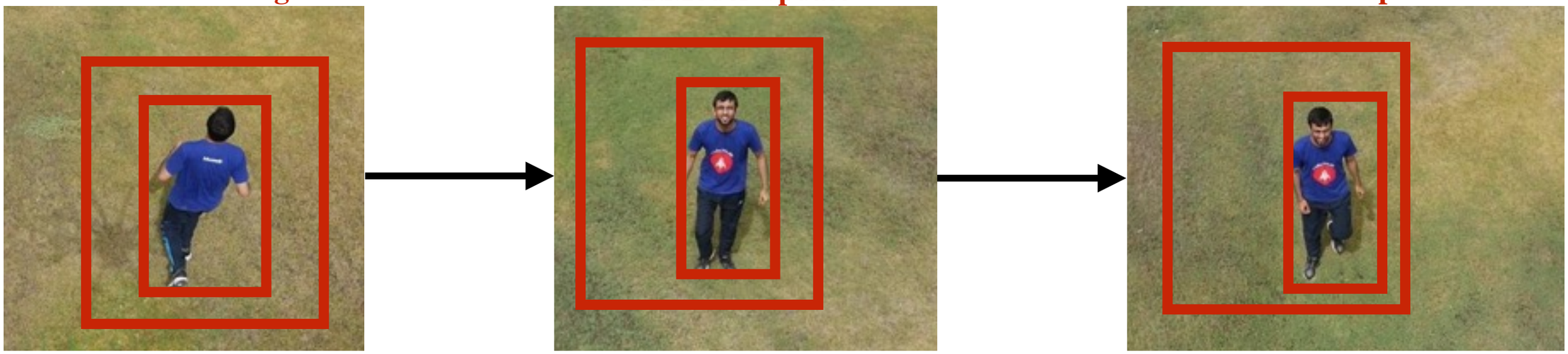
Correlation Filter Trackers
Tracking-by-detection Algorithms

Introduction

Initialization - Frame *1*
Model Learning

1.Detection - Frame *k*
2.Model Update

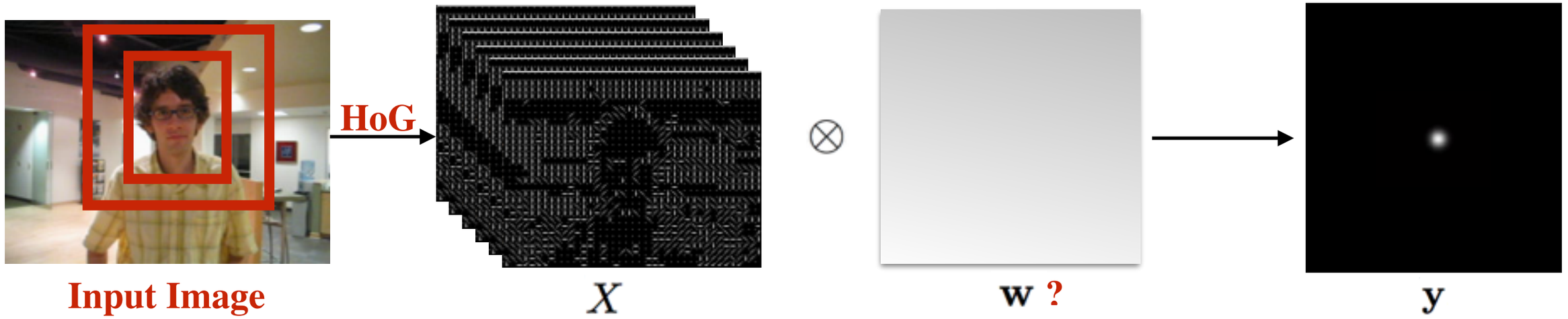
1.Detection - Frame *k+n*
2.Model Update



Tracking-by-detection Framework - KCF

- MIL, Struct, KCF, CCOT and ECO are some of the *tracking-by-detection* algorithms.
- In comparison to deep-learning trackers, their advantages can be listed as
    1. No offline training
    2. Small memory footprint
    3. No need to have GPUs installed on an embedded system.

Correlation Filter Tracking

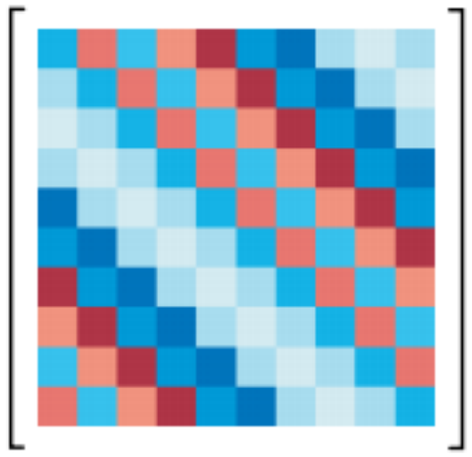


Ridge Regression -> $\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2$

Analytical Solution -> $\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$

requires $O(n^3)$ and expensive matrix inversion

Kernelized Correlation Filter Tracking

$X = C($  $) =$ 

Base sample

x

+30 +15 Base sample -15 -30

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

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

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

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

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

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

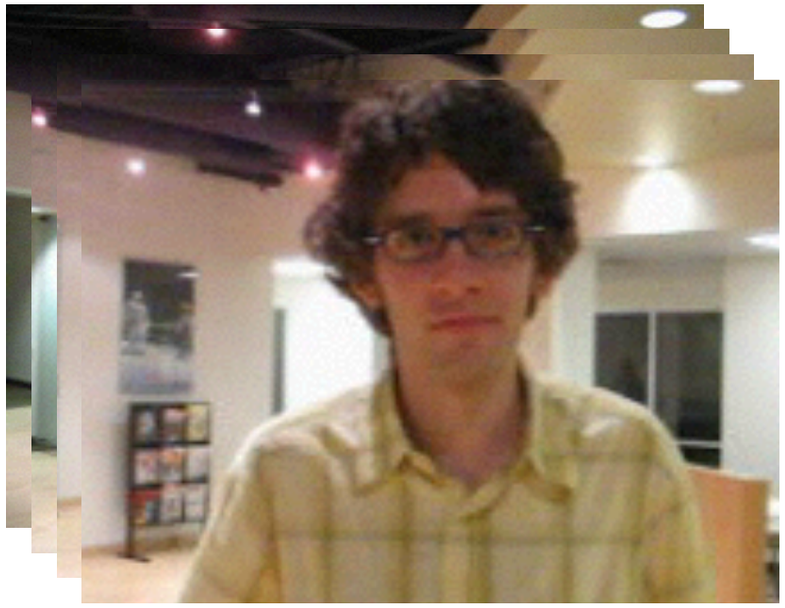
runs at >300 fps!!! 

not scale-adaptive 

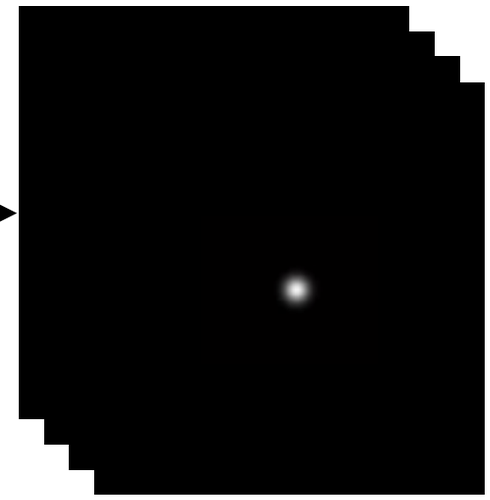
Henriques, J.F., Caseiro, R., Martins, P. and Batista, J., 2015. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), pp.583-596.

Scale Adaptive KCF Trackers (<50fps)

Scale Adaptive Multi-Feature Integration Tracker (SAMF) (HoG + Color Features) (Yang et al ECCV14)

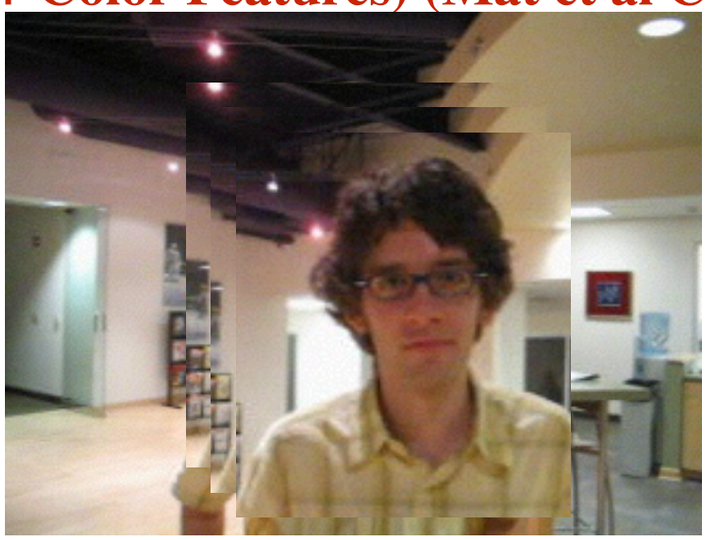


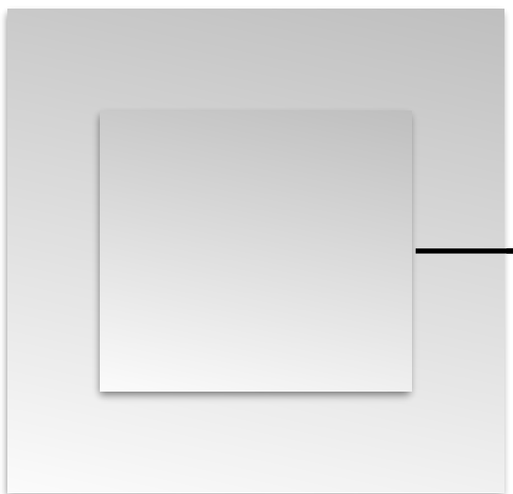
33 **candidates in scale space**

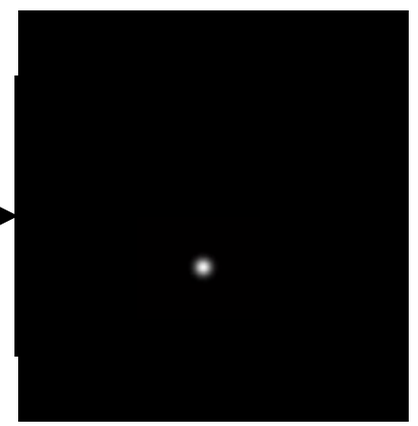


$$\underset{i \in S}{argmax}(PSR(\boldsymbol{R}(\boldsymbol{z}^i)))$$

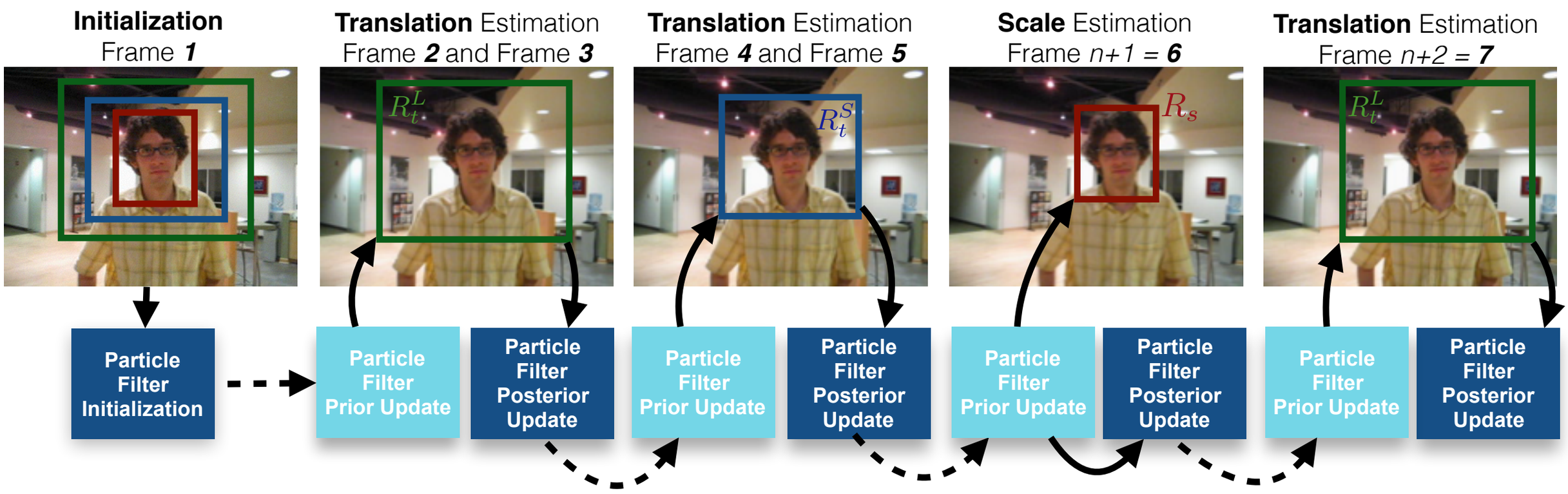
Long-Term Correlation Filter Tracker (HoG + Color Features) (Mat et al CVPR15)



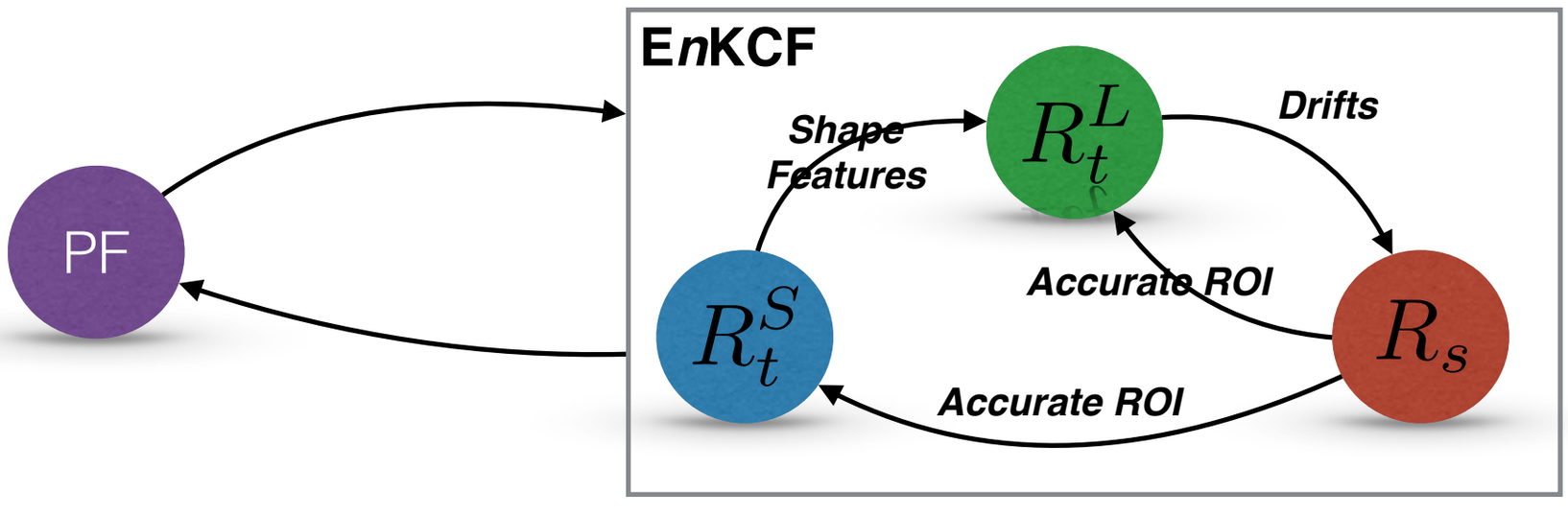




E*n*KCF (Scale Adaptive Tracking at ***>300*** fps)



The proposed EnKCF Framework with Particle Filter

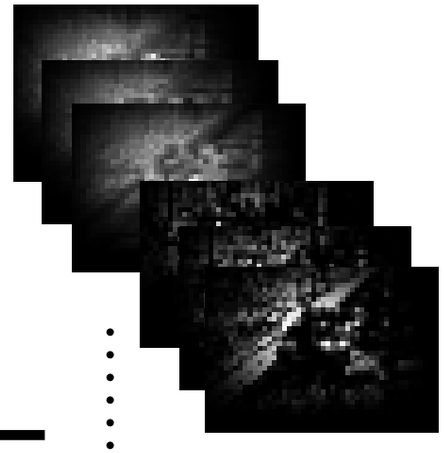


Object Representation

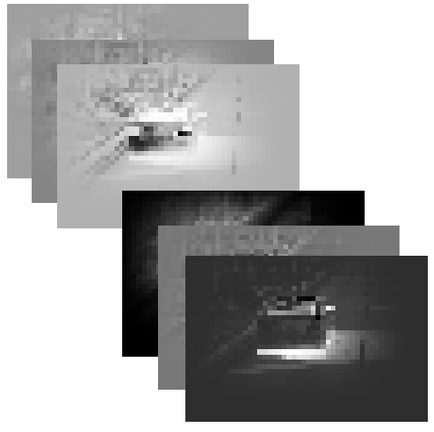
Large ROI Translation Filter

Resize (96xn)

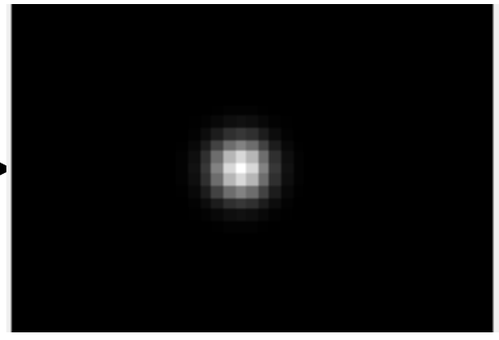
fHoG



Color-naming

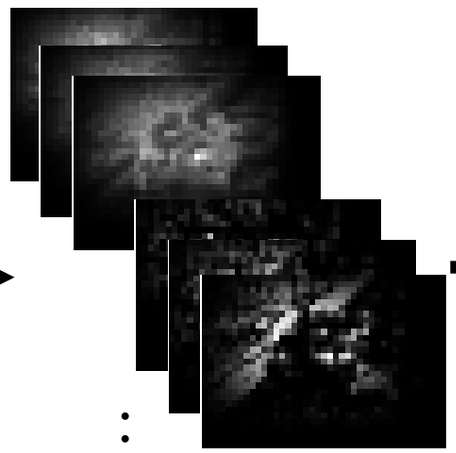


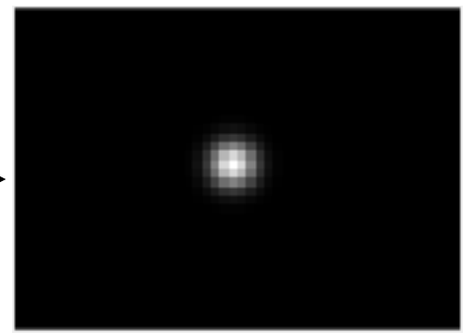
Desired Response



Small *ROI* Translation Filter

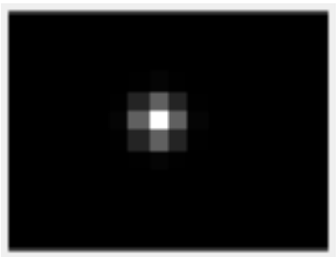
Resize (96xn)





Target-only Scale Filter

Resize (64xn)



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 🙂

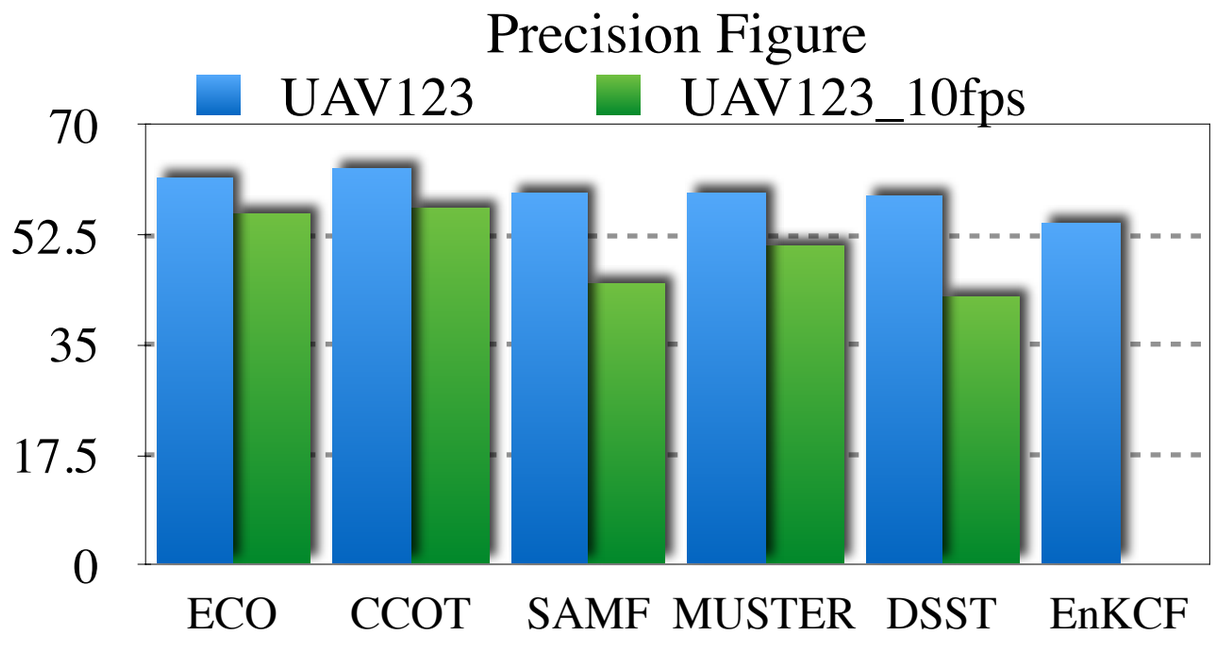
<50fps Trackers	EnKCF	ECO	CCOT	SAMF	MUSTER	DSST
Precision (20 px, %)	54.5	61.6	63.3	59.2	59.3	58.6
Success Rate (AUC, %)	40.2	49.1	49.8	40.3	39.9	36.1
FPS	416	53	12	5	1	35

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

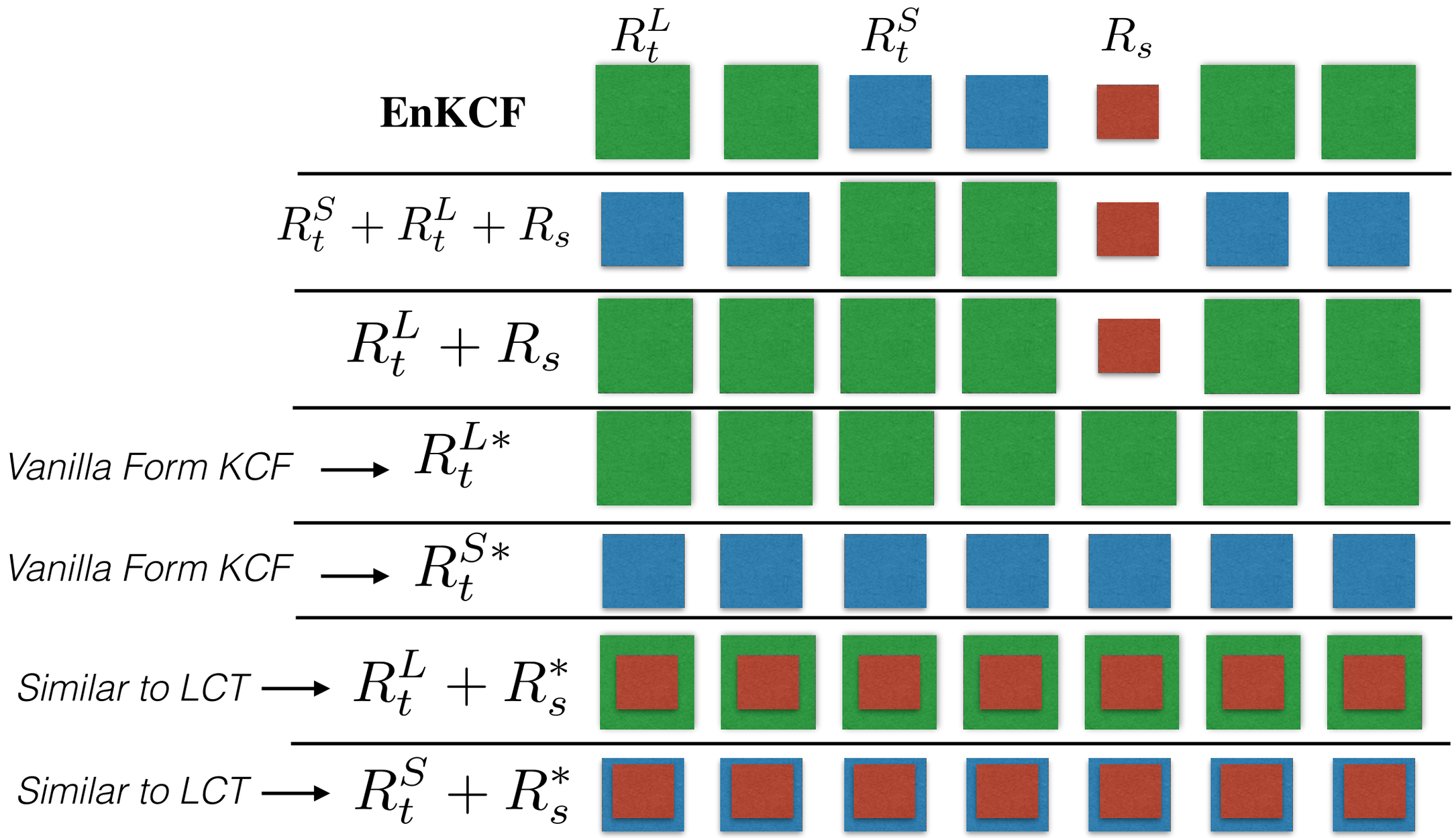
>

UAV123_10fps

ECO, CCOT, DSST, MUSTER, SAMF

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

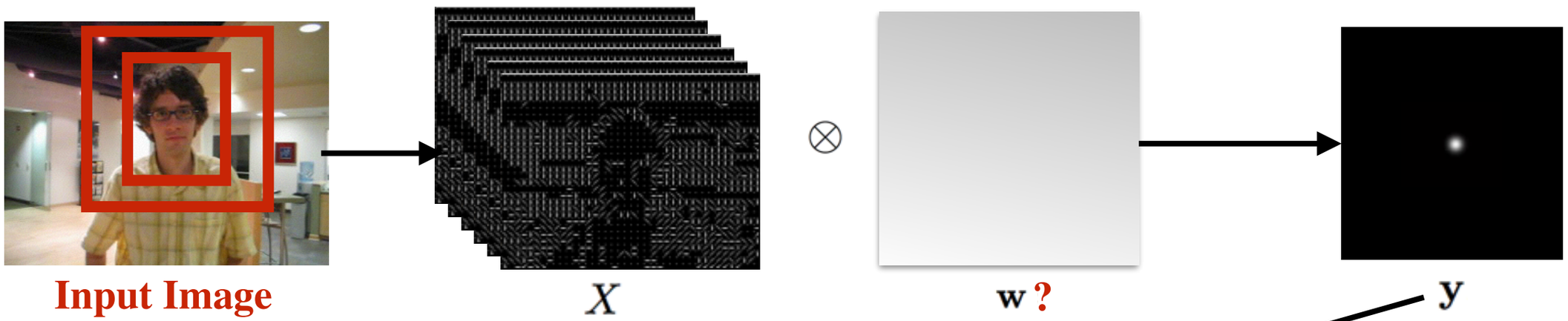
Optimal Combination and Order of Deployment



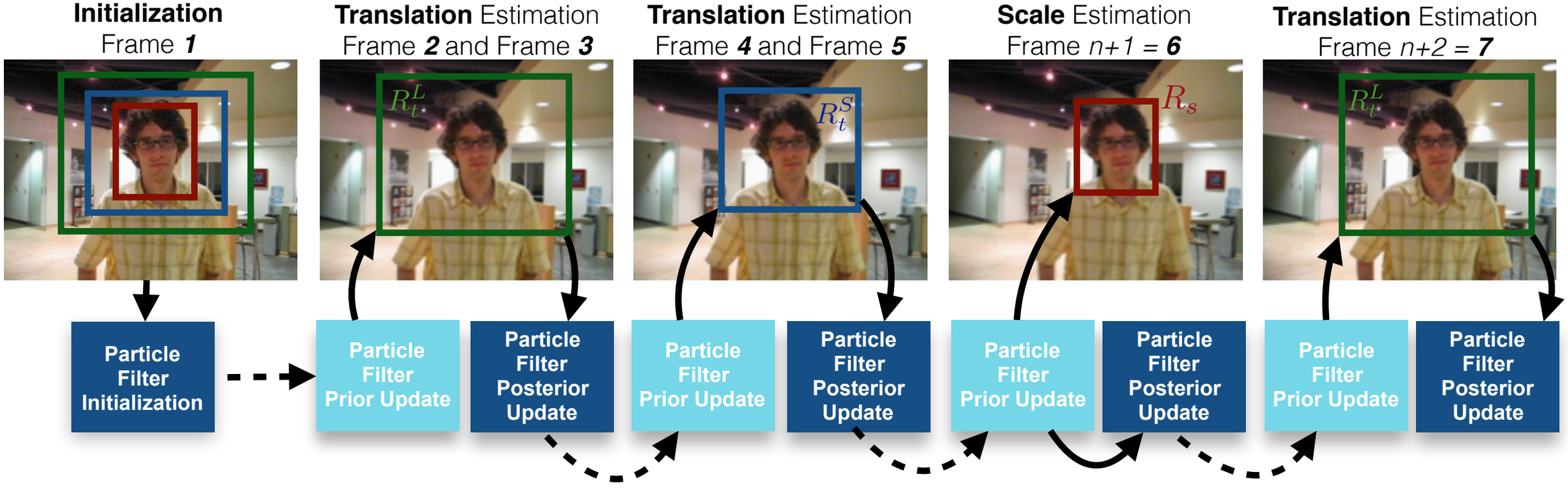
Method	*E*nKCF	$R_t^S+R_t^L+R_s$	$R_t^L+R_s$	$R_t^S+R_s$	R_t^{L*}	R_t^{S*}	$R_t^L+R_s*$	$R_t^S+R_s*$
		— Best		— 2nd Best			— 3th Best	
Pr. (20px)	53.9	48.93	52.41	48.10	51.88	51.29	55.85	52.14
SR (50%)	40.2	36.75	38.23	36.04	35.12	34.43	39.89	38.51
FPS	416	412	370	425	365	384	135	151

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

Future Work



***Can we regress bounding box parameters in a single shot?**



The proposed EnKCF Framework with Particle Filter

***Can we adaptively determine when to deploy the scale filter?**

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
```

Aerial Vehicle Tracking using a Multi-modal Adaptive Hyperspectral Sensor

Name : Burak Uzkent

Advisor : Dr. Matthew J. Hoffman

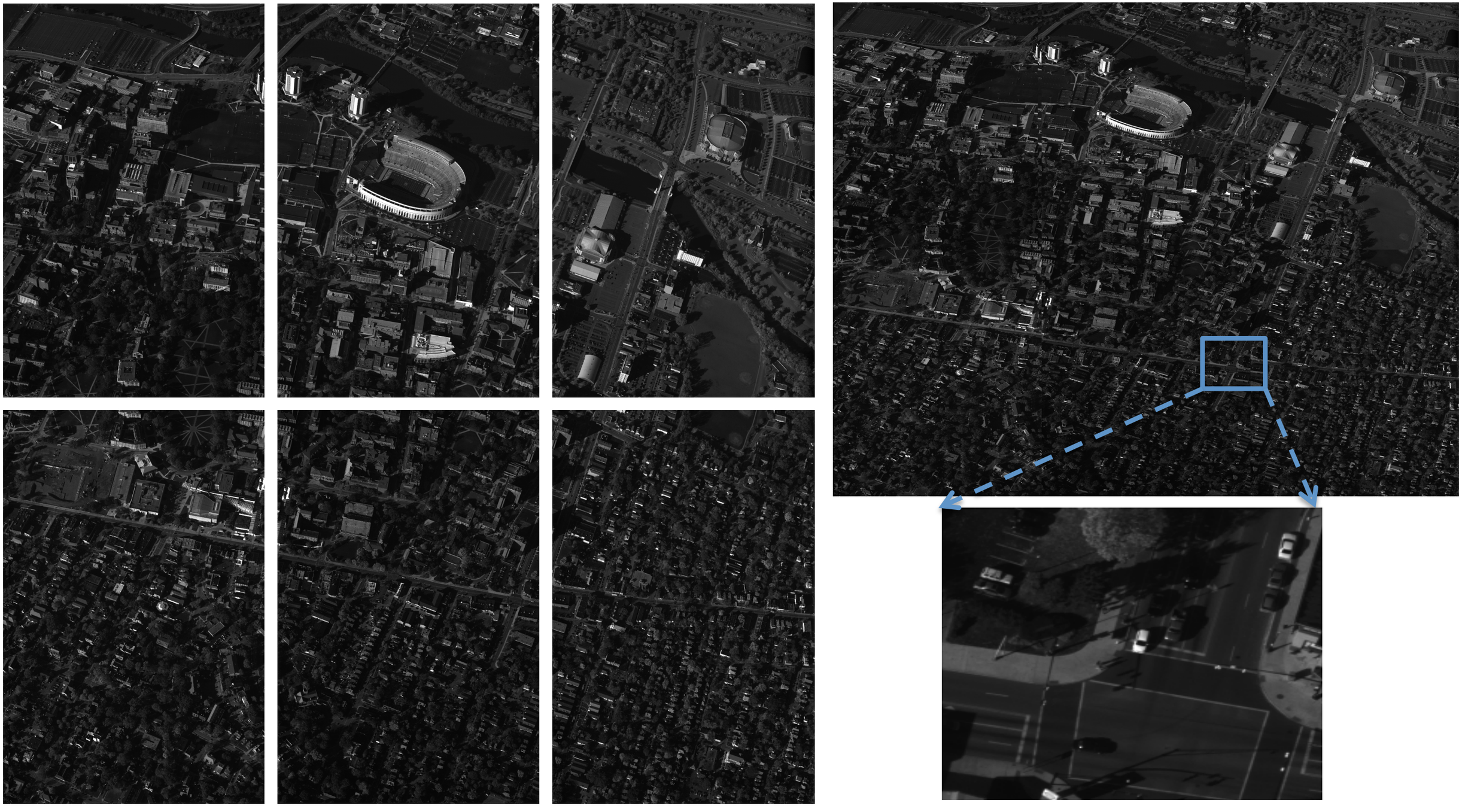
Co-Advisor : Dr. Anthony Vodacek

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

Introduction

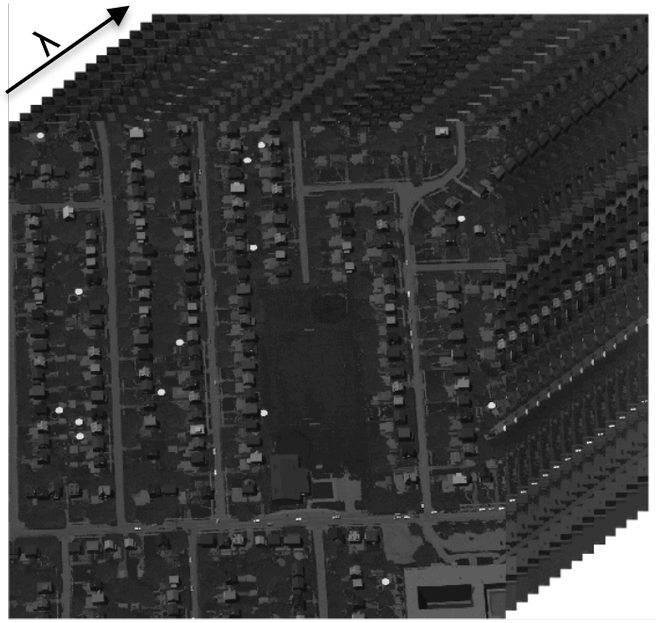
- By using an aerial platform, we want to track all moving objects or an object of interest persistently.
- Aerial Tracking is a more challenging task than the traditional object tracking due to
  - Small number of pixels representing a vehicle
  - Large Camera Motion
  - Parallax effect due to 3-D structures in the scene.
  - Registration errors
  - Severe occlusions
- The Wide Area Motion Imagery (WAMI) Platform is the state-of-the-art sensor that is used for aerial vehicle tracking.

Low Resolution Effect - WAMI



Introduction

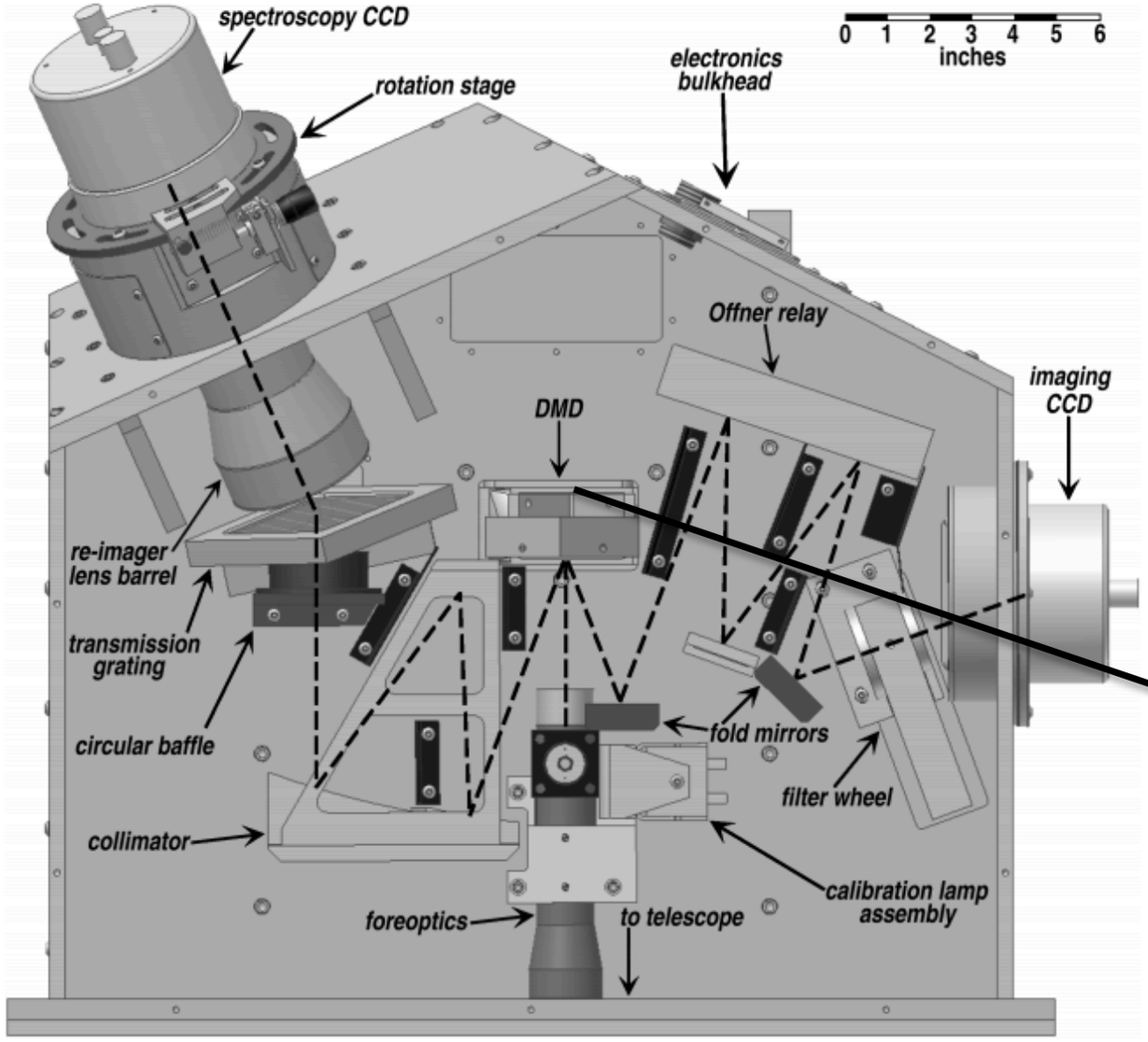
- However, still, more descriptive sensory information is required to address the challenges of aerial tracking.
- With recent advancements in the sensor technology, quick hyperspectral data acquisition is possible.



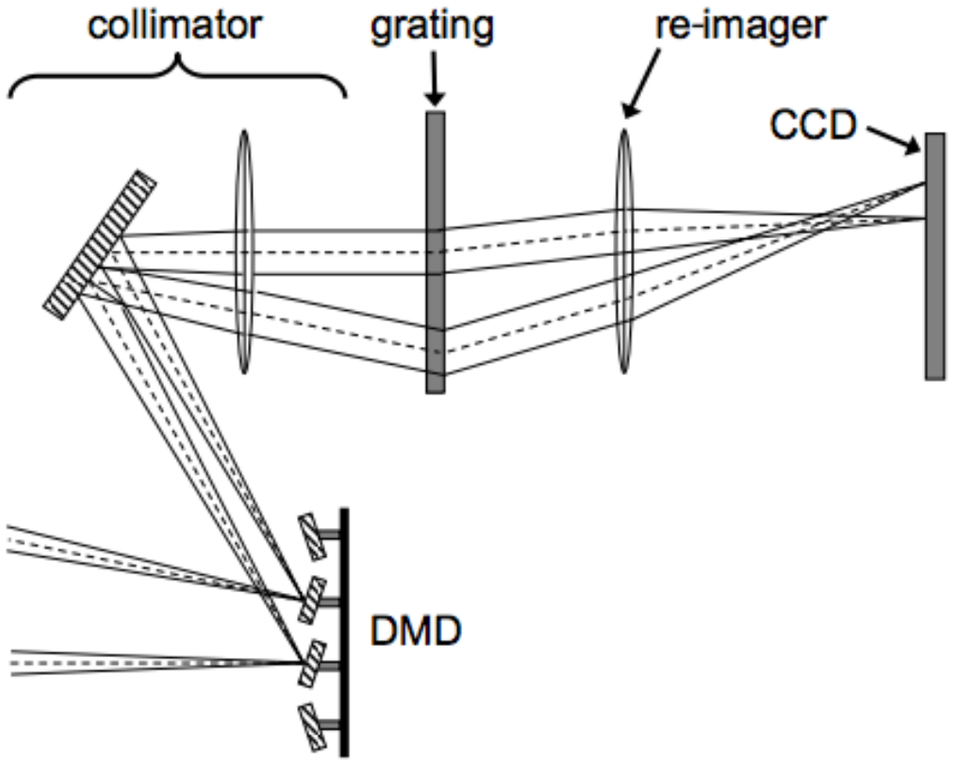
Visualization of a Hyperspectral Image Cube

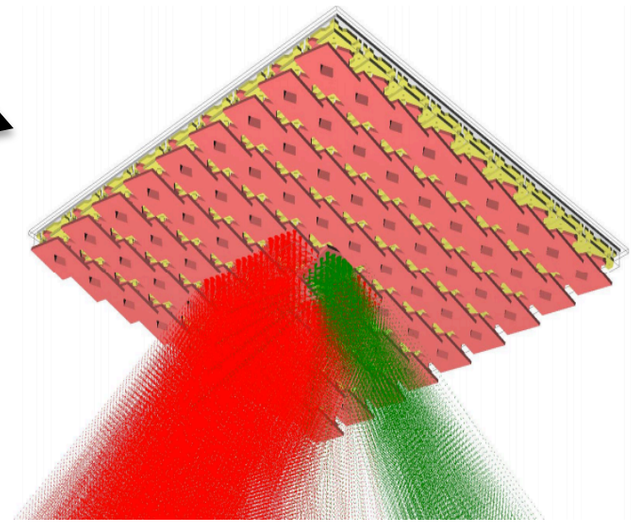
- One example of such sensor is the Rochester Institute of Technology Multi-object Spectrometer (RITMOS).

Adaptive Hyperspectral Sensor



Top View of RITMOS





Micromirror Arrays

Meyer, Reed D., et al. "RITMOS: a micromirror-based multi-object spectrometer." SPIE Astronomical Telescopes+ Instrumentation. International Society for Optics and Photonics, 2004.

Scenario Generation

DIRSIG generated RGB image of the scenario
MegaScene-1 Area - Northern Rochester, NY

RGB image screenshot taken from Google Map
(Histogram Equalized)

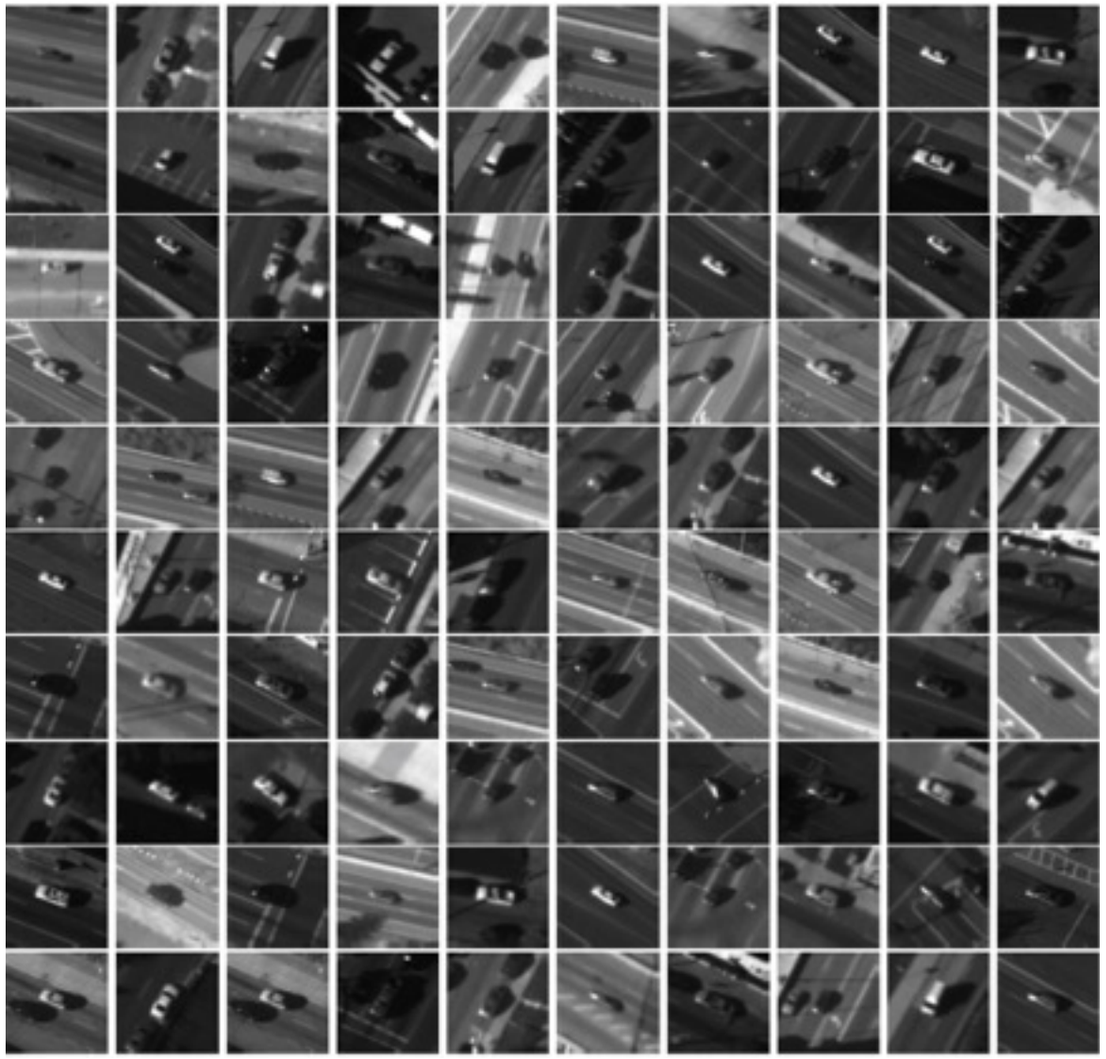
Synthetic Aerial Video

Synthetic Vehicle Classification Dataset

64x64

DIRSIG samples - Synthetic Dataset
55226 Samples

64x64



WAMI samples - Real Dataset
600 Samples

https://buzkent86.github.io/datasets/

Uzkent, Burak, Aneesh Rangnekar, and Matthew J. Hoffman. "Tracking in Aerial Hyperspectral Videos using Deep Kernelized Correlation Filters." IEEE Transactions on Geoscience and Remote Sensing.

Vehicle Classification on WAMI

Training Dataset (DIRSIG - *38000* Samples)

Validation Dataset (DIRSIG -*17000* Samples)

Test Dataset (WAMI - *600* Samples)

- We follow two different strategies to train a convolutional classifier.
  1. Training from scratch.
  2. Fine-tuning on models trained on ImageNet.
  3. *Dilated convolutions* in the first several layers.

Method	ZFNet (from Scratch)	ZFNet (ImageNet)	ZFNet*	AlexNet (WAMI [22])
Accuracy (%)	93.20	92.230	97.0	97.1

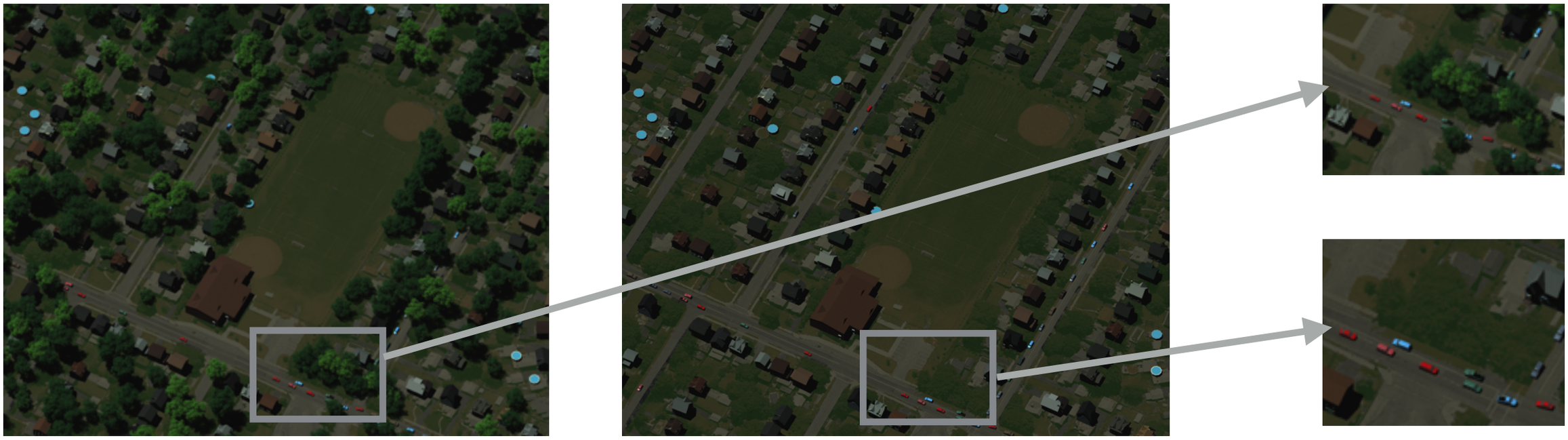
Splits WAMI video (WPAFB09) into *training* and *test* set

Classification accuracies on the WAMI dataset.
The * denotes second stage fine-tuning with 200 WAMI samples.

[22] - Yi, Meng, Fan Yang, Erik Blasch, Carolyn Sheaff, Kui Liu, Genshe Chen, and Haibin Ling. "Vehicle classification in WAMI imagery using deep network." In Sensors and Systems for Space Applications IX, vol. 9838, p. 98380E. International Society for Optics and Photonics, 2016.

Tracking in Aerial Videos

- Tracking in medium-to-high altitude platform is a challenging task due to
    1. Low spatial resolution. (0.3m GSD)
    2. Low temporal resolution. (1.42 fps)
    3. Severe occlusions.
    4. Large camera motion.



A Frame with Dense Trees

A Frame without Trees

Challenges with Tracking-by-Detection Algorithms

- Discriminative trackers work well in *high frame rate* videos where the objects are represented by *large number of pixels*.
- Recently, they are replaced with end-to-end deep learning trackers with the availability of large datasets
  - UAV123 (2017)
  - OTB100 (2015)
  - TrackingNet (2018)
- Another way of improving tracking-by-detection algorithms is to use *deep convolutional features* rather than hand-crafted ones.
- Designing an *end-to-end deep learning tracker* for aerial platforms are not feasible due to the lack of a large dataset.

Expanding ROI with KCF

Frame k ***Frame k+1***

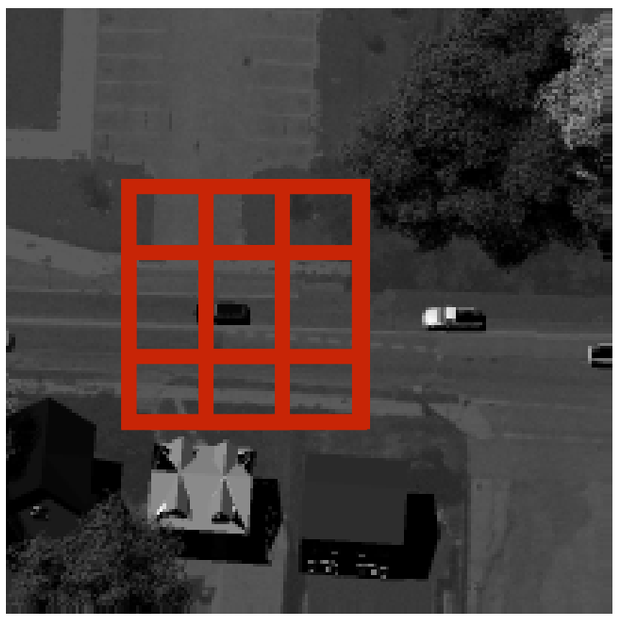
1.42 fps

The search area of the ***KCF***

Frame k ***Frame k+1***

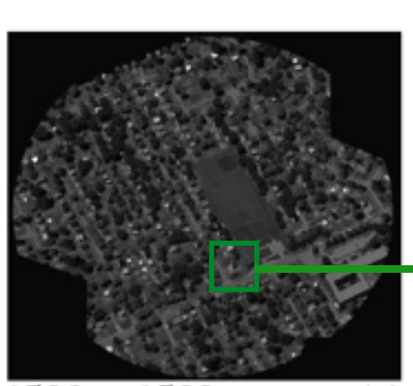
30 fps

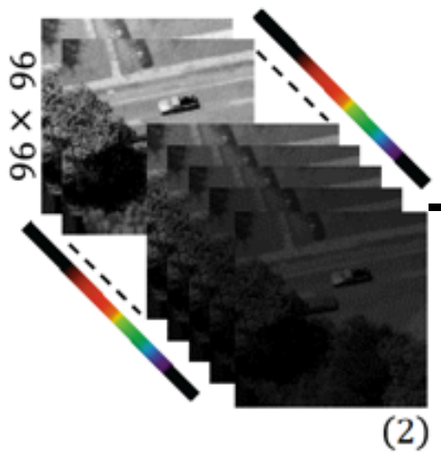
- One can run a KCF on different regions to cover a large area.

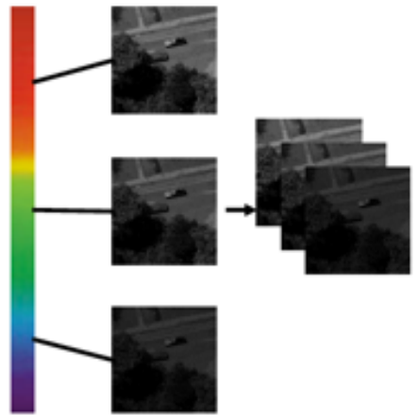


Computationally *expensive* due to *feature extraction* at each region

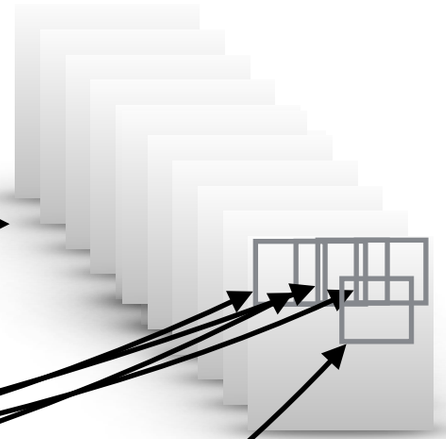
Proposed DeepHKCF Tracker

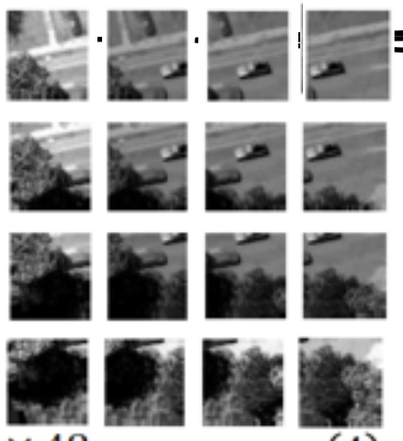


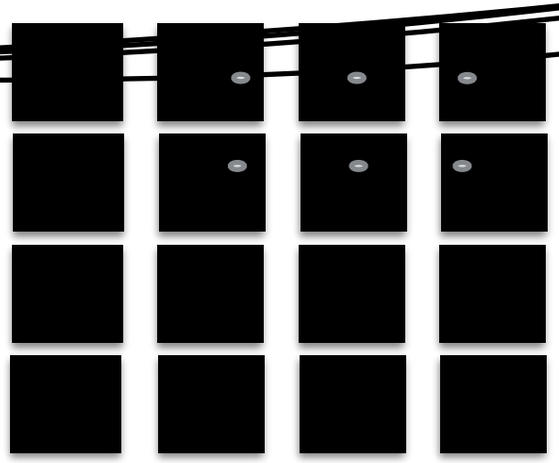




Forward-pass through CNN layers







Update KCF Model

(1) Image Registration — **0.1 s.**

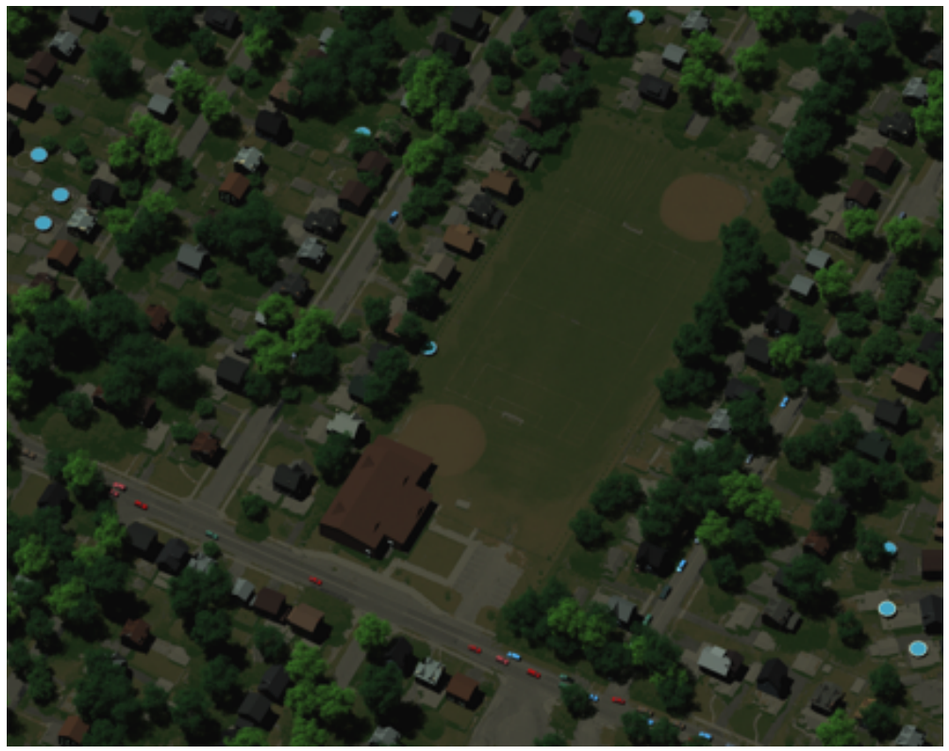
(2) Small FOV HSI Image Acquisition — **0.1 s.**

(3) Selecting RGB channels for feature extraction

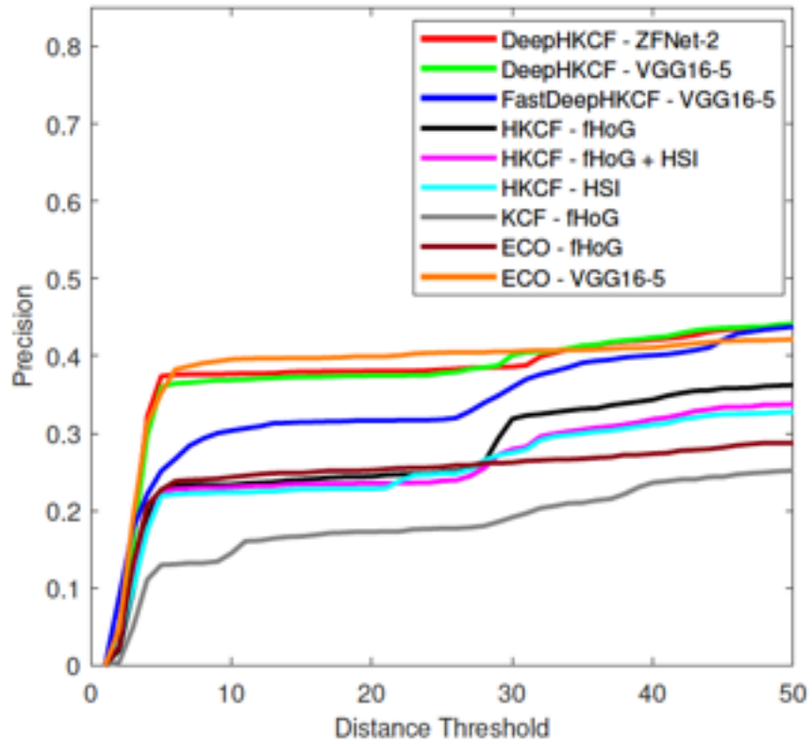
(4) Single KCF multiple ROIs Approach — **0.5 s.**

(5) a. Detection
(6) b. Training

Comparison with Advanced Discriminative Trackers



81th **Frame of the Dense Trees Scenario**

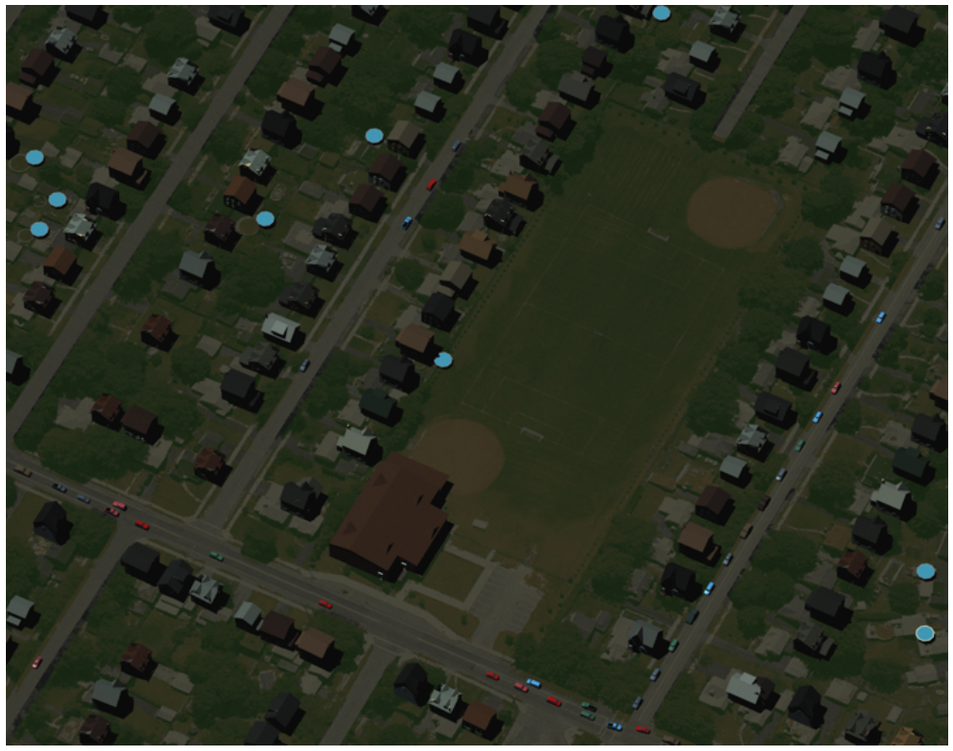


Method	DeepHKCF ZFNet-2 (4 x 4 ROI)	DeepHKCF VGG16-5 (4 x 4 ROI)	FastDeepHKCF VGG16-5 (4 x 4 ROI)	HKCF HSI + fHOG (4 x 4 ROI)	HKCF HSI (4 x 4 ROI)	HKCF fHOG (4 x 4 ROI)	HKCF fHOG (1 x 1 ROI)	ECO fHOG	ECO VGG16-5
Pr. (20 px)	38.08	37.48	31.68	23.57	23.88	24.69	17.31	25.51	40.07
Pr. (50 px)	43.83	44.16	44.01	33.80	32.82	36.28	25.33	28.81	42.15
CLE	156.77	156.67	143.66	196.80	191.19	180.47	209.63	222.61	181.64
FPS	0.41	0.17	0.65	1.67	2.10	2.59	8.22	2.55	0.83

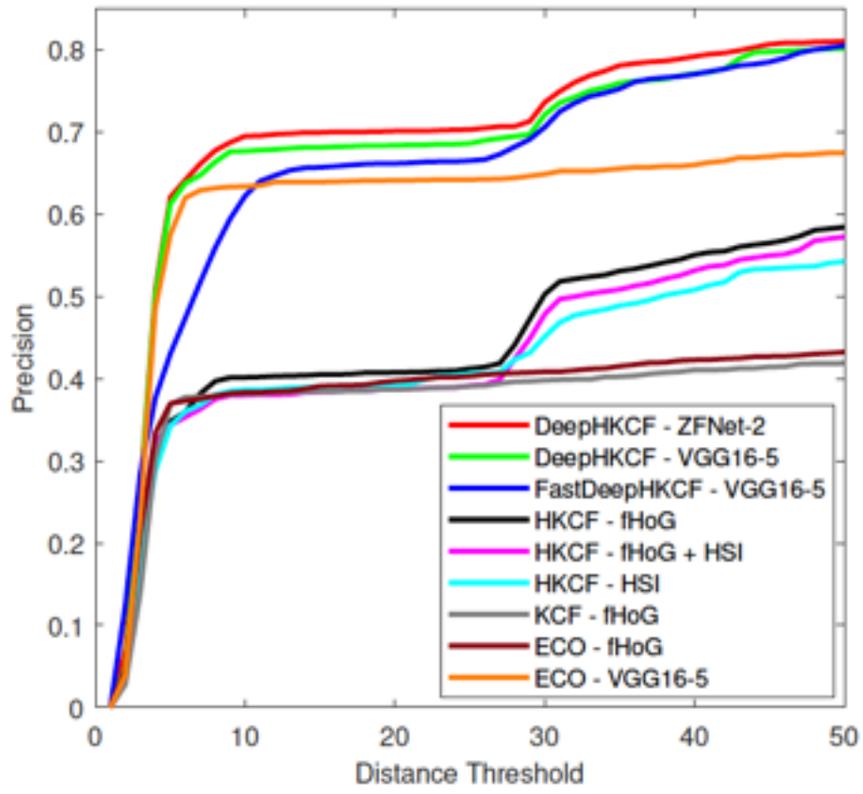
— Best — 2^{nd} Best — 3^{th} Best

Detailed Analysis on the Without Trees Scenario

Comparison with Advanced Discriminative Trackers



81th Frame of the Without Trees Scenario

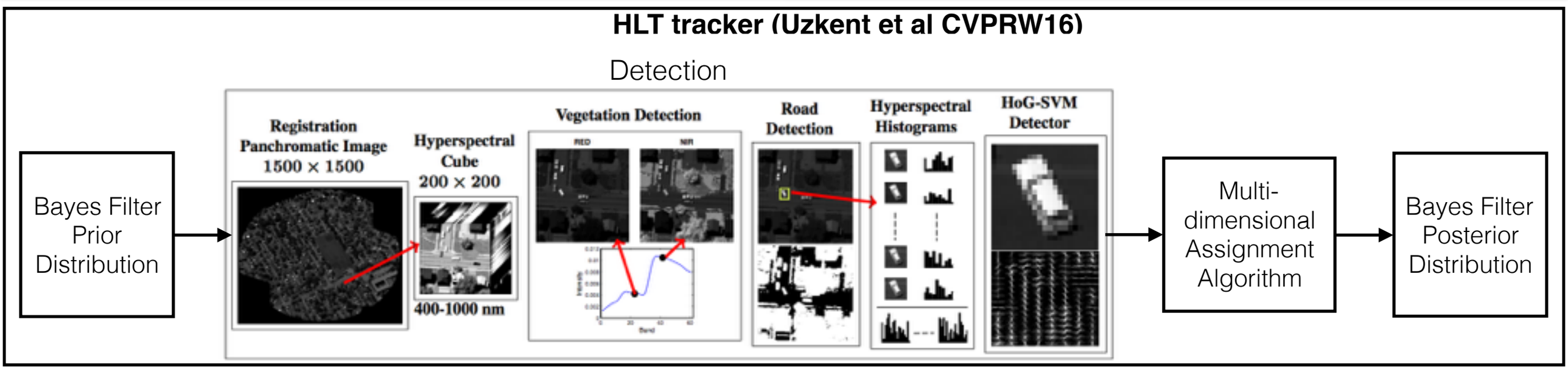


Method	DeepHKCF ZFNet-2 (4 x 4 ROI)	DeepHKCF VGG16-5 (4 x 4 ROI)	FastDeepHKCF VGG16-5 (4 x 4 ROI)	HKCF HSI + fHOG (4 x 4 ROI)	HKCF HSI (4 x 4 ROI)	HKCF fHOG (4 x 4 ROI)	HKCF fHOG (1 x 1 ROI)	ECO fHOG	ECO VGG16-5
Pr. (20 px)	70.13	68.45	66.26	38.79	39.53	39.30	38.74	39.86	64.15
Pr. (50 px)	81.05	80.27	80.65	57.56	54.30	58.58	42.12	43.24	67.61
CLE	48.97	51.71	51.15	118.73	146.30	119.04	179.71	168.43	113.46
FPS	0.51	0.22	1.11	3.01	2.32	2.98	25.11	2.70	1.19

— Best — 2^{nd} Best — 3^{th} Best

Detailed Analysis on the Without Dense Trees Scenario

Comparison with Recent Hyperspectral Trackers

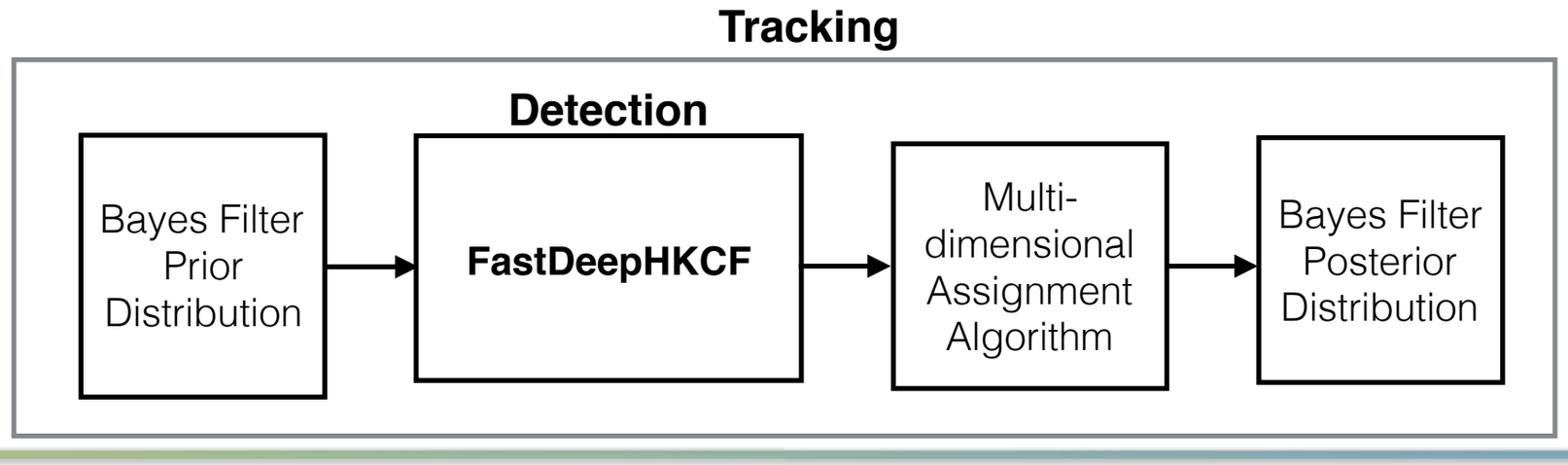


Method	DeepHKCF ZFNet-2	FastDeepHKCF VGGNet-5	HLT [6] (5D)	HLT [6] (2D)
Pr. (20 px) Trees	38.08	31.71	**51.69**	41.86
Pr. (20 px) No trees	**70.13**	66.26	64.42	57.25
Pr. (50 px) Trees	43.83	44.01	**55.12**	46.72
Pr. (50 px) No trees	**81.05**	80.27	71.27	68.31
CLE Trees	156.74	143.66	**135.03**	158.12
CLE No trees	**48.97**	51.71	65.36	91.97
FPS	0.51	**1.11**	1.01	1.09

Comparison to the HLT tracker

Conclusion

- A discriminative tracker (KCF) is translated to aerial tracking domain and its search area is enlarged with the single-KCF-multiple ROIs approach.
- Tracking performance is boosted by using deep convolutional features pre-trained on ImageNet.
- FastDeepHKCF outperforms ECO, but is outperformed by HLT in the dense trees scenario.
- To improve tracking in dense trees scenario, we can integrate a Bayes Filter and Multi-dimensional Assignment algorithm.



Object Detection in Low Resolution Satellite Images

Burak Uzkent
Computer Vision Engineer, Planet Labs.

Mailiao Refinery, Taiwan – May 31, 2016

Introduction to Satellite Images

SkySat Image (0.9m)

DigitalGlobe Image (0.3m)

PlanetScope Image (3.0m)

PlanetScope Image (3.0m)

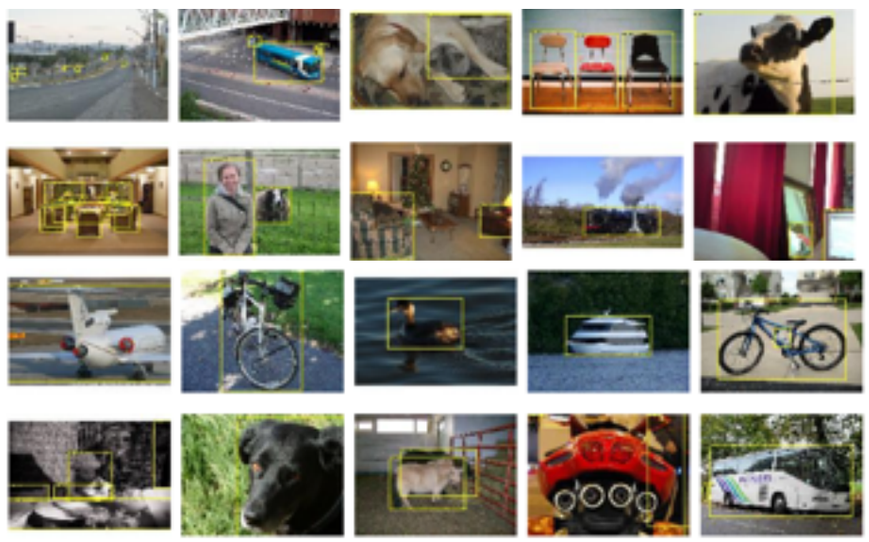
Introduction

- Object detection in satellite images is a challenging task due to:
  - Low spatial resolution.
  - Occlusions cast by clouds.
  - Background clutter.
- To tackle this problems, one can rely on large datasets covering all the scenarios for each object.
- With large datasets, we can train convolutional object detectors to detect the objects.
- However, publicly available datasets in satellite imaging domain is limited and mostly come from higher spatial resolution images.

Object Detection Datasets

Traditional Object Detection Datasets

PASCALVOC07 - 20 Classes - 10K Images



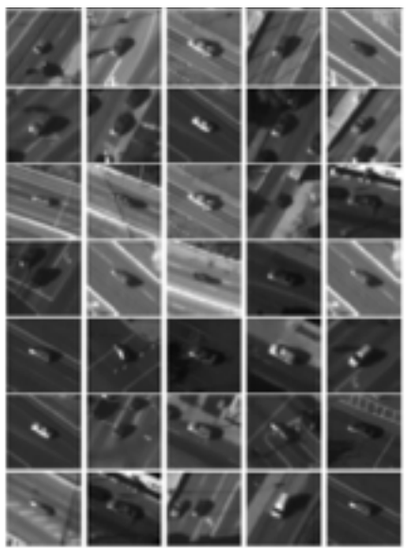
PASCALVOC12 - 20 Classes - 12K Images

MSCOC0 - 80 Classes - 330K Images

Aerial Object Detection Datasets

CCOW (32.700 cars)

DIRSIG (56.000 cars)



WAMI (10.000 cars)

PlanesNet - 8000 Planes

ShipsNet - 700 Ships

+ Soon to be Released Datasets

DigitalGlobe is releasing a large scale object detection dataset!!!

https://arxiv.org/abs/1802.07856

Non-stationary Objects						Stationary Objects	
Fixed-Wing Aircraft	**Passenger Vehicle**	**Truck**	**Railway Vehicle**	**Maritime Vessel**	**Engineering Vehicle**	**Building**	**None**
Small Aircraft	Small Car	Pickup Truck	Passenger Car	Motoboat	Tower Crane	Hut/Tent	Helipad
Cargo Plane	Bus	Utility Truck	Cargo Car	Sailboat	Container Crane	Shed	Pylon
		Cargo Truck	Flat Car	Tugboat	Reach Stacker	Aircraft Hangar	Shipping Container
		Truck w/Box	Tank Car	Barge	Straddle Carrier	Damaged Building	Shipping Container Lot
		Truck Tractor	Locomotive	Fishing Vessel	Mobile Crane	Facility	Storage Tank
		Trailer		Ferry	Dump Truck		Vehicle Lot
		Truck w/Flatbed		Yacht	Haul Truck		Construction Site
		Truck w/Liquid		Container Ship	Scraper/Tractor		Tower Structure
				Oil Tanker	Front Loader		Helicopter
					Excavator		
					Cement Mixer		
					Ground Grader		
					Crane Truck		

Table 2: Parent and child denominations for all 60 classes. Parent classes are at the headings of each column. The only exception is the last column 'None', which corresponds to classes that have no parent.

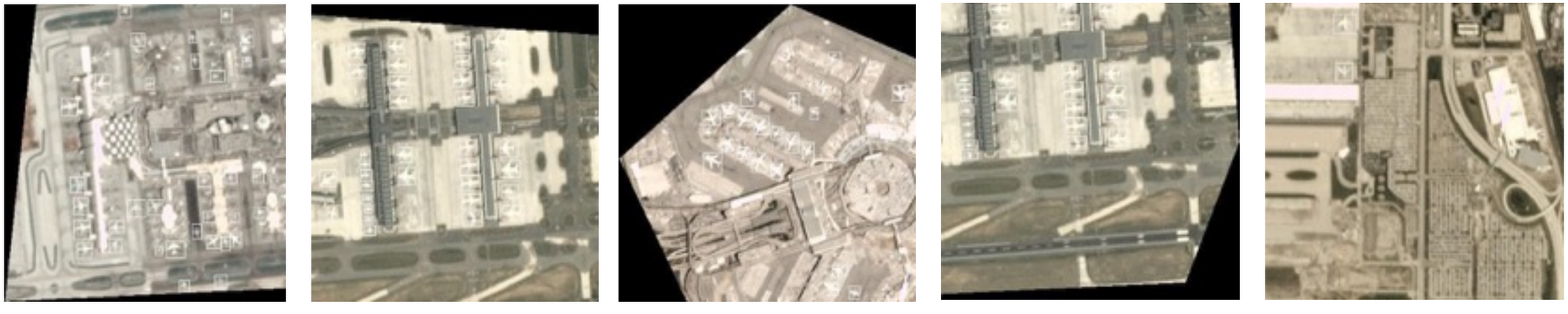
Planet Labs. is releasing a temporal object detection dataset soon!!!

Building a Dataset for Object Detection

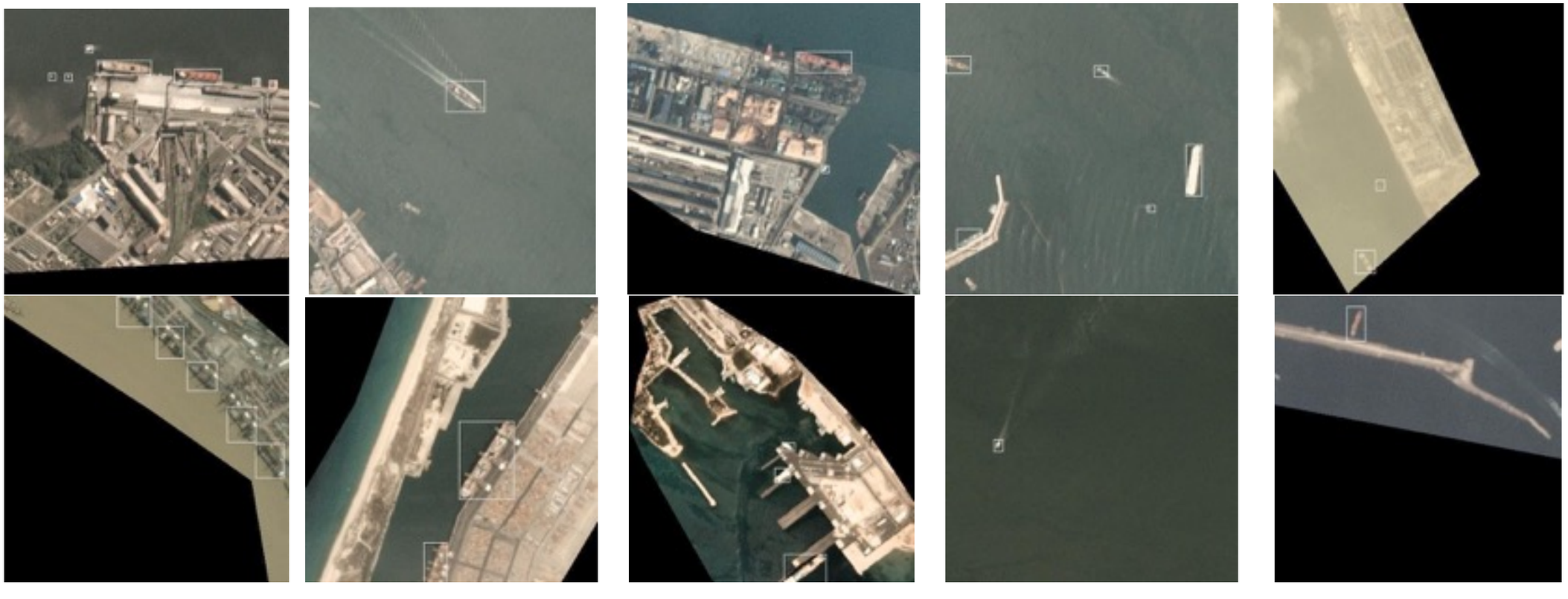
- The images captured in Planet can be used to build a large object detection dataset, removing the need to rely on publicly available datasets.
- In this direction, so far, we used *PlanetScope* images to build
  - Ship Detection Dataset
  - Plane Detection Dataset
- In order to have a general detector, the training dataset should cover the object samples from all around the world.

Port Samples from different parts of the world

Final Datasets



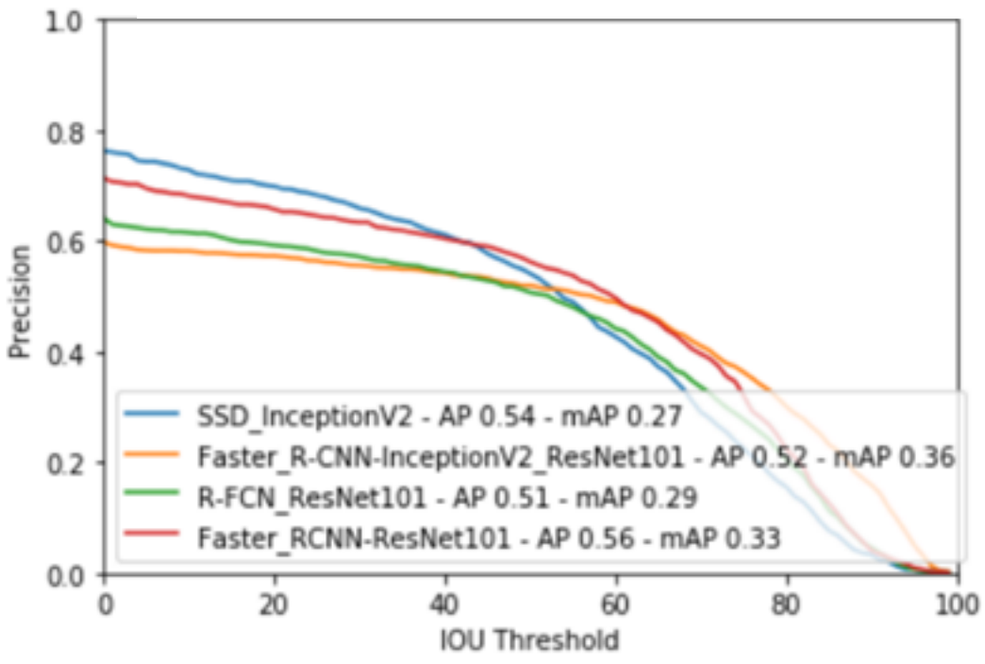
Plane Detection Dataset (2000 Chips and 5000 planes - 150 Airports)

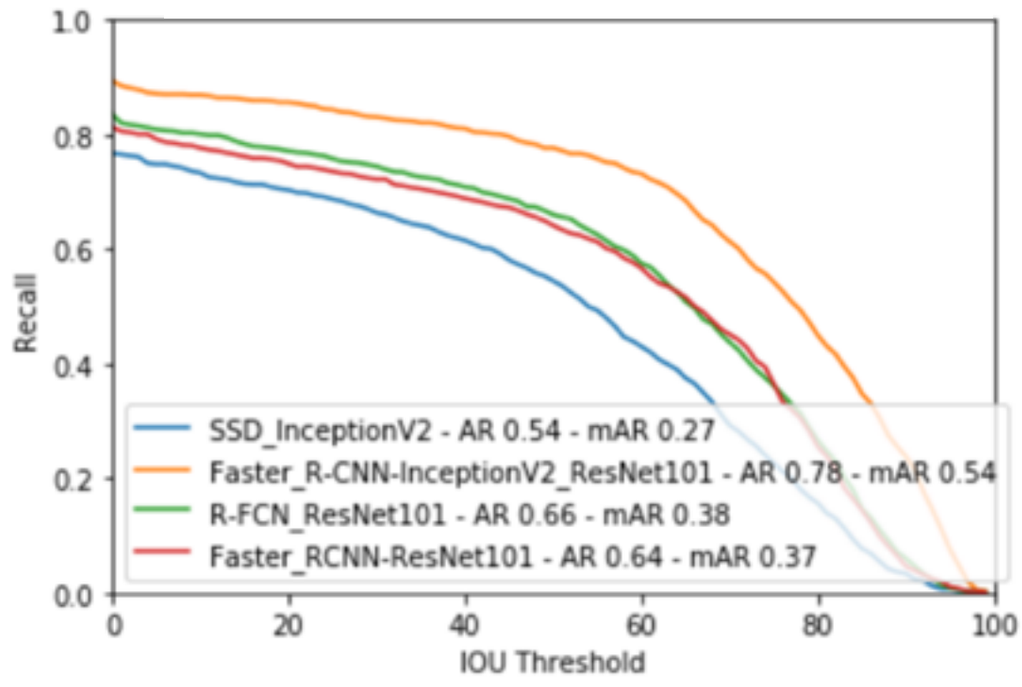


Ship Detection Dataset (3500 chips and 6000 ships - 400 Ports)

Experiments with Convolutional Object Detectors

- There are a number of well-known detection architectures that perform well on the MSCOCO dataset.
  1. Fast-RCNN - **Two Stage Detection**
  2. Faster-RCNN - **Two Stage Detection**
  3. R-FCN - **Two Stage Detection**
  4. SSD - **One Stage Detection**
  5. YOLO2 - **One Stage Detection**
- However, they perform poorly on small objects with **AP** of **%20-%40**.

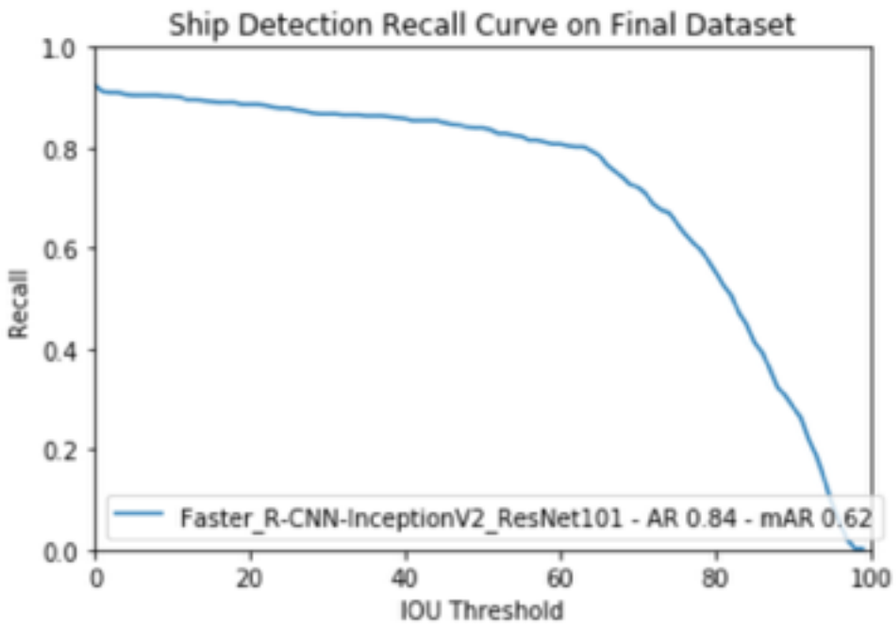




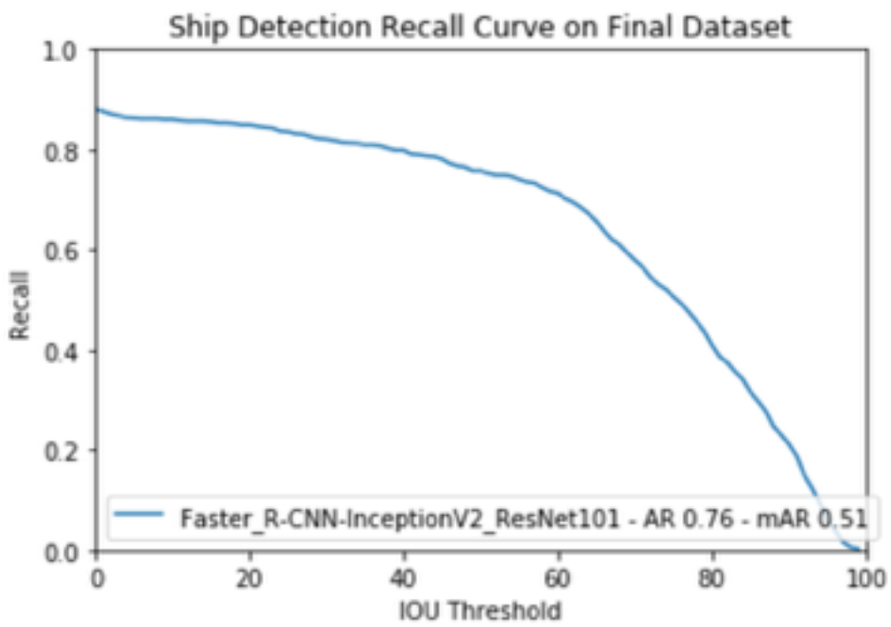
Small Object Detection Problem

Detection Results on MSCOCO Dataset

	AP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR^{1}	AR^{10}	AR^{100}	AR^{S}	AR^{M}	AR^{L}	date
Megvii (Face++)	0.526	0.730	0.585	0.343	0.556	0.660	0.391	0.645	0.689	0.513	0.727	0.827	2017-10-05
UCenter	0.510	0.705	0.558	0.326	0.539	0.648	0.392	0.640	0.678	0.497	0.720	0.829	2017-10-05
MSRA	0.507	0.717	0.566	0.343	0.529	0.627	0.379	0.638	0.690	0.524	0.720	0.824	2017-10-05
FAIR Mask R-CNN	0.503	0.720	0.558	0.328	0.537	0.627	0.380	0.622	0.659	0.485	0.704	0.800	2017-10-05
Trimps-Soushen+QINIU	0.482	0.681	0.534	0.310	0.512	0.610	0.373	0.611	0.652	0.466	0.688	0.801	2017-10-05
bharat_umd	0.482	0.694	0.536	0.312	0.514	0.606	0.365	0.605	0.647	0.456	0.696	0.793	2017-10-05
DANet	0.459	0.676	0.509	0.283	0.483	0.591	0.358	0.587	0.625	0.427	0.664	0.783	2017-10-05
BUPT-Priv	0.435	0.659	0.475	0.251	0.477	0.566	0.337	0.544	0.579	0.366	0.626	0.743	2017-10-05
DL61	0.424	0.633	0.471	0.246	0.458	0.551	0.337	0.563	0.607	0.388	0.654	0.777	2017-10-05
DeNet	0.424	0.615	0.455	0.222	0.463	0.584	0.347	0.551	0.582	0.371	0.623	0.756	2017-10-05
IL	0.420	0.633	0.460	0.234	0.452	0.554	0.338	0.552	0.590	0.380	0.635	0.757	2017-10-05

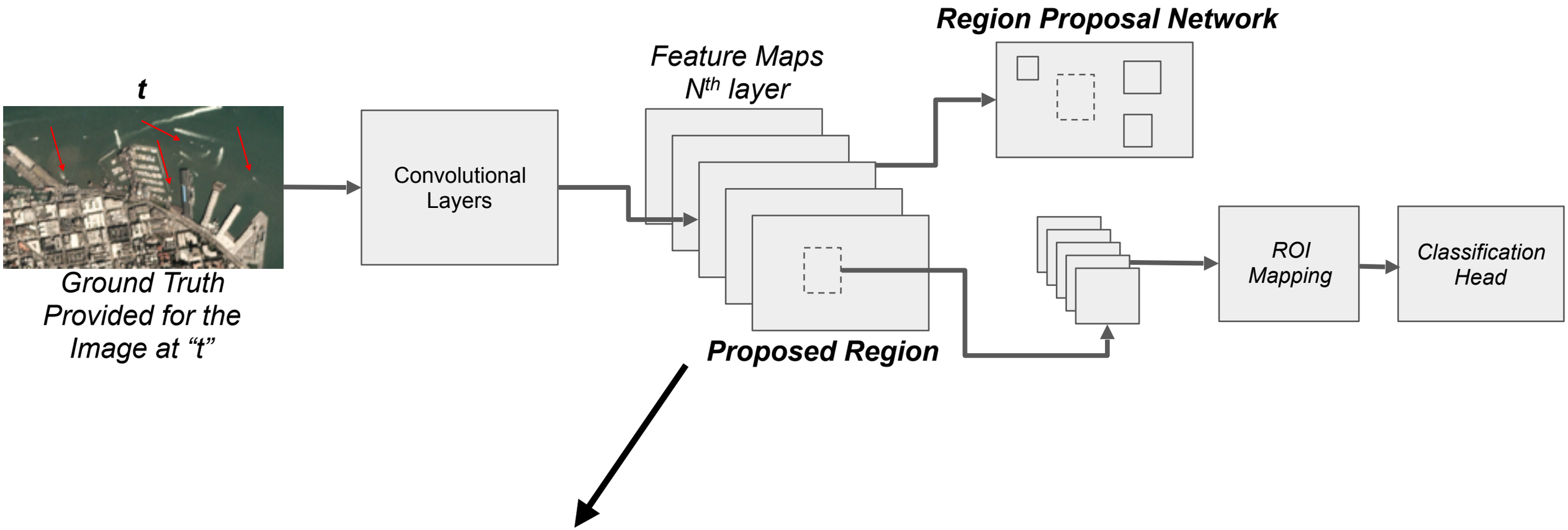


Ships > 100 m



Ships < 100 m

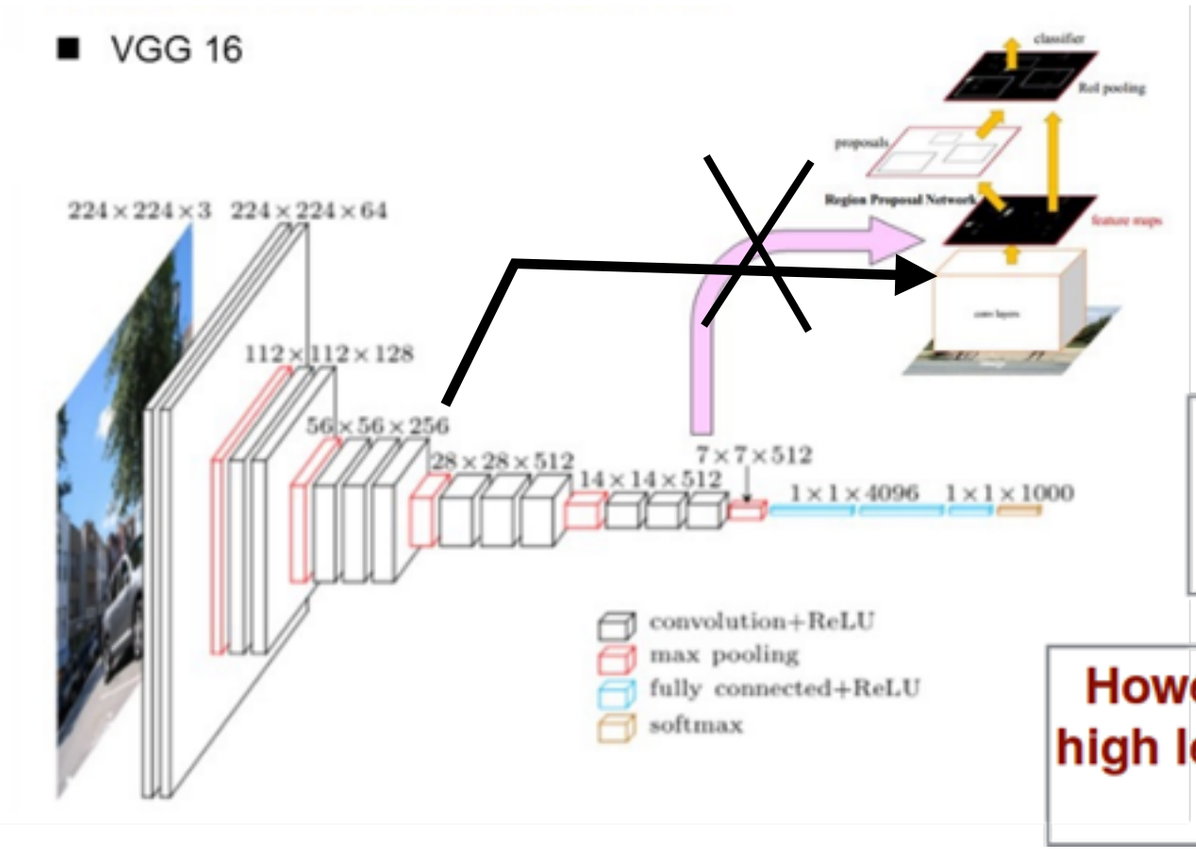
Current Detection Framework - Faster RCNN



***A pixel in the features maps used by RPN correspond to larger number of pixels in the input image.**

Related Work on Small Object Detection

- VGG16 has the stride of 16 which increases the number of pixels covered by the feature map.
- There is a number of techniques to improve small object detection with Faster-RCNN.

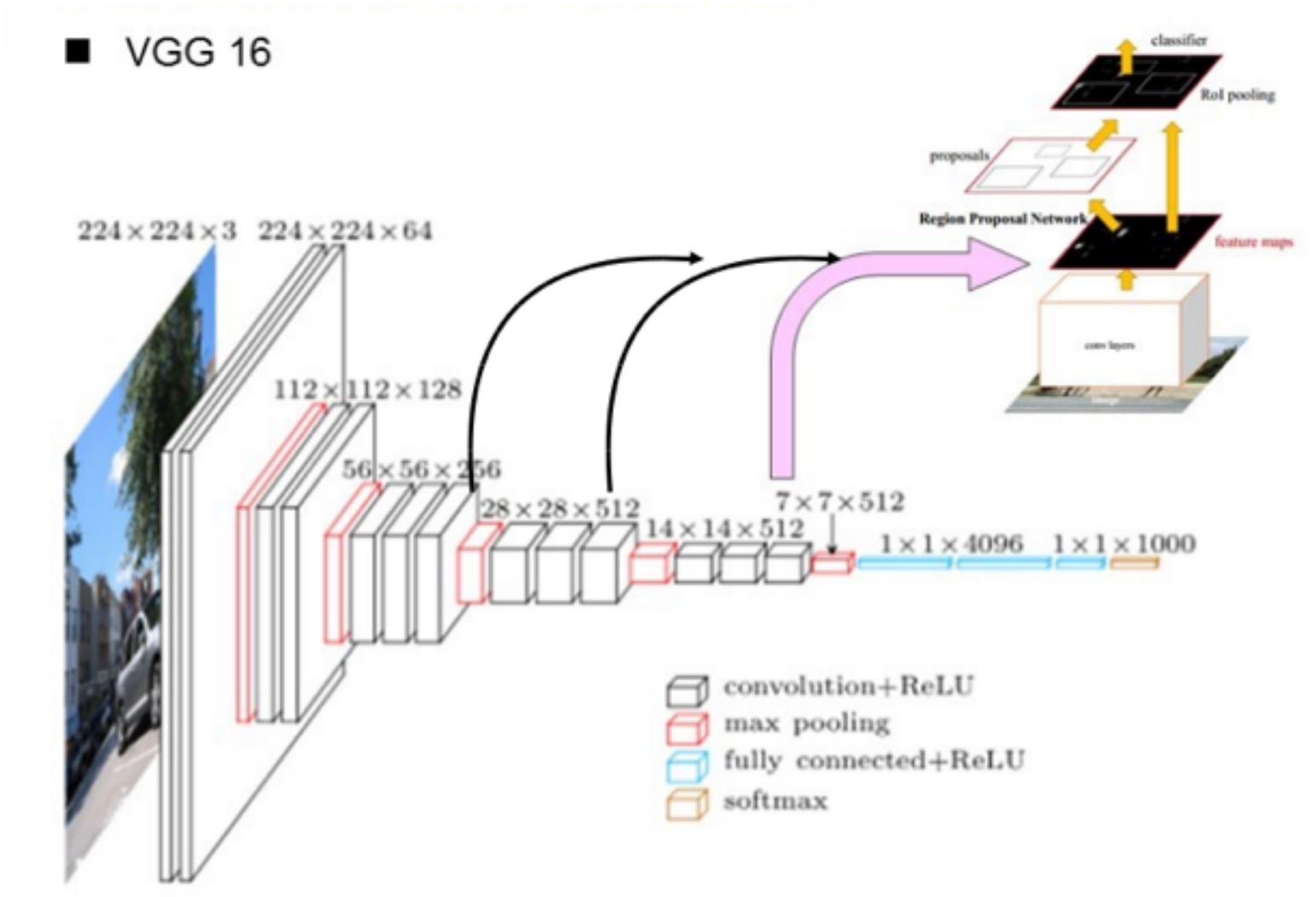


Use earlier layers features maps in the RPN

However, it does not provide high level feature encodings to RPN

Related Work on Small Object Detection

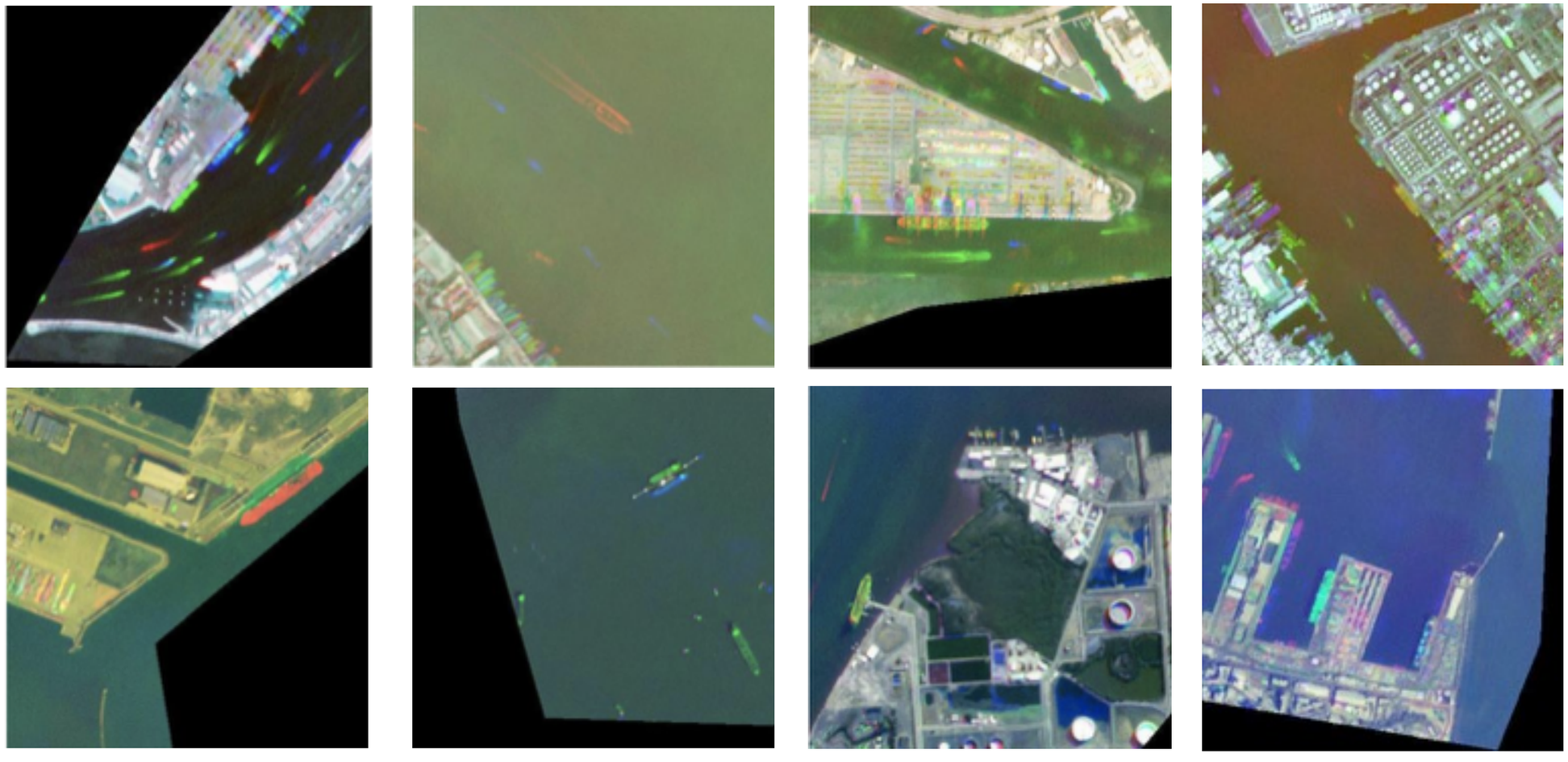
- Other methods include concatenating multiple layer features for the RPN.



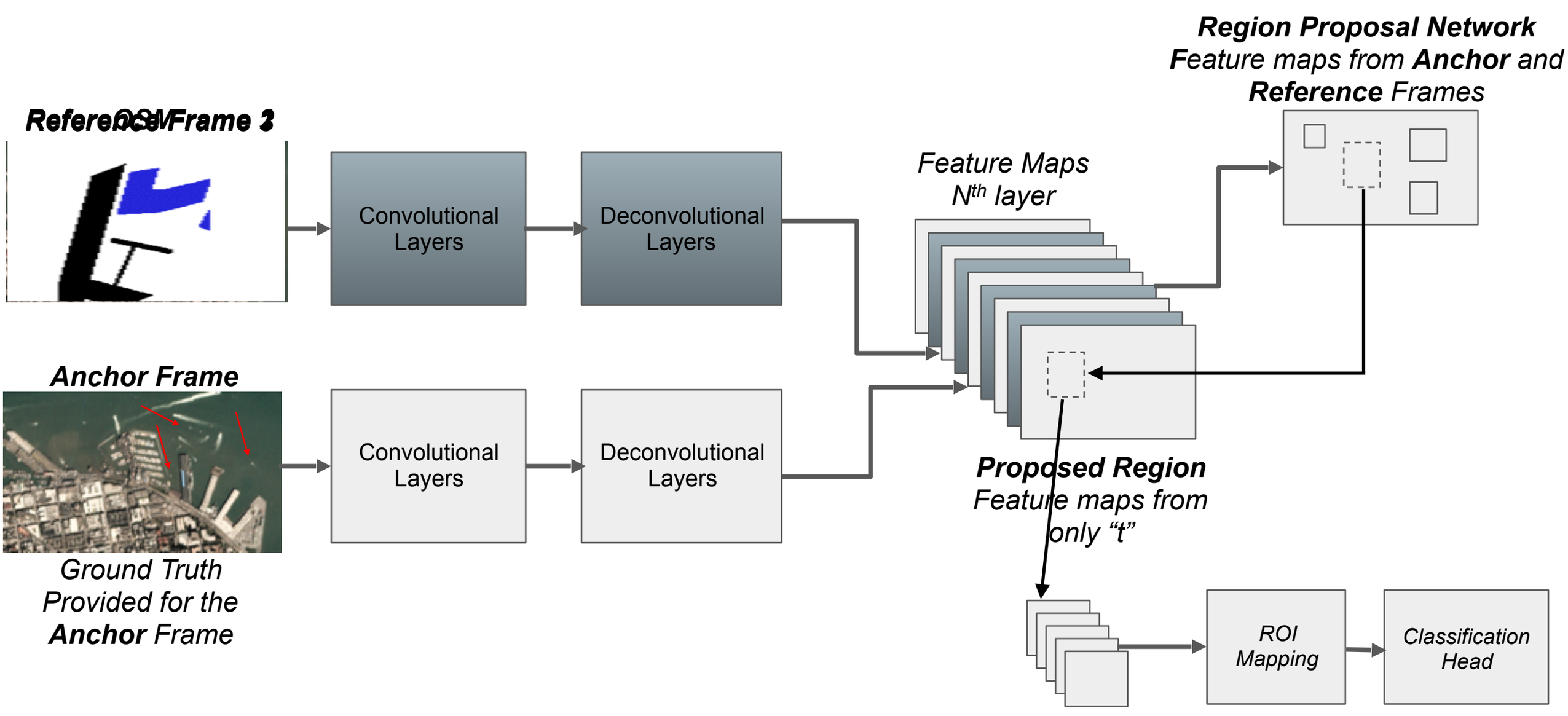
Temporal Data for Small Object Detection

- Object detection using *only spatial information* is challenging in low resolution satellite images.
- To tackle this, one can perform *spatiotemporal object detection*.

Temporal PlanetScope Images (3.0 m GSD)



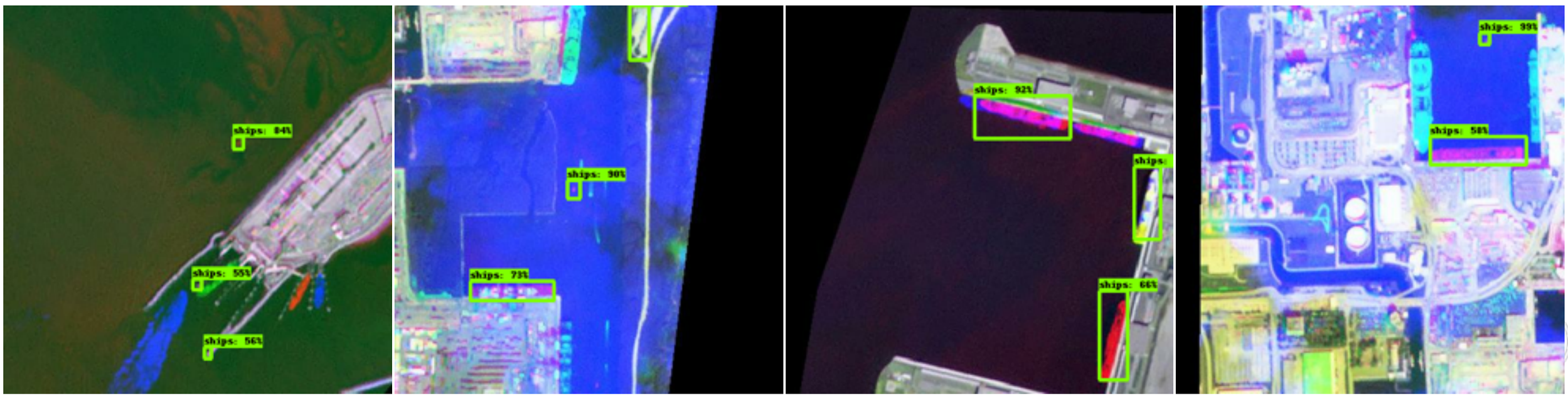
Proposed SpatioTemporal Faster R-CNN



Preliminary Results

Model	AP	mAP	AP^S	AP^M	AP^L
Spatial Faster-RCNN	52.15	36.23	30.45	45.11	58.22
SpatioTemporal Faster-RCNN	65.32	45.78	45.22	58.68	72.91

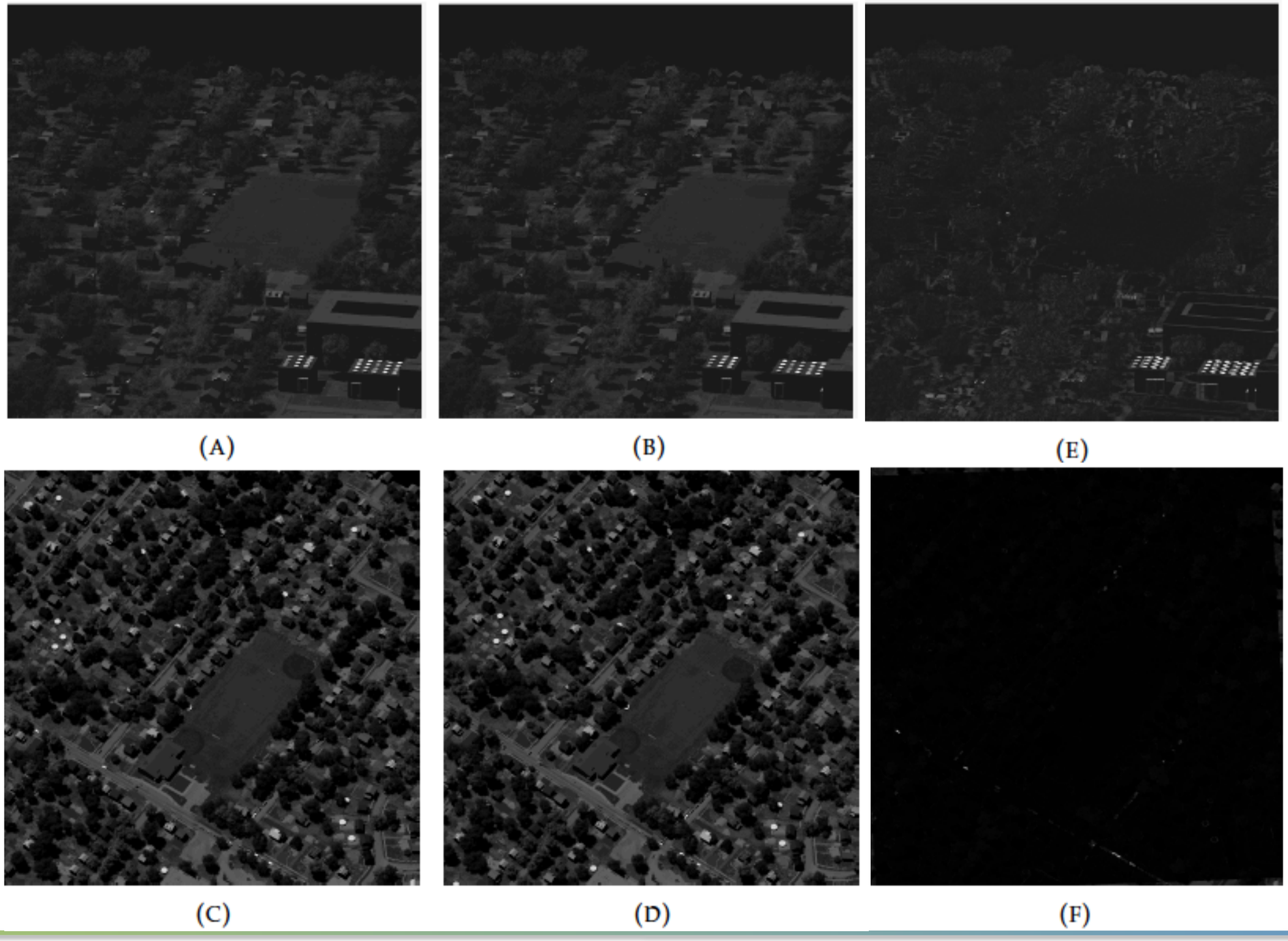
Performance on the Object Detection Test Set



Visual Results on the Test Set

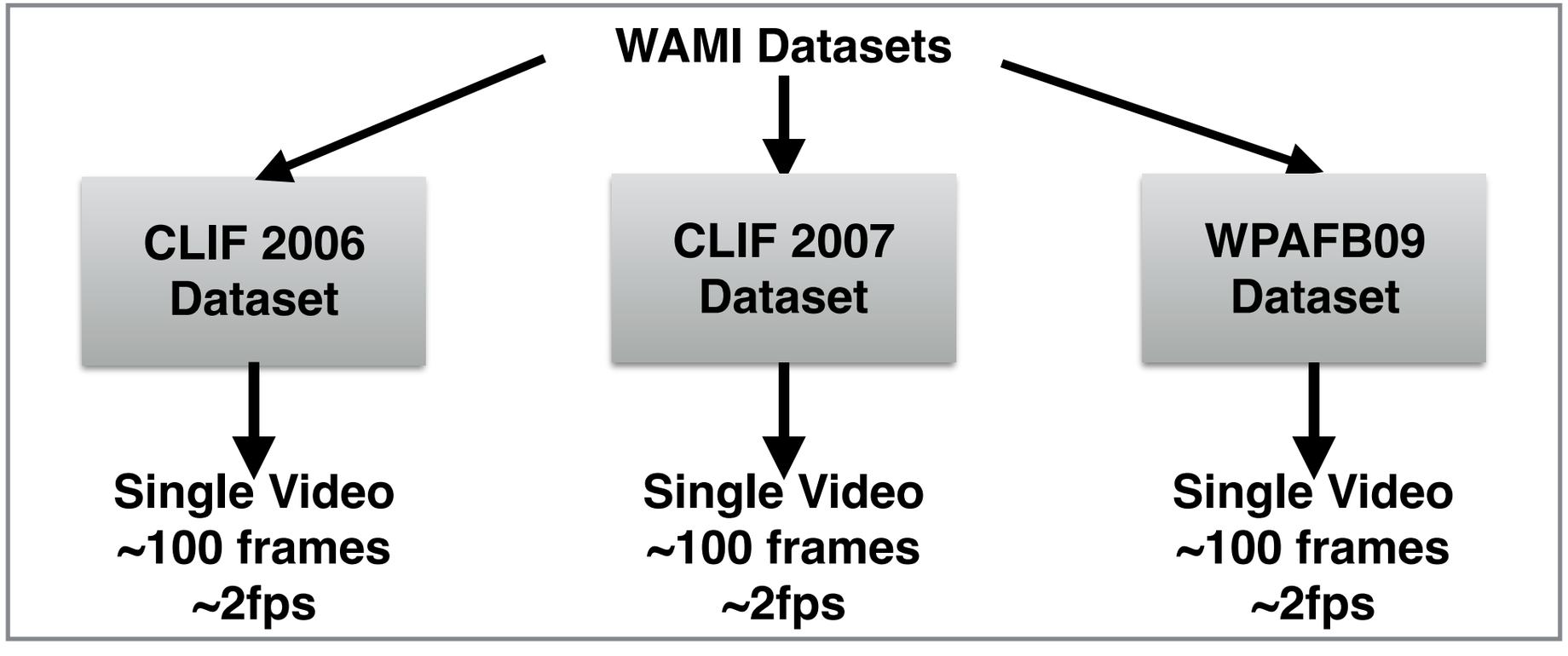
THANK YOU!

Parallax Effect on Registration



Vehicle Classification on the WAMI Platform

- We build a synthetic single-channel vehicle classification dataset for the following goals.
    1. Achieve high classification accuracy on the WAMI platform without using any WAMI training data.
    2. Prove the high-fidelity of synthetic aerial hyperspectral videos used to test the proposed trackers.



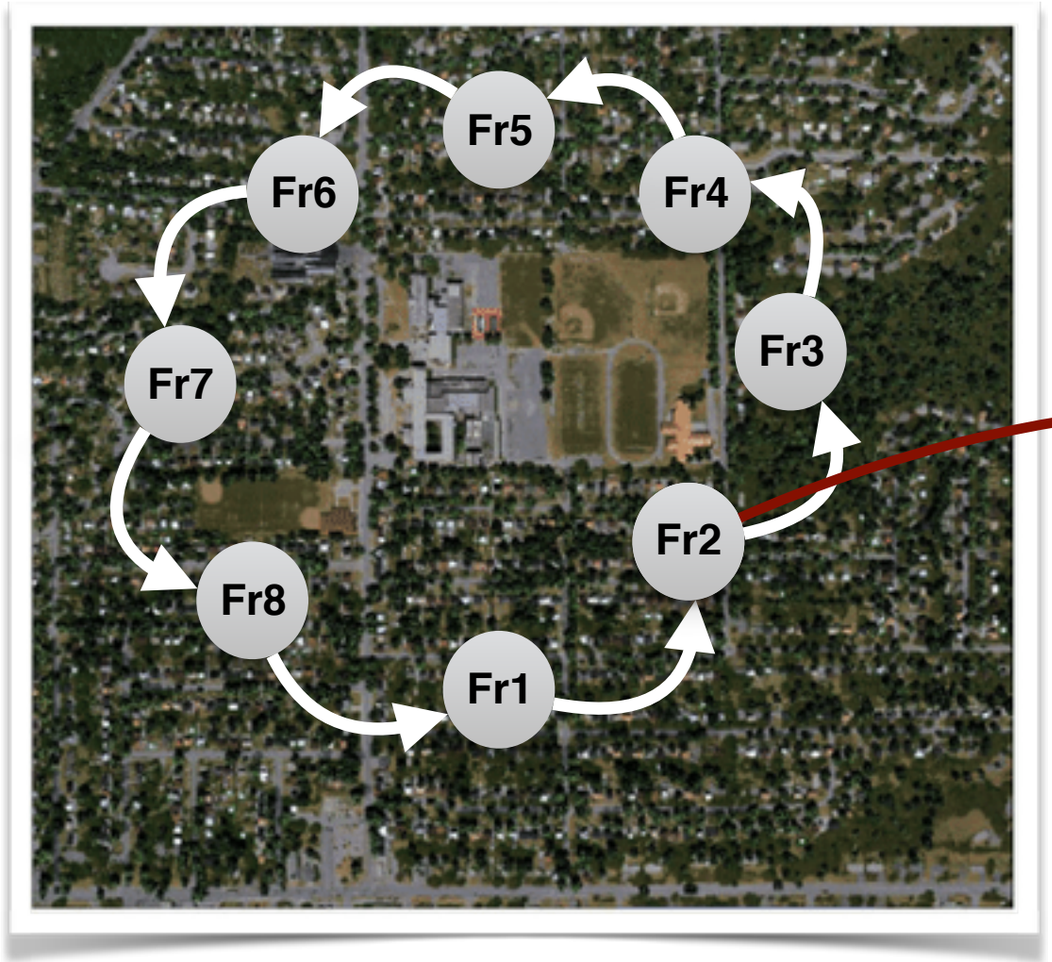
Building the Synthetic Dataset

- We match the ground sampling distance to the WAMI data (0.3m) where vehicles are represented by 100-200 pixels.

Part of Northern Rochester NY

Left - *DIRSIG* **Right** - *Google Maps*

Temporal Variance



Part of Northern Rochester NY

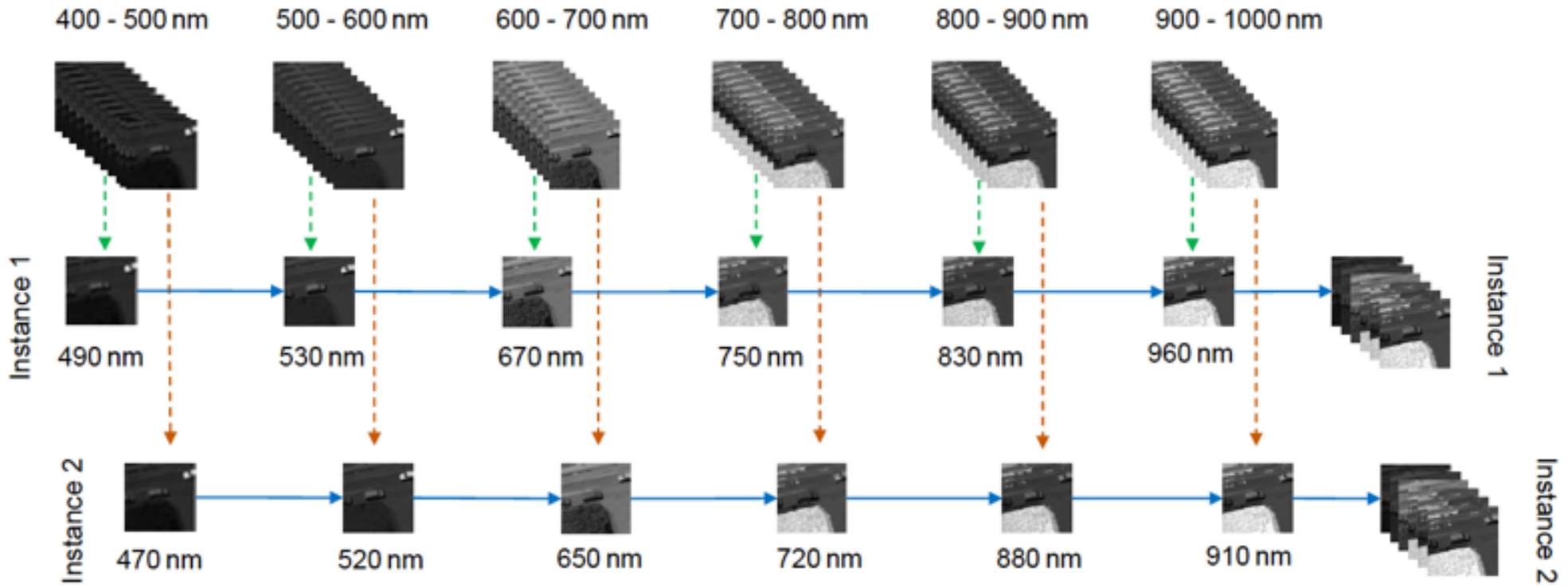
2016
January

- - -

2017
January

Hyperspectral Data Augmentation

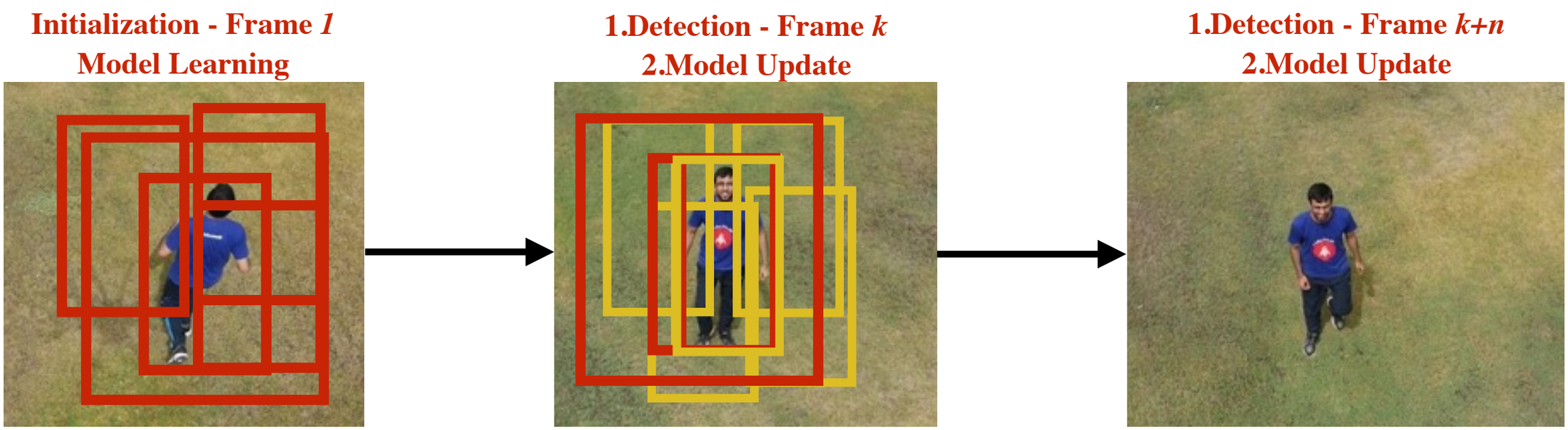
- Since we do not fully know the internal mechanics of the WAMI platform, it is ideal to use samples of the same image captured at different wavelengths.



Hyperspectral Data Augmentation

- This way, we sample 12 images from the same sample.

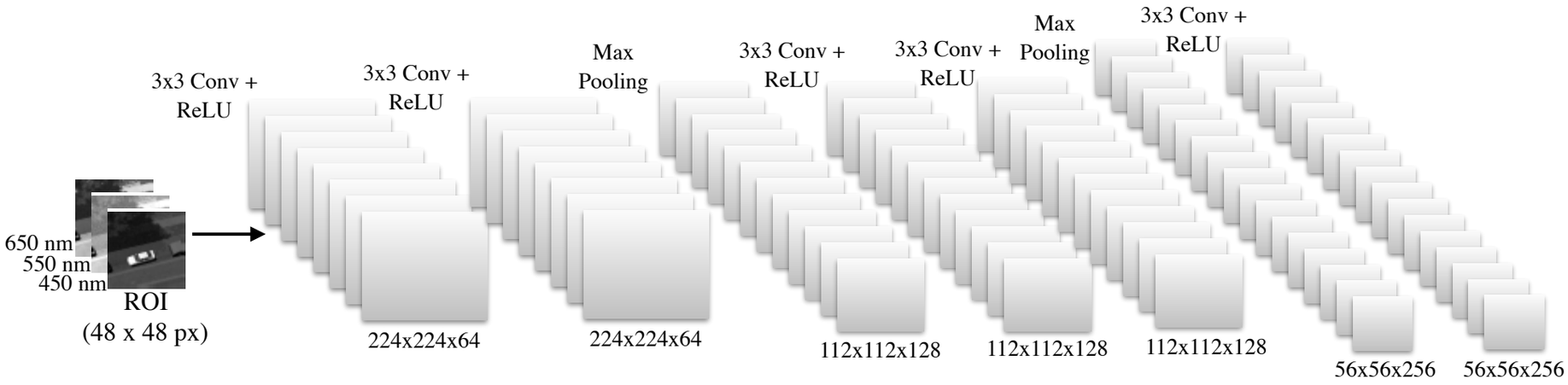
Introduction



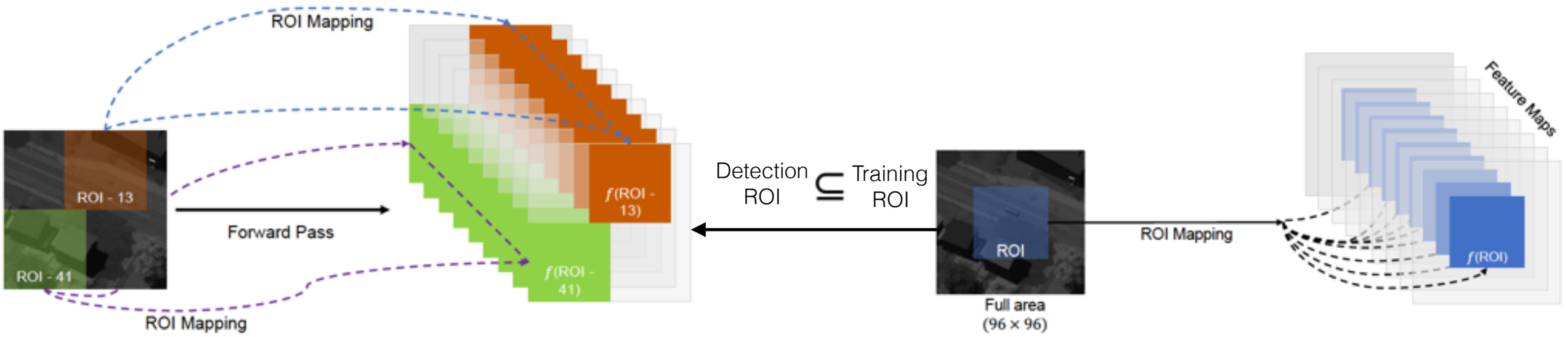
Tracking-by-detection Framework

- MIL, Struct, KCF, CCOT and ECO are some of the *tracking-by-detection* algorithms.
- In comparison to deep-learning trackers, their advantages can be listed as
    1. No offline training
    2. Small memory footprint
    3. No need to have GPUs installed on an embedded system.

Deep Object Representation



Traditional Feature Extraction



Fast Feature Extraction