Object Detection/Tracking as a “Black Box”

Instructor - Simon Lucey

16-423 - Designing Computer Vision Apps

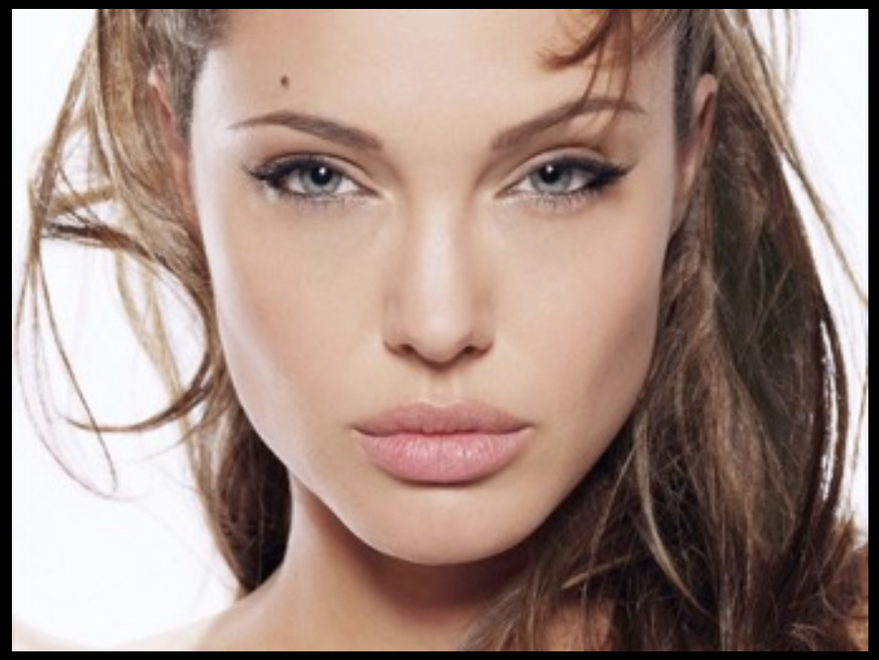
"The Virtually Impossible"

Today

- Motivation - Object Detection/Tracking
- Computer Vision as a “Black Box” - Considerations
- Detection
  - Computer Vision as a Service (VMX, Project Oxford, Clarifai).
  - OpenCV 3.0 (face and pedestrian detectors, what’s new?)
  - DLib C++ (create your own detector!!!)
  - Caffe (Deep Learning)
- Tracking
  - Correlation Filters (fast in tracking in just a few lines of code)
  - Predator (tracking an object efficiently/quickly)

Just a sampling - by no means a complete list!!!

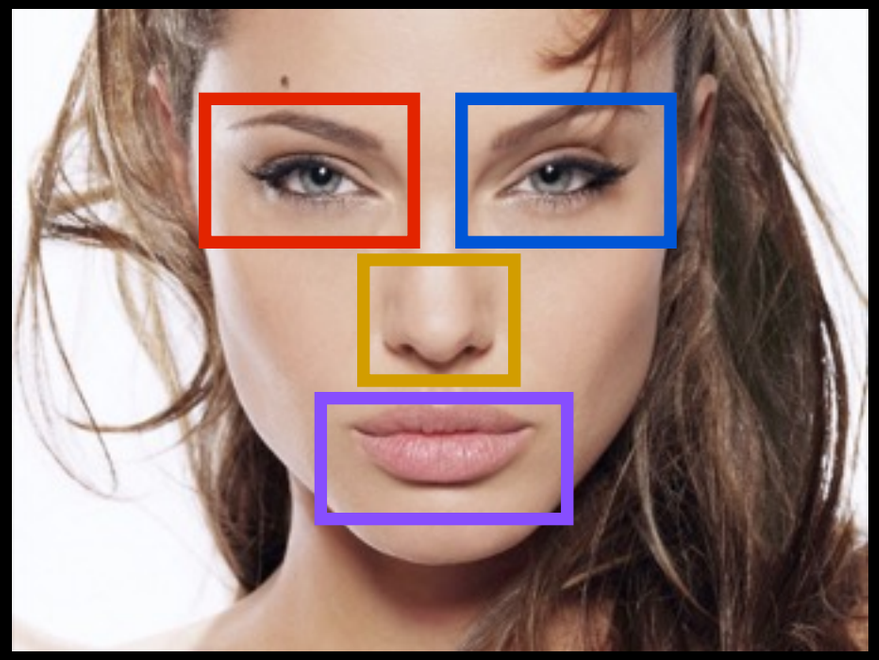
What is an Object?



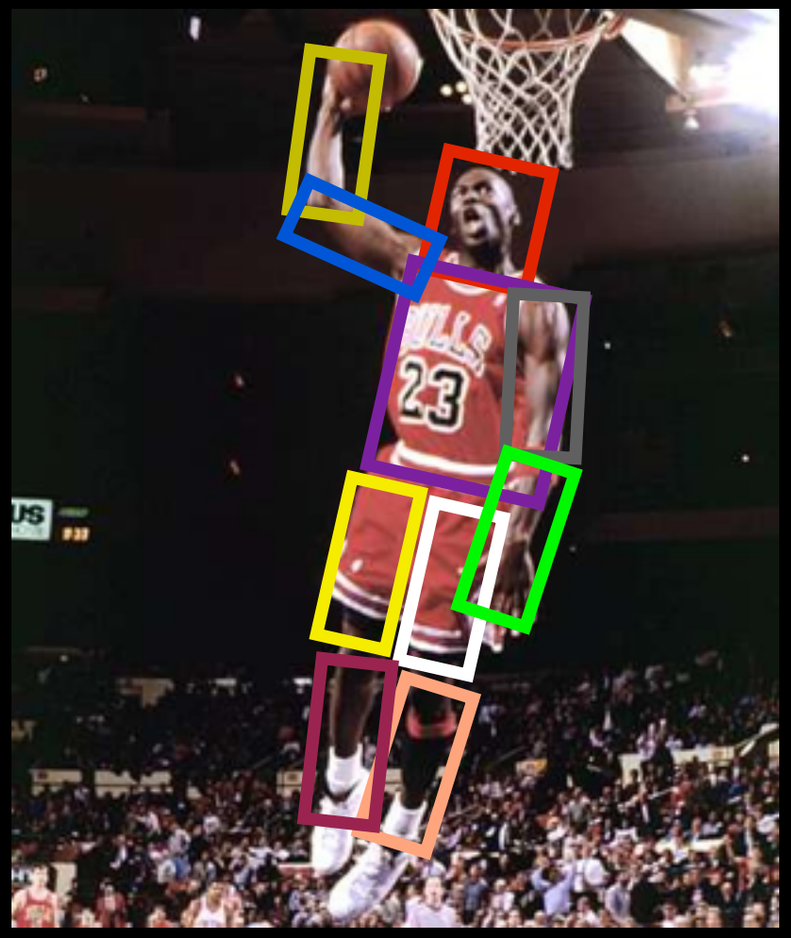
Face

Body

What is an Object?

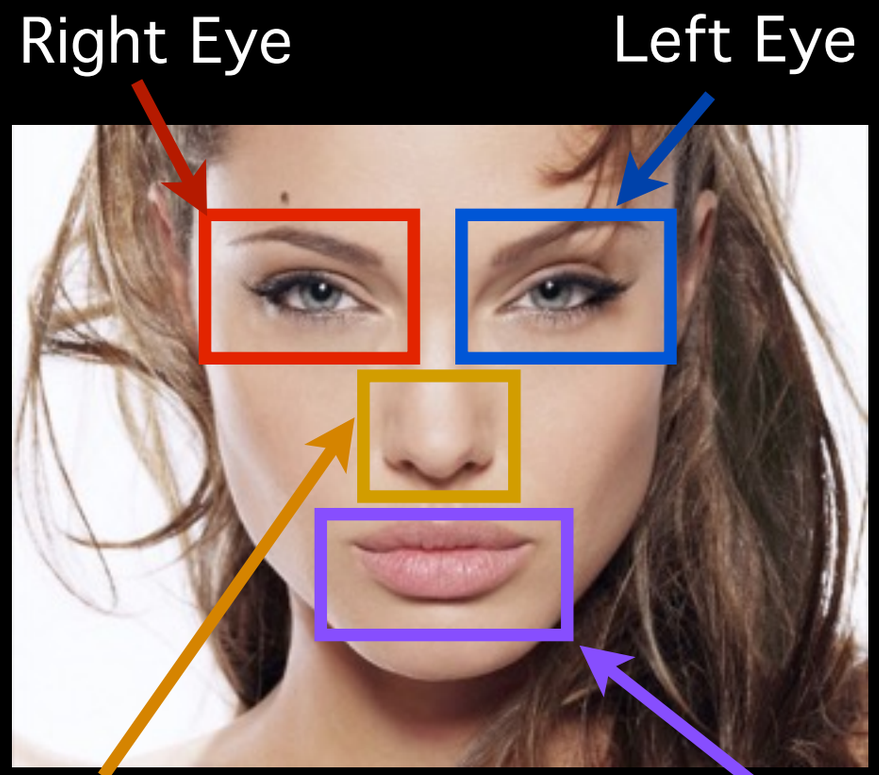


Face

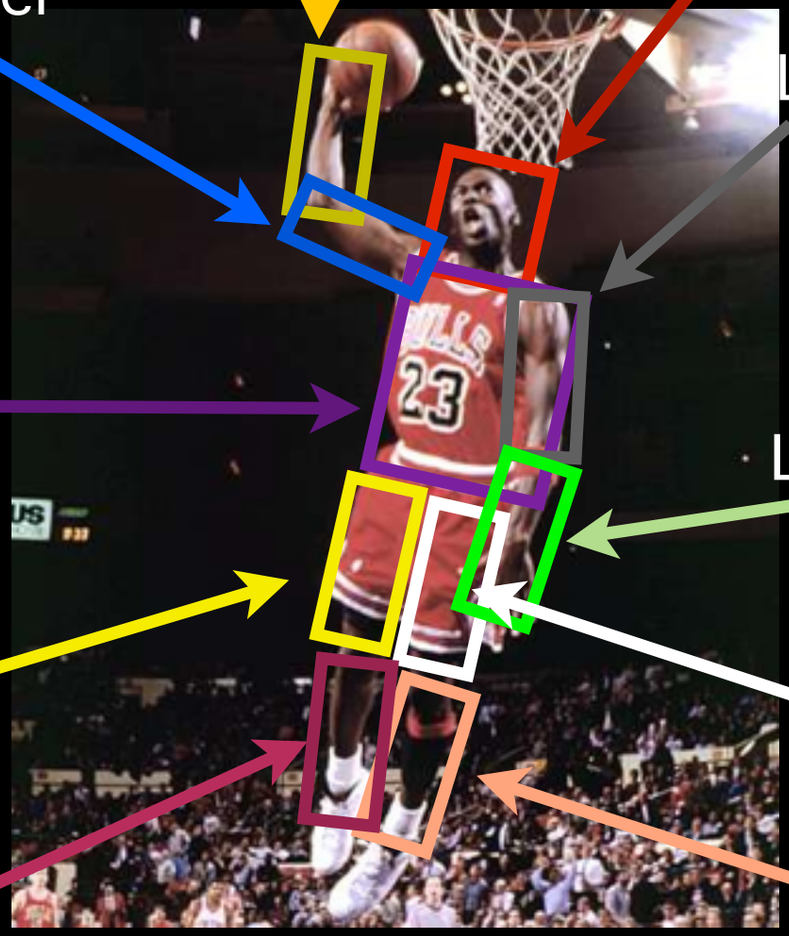


Body

What is an Object?

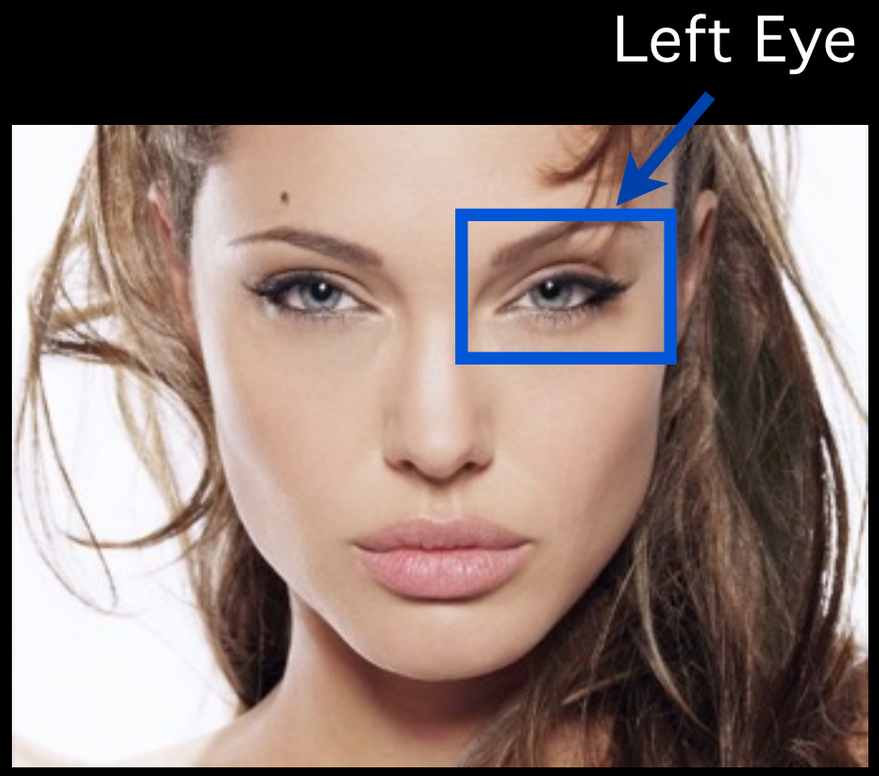


Face



Body

What is a Part?

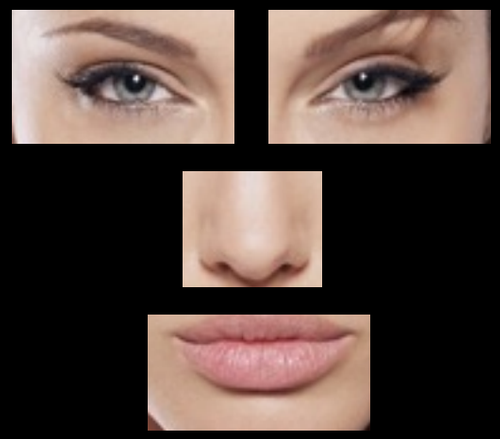


Face

Not Left Eye

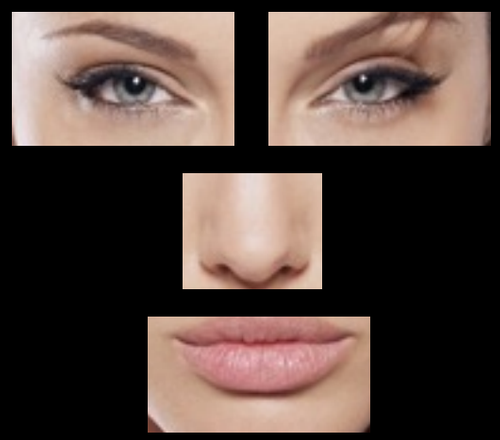
When is a Collection of Parts an Object?

When is a Collection of Parts an Object?

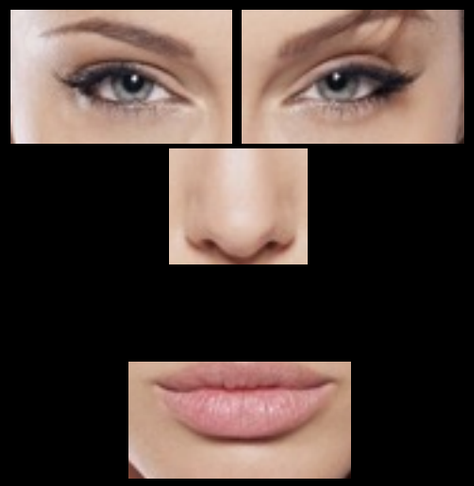


Face

When is a Collection of Parts an Object?

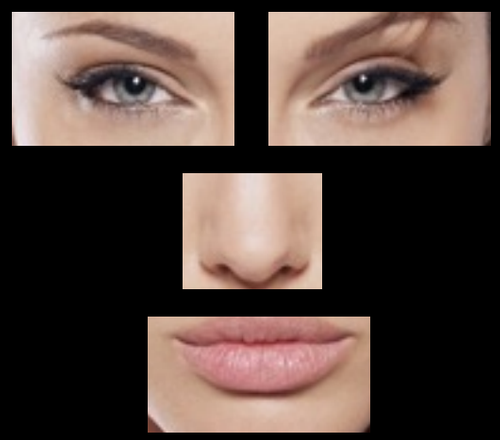


Face

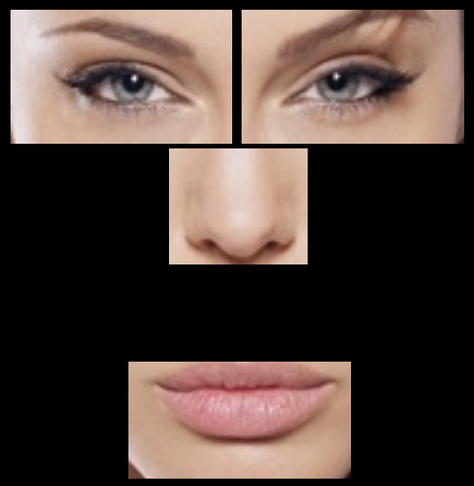


Face

When is a Collection of Parts an Object?



Face



Face

Not a Face

What is an Object?

"A collection of semantically meaningful components with geometrical constraints on their spatial configuration"

Data

Best Method is Domain Specific

No Silver Bullet

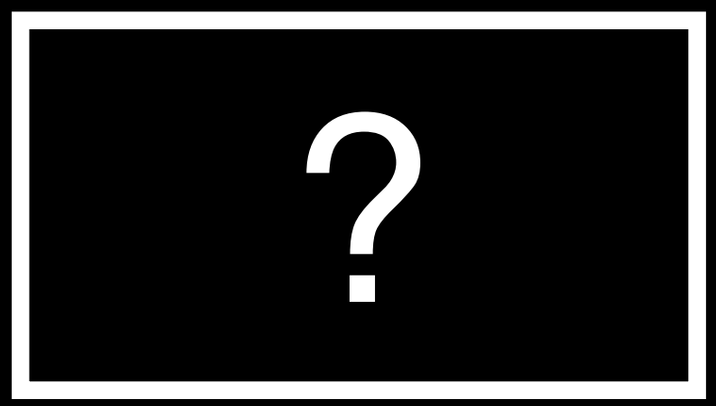
Today

- Motivation - Object Detection/Tracking
- Computer Vision as a “Black Box” - Considerations
- Detection
  - Computer Vision as a Service (VMX, Project Oxford, Clarifai).
  - OpenCV 3.0 (face and pedestrian detectors, what’s new?)
  - `DLib` C++ (create your own detector!!!)
  - Caffe (Deep Learning)
- Tracking
  - Correlation Filters (fast in tracking in just a few lines of code)
  - Predator (tracking an object efficiently/quickly)

Just a sampling - by no means a complete list!!!

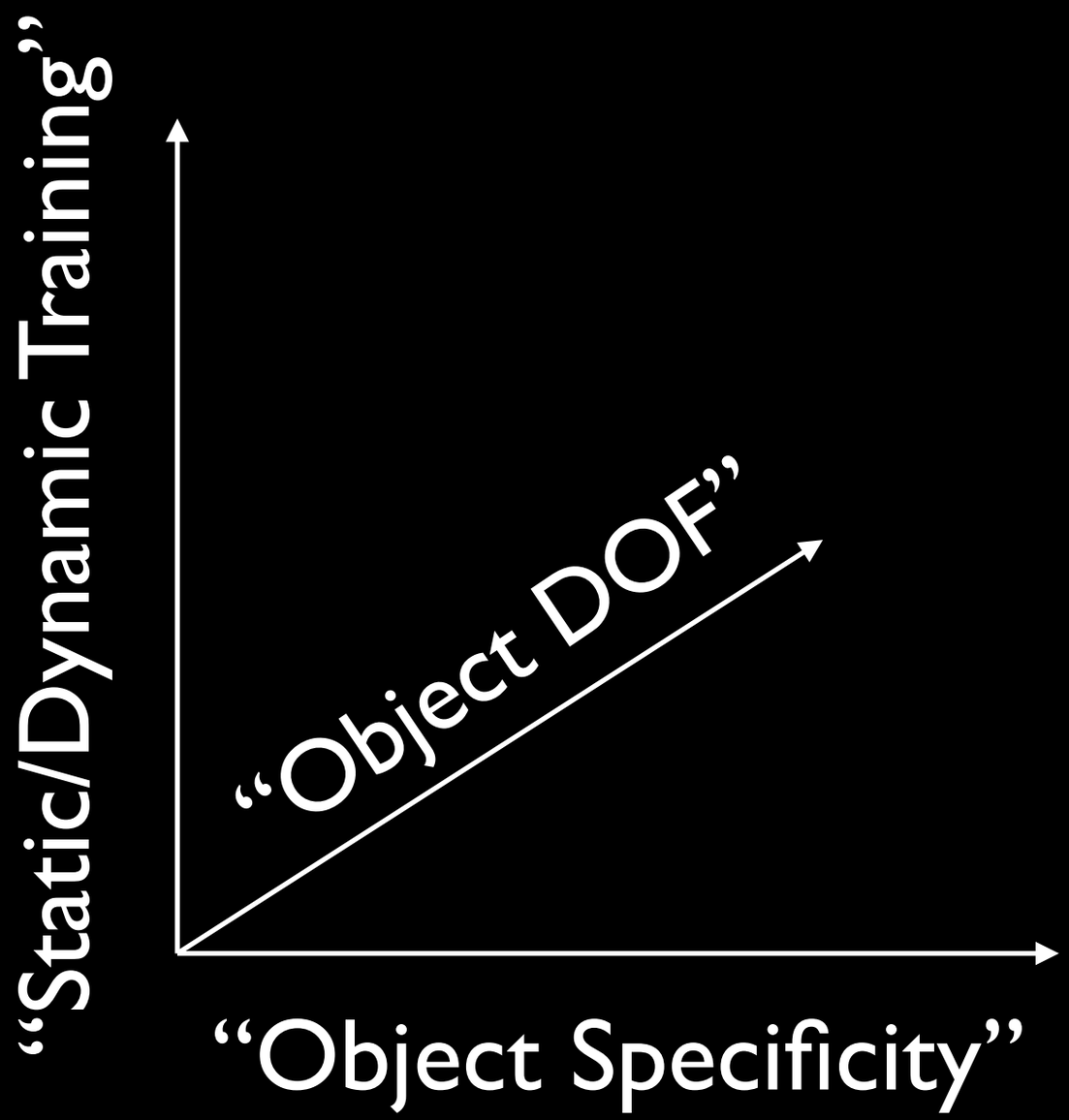
Computer Vision as a “Black Box”

- Increasingly, people want to employ computer vision in a variety of new applications.
- Want to use Computer Vision as a “Black Box” - i.e. do not care about how it works as along as it does what it is supposed to do.
- Your ability to treat computer vision as a “Black Box” is a function of what you want to do.
- In this lecture, we will try and give a brief overview of the this space.



Solutions Exist

- You need to consider where your problem/task lies.



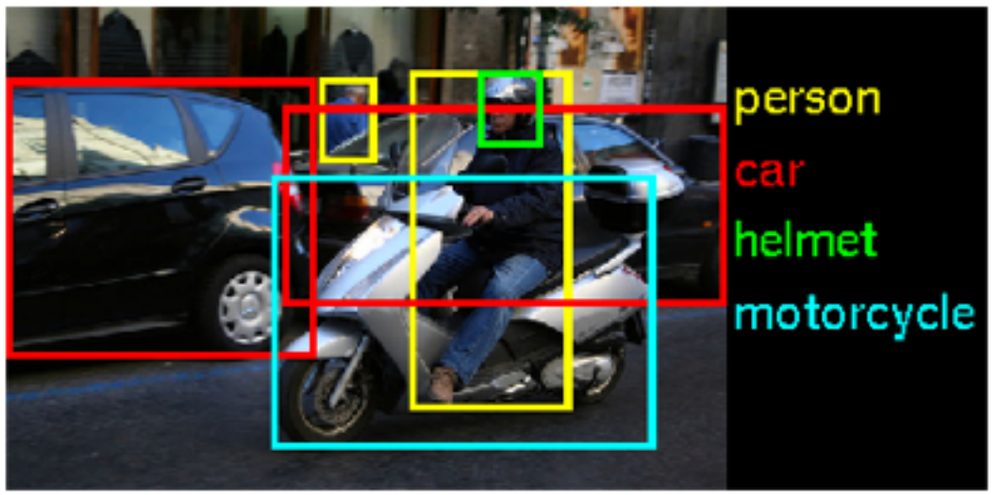
?

Object Specificity

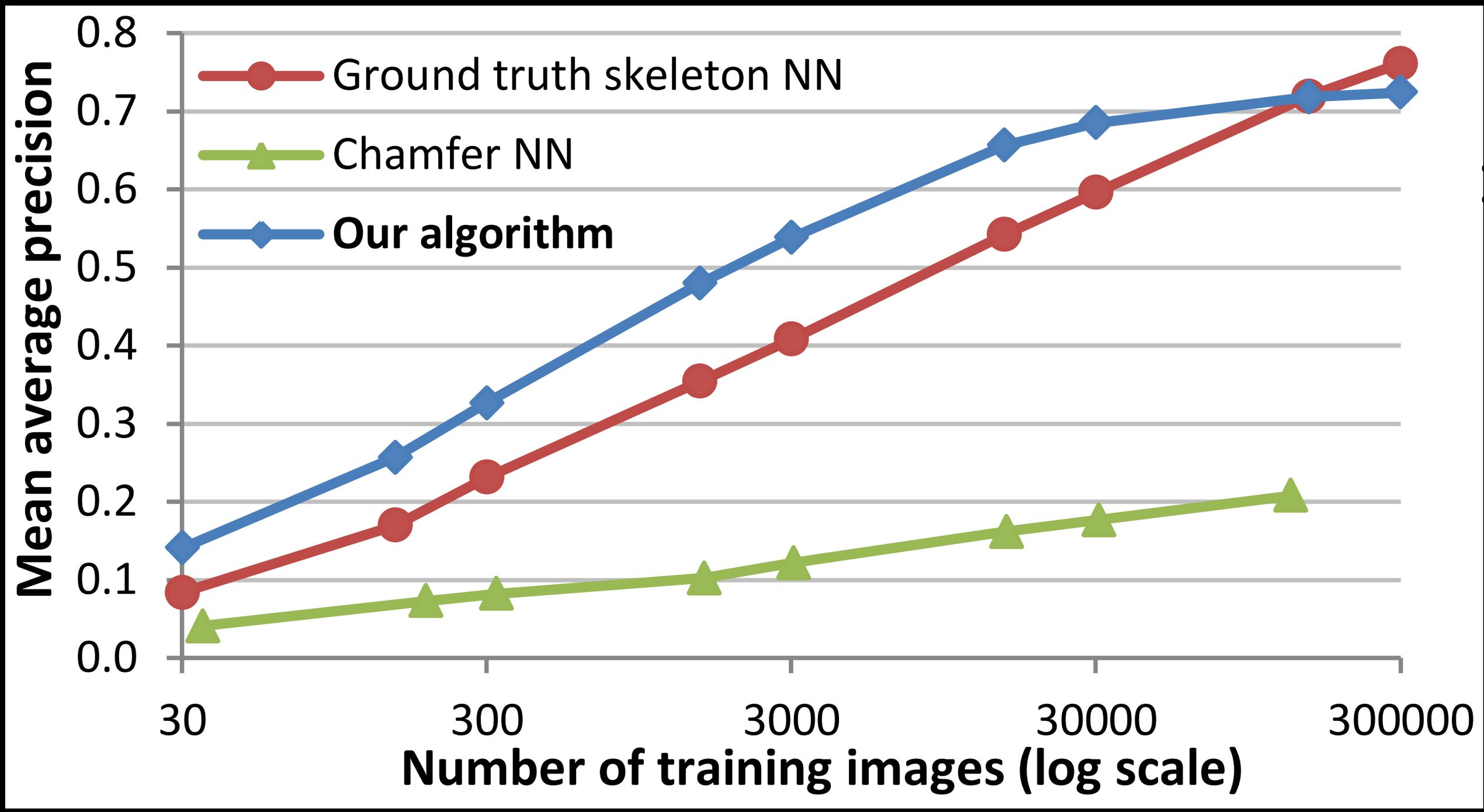
- Faces (well established - mature)

- Bodies (becoming more mature)

- General Objects (emerging)

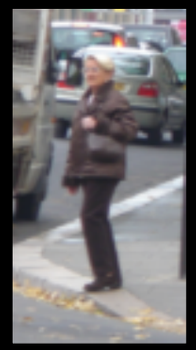


“Big Data Effect”



Taken from - Shotton et al. “Real-Time Human Pose Recognition in Parts from Single Depth Images” CVPR 2011.



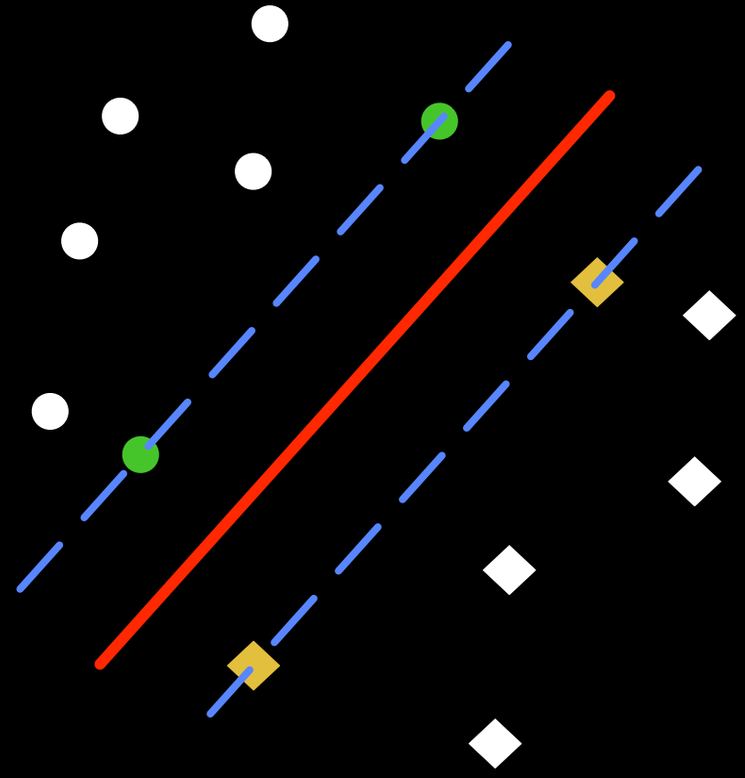


200 West


200 West

Hard Negative Mining

- Hard Negative Mining comes about from realization that not ALL samples are important when learning a classifier.
- Useful for “Big Data” as one does not need to keep all data in memory during learning.



Hard Negative Mining

- Common methodology (Dalal & Triggs, 2005) is to,
- Start with random negatives, then repeat
  1. Train model
  2. Harvest false positive to define “hard negatives”.
- HNM is largely based on heuristics.
- Notoriously slow and messy to determine the hard negatives.

Static/Dynamic Training

- We can categorize object detection methods into two camps
  1. **static training** - involves learning a detector from a dataset that is static and does not change from application to application (e.g. face detection).
  2. **dynamic training** - involves using your own dataset for the object you are wanting to detect.
- Dynamic training is also useful in object tracking - where one wants to employ a “track by update” paradigm.

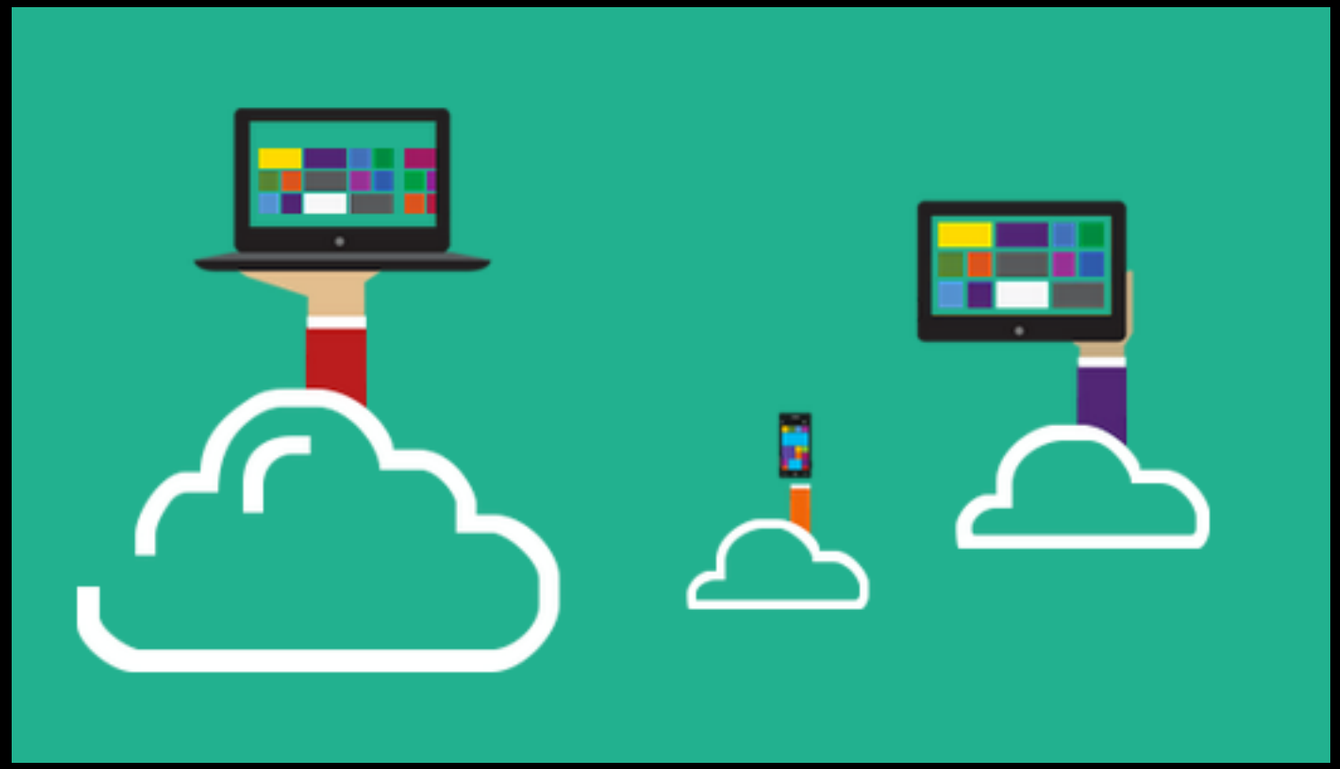
Today

- Motivation - Object Detection/Tracking
- Computer Vision as a “Black Box” - Considerations
- Detection
  - Computer Vision as a Service (VMX, Project Oxford, Clarifai).
  - OpenCV 3.0 (face and pedestrian detectors, what’s new?)
  - DLib C++ (create your own detector!!!)
  - Caffe (Deep Learning)
- Tracking
  - Correlation Filters (fast in tracking in just a few lines of code)
  - Predator (tracking an object efficiently/quickly)

Just a sampling - by no means a complete list!!!

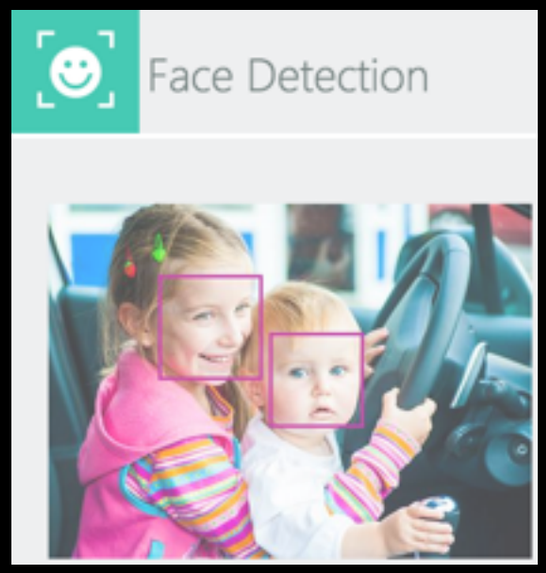
Computer Vision as a Service

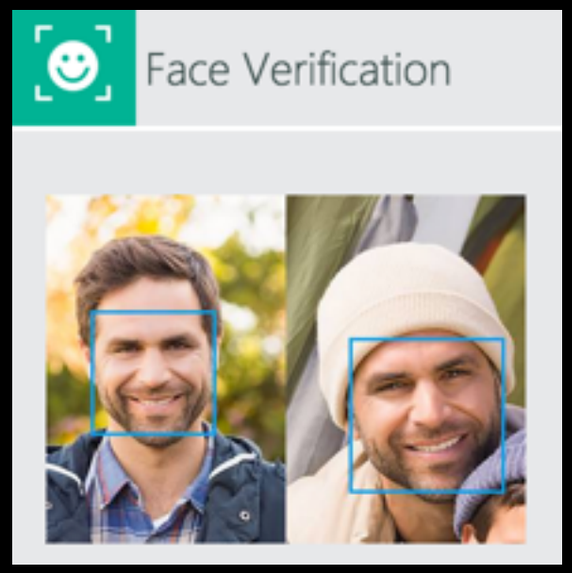
- Provide computer vision services through the cloud.
- Increasingly popular as developers do not need any intimate knowledge of computer vision.
- Pros - automatically support multiple platforms.
- Drawbacks - have to consider bandwidth & speed.

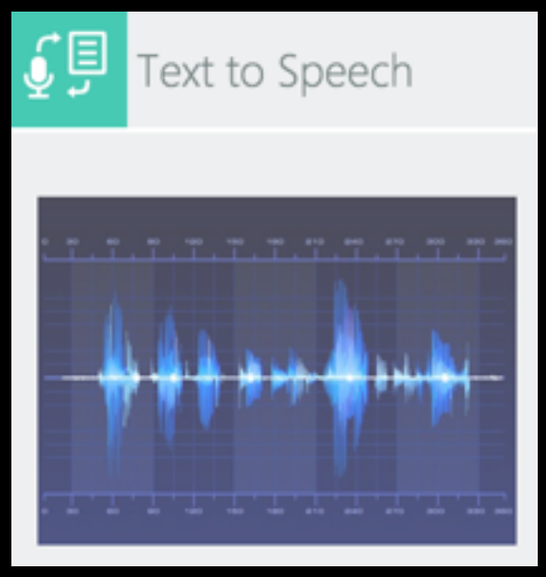


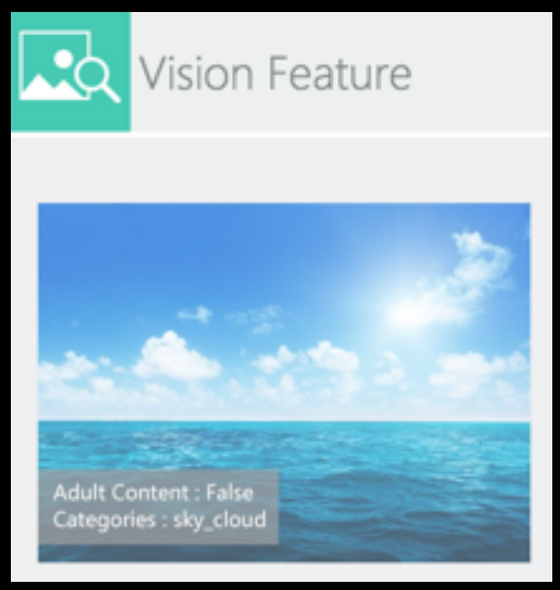
Project Oxford

- Well known in this space is "Project Oxford" from Microsoft.
- Contains an evolving portfolio of REST APIs & SDKs enabling developers to easily add intelligent services.
- Other services include - VMX & Clarifai.









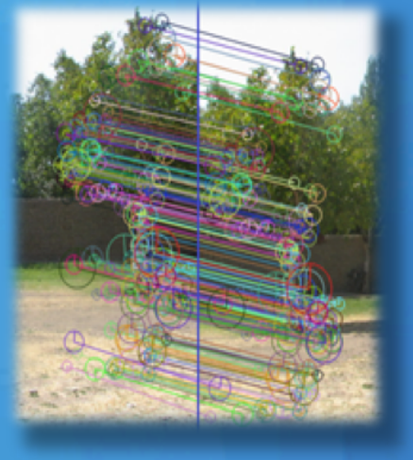
What can OpenCV do?

Image Processing

Filters

Transformations

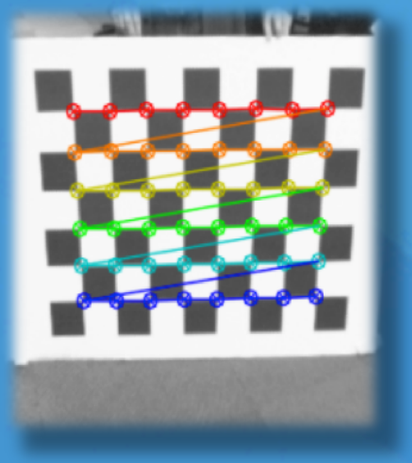
Edges, contours



Robust features

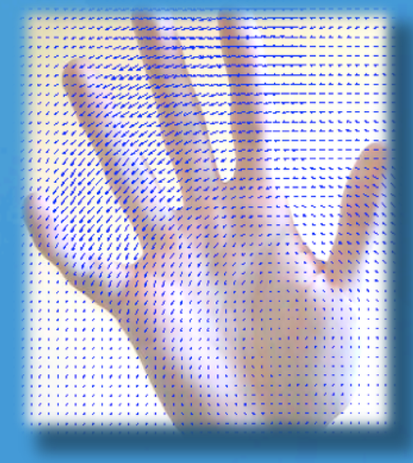
Segmentation

Video, Stereo, 3D

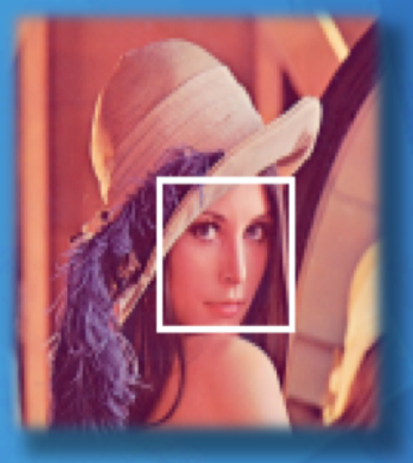


Calibration

Pose estimation



Optical Flow



Detection and recognition

Depth

itseez

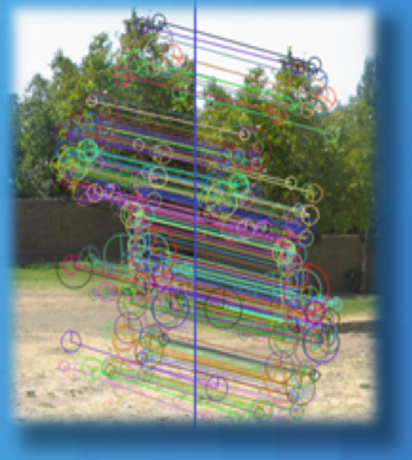
What can OpenCV do?

Image Processing

Filters

Transformations

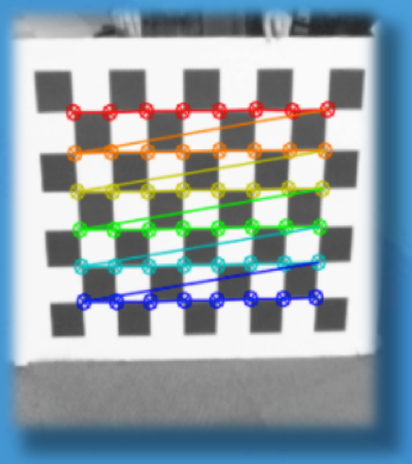
Edges, contours



Robust features

Segmentation

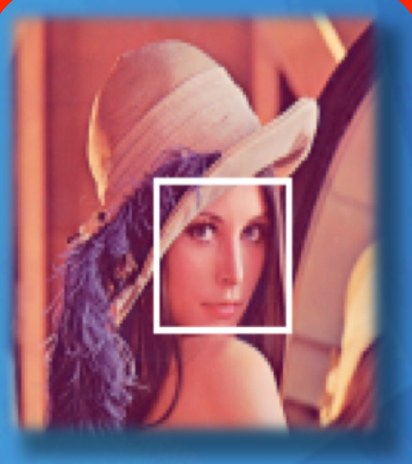
Video, Stereo, 3D



Calibration

Pose estimation

Optical Flow



Detection and recognition

Depth

itseez

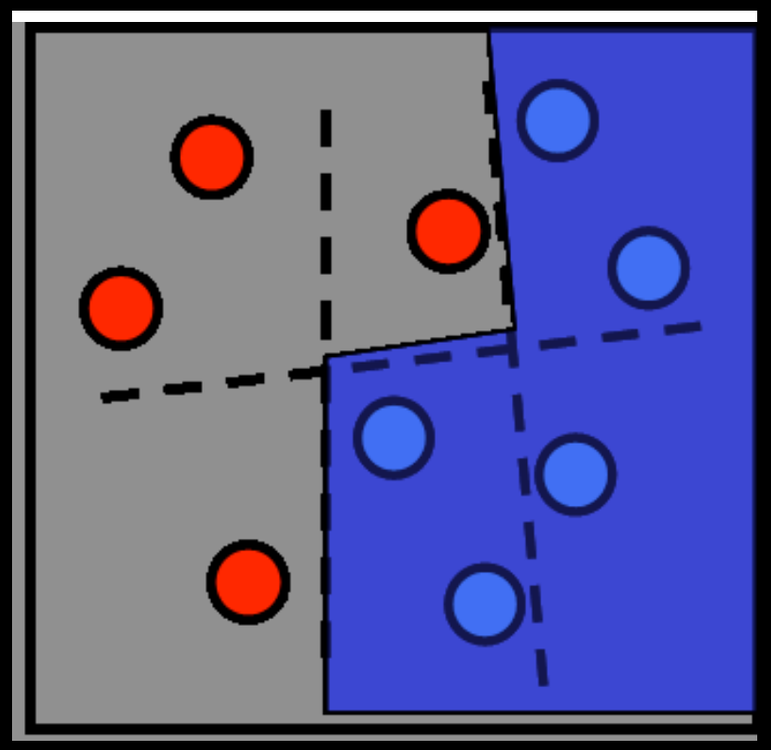
OpenCV 3.0

- In terms of detectors all the standards one are still there:-
  - Viola & Jones style face detector.

  - Dalal & Triggs style pedestrian detector.

- Support in 3.0 now for deformable parts based models.

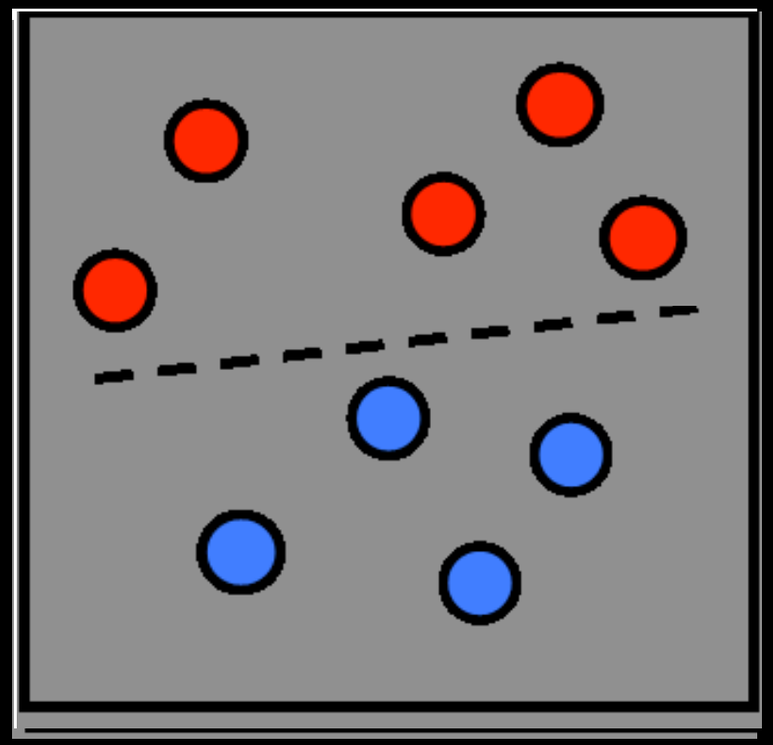
Viola & Jones



Computationally Expensive

“High Capacity”

Viola & Jones

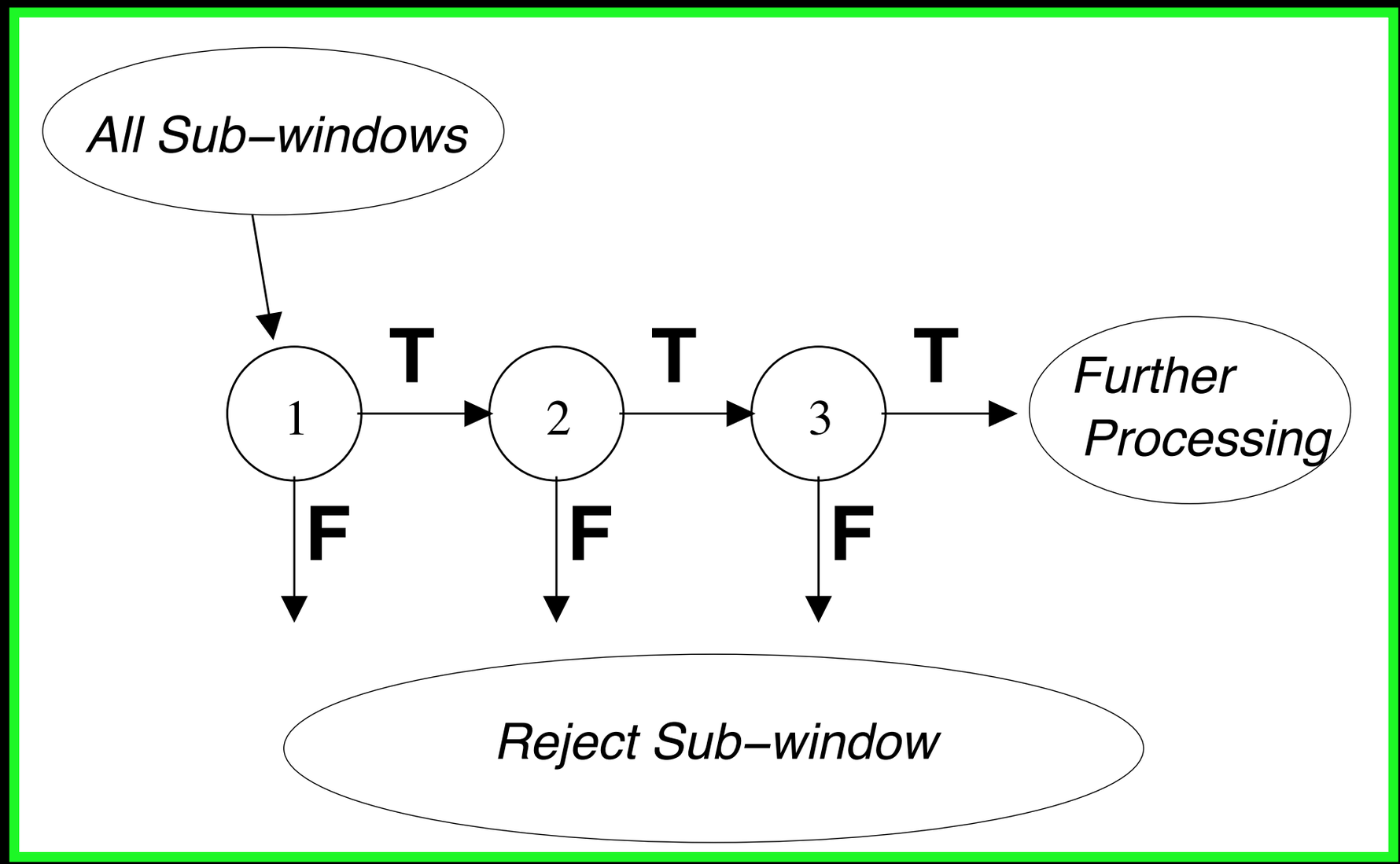


Computationally Cheap

“Low Capacity”

Viola & Jones

- Instead of searching all regions of an image with the same complexity classifier, we can use a cascade.



Viola & Jones

- Vast majority of regions in an natural image can be rejected using small capacity classifiers.

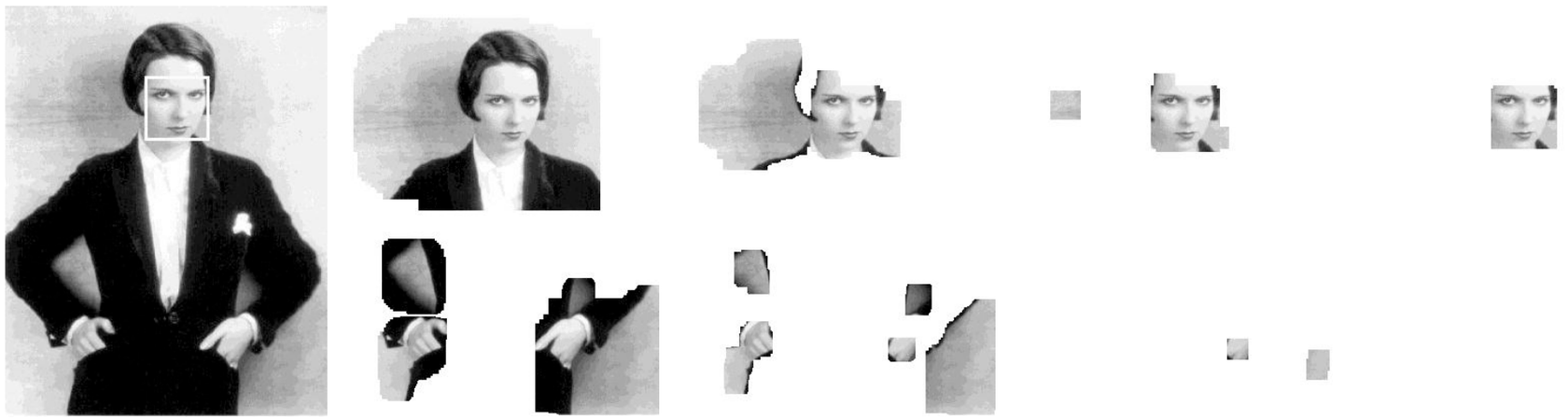
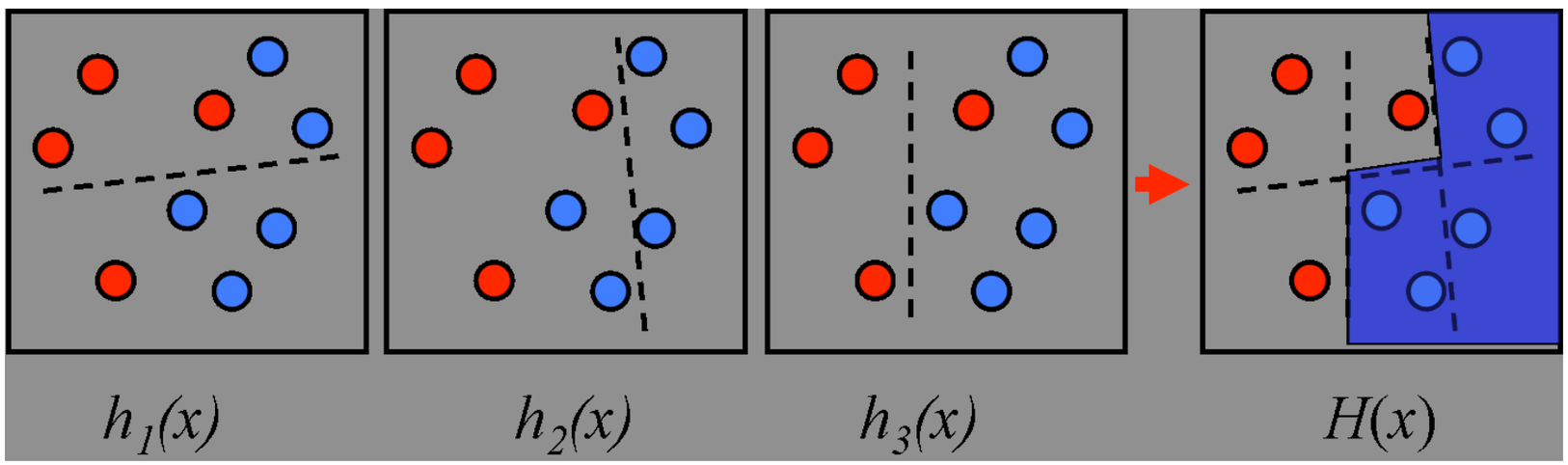


Figure 4: **From left to right:** *input image, followed by portions of the image which contain un-reject patches after the sequential evaluation of 1* (13.3% patches remaining), *10* (2.6%), *20* (0.01%) *and 30* (0.002%) *support vectors. Note that in these images, a pixel is displayed if it is part of any remaining un-rejected patch at any scale, orientation or position This explains the apparent discrepancy between the above percentages and the visual impression.*

Viola & Jones

- Boosting is ideally suited to be used with box filters.
- Techniques like AdaBoost, LogitBoost or GentleBoost can naturally learn a complex classifier from a cascade of weak classifiers.

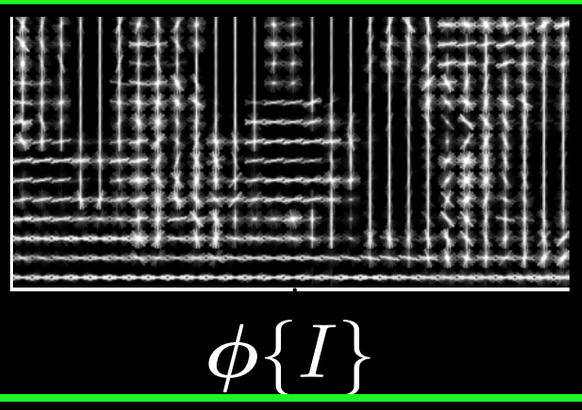
$$H(\mathbf{x}) \;=\; \text{sign}\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$$



Deformable Parts Models

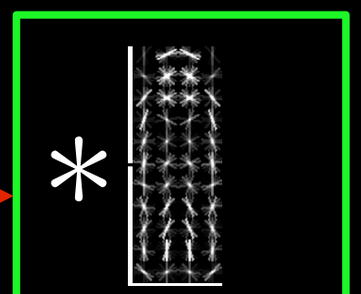
I

Feature Extraction

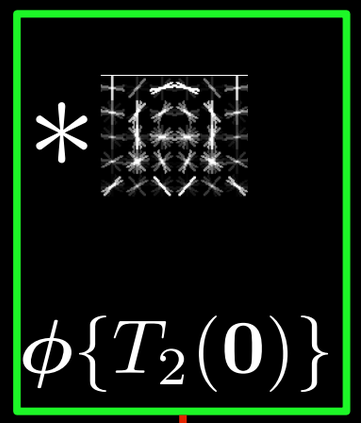


$\phi\{I\}$

Local Search

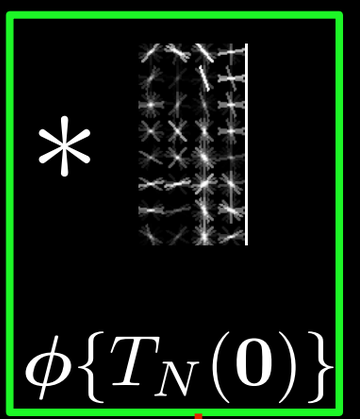


$\phi\{T_1(\mathbf{0})\}$



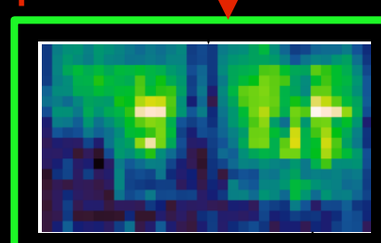
$\phi\{T_2(\mathbf{0})\}$

.....

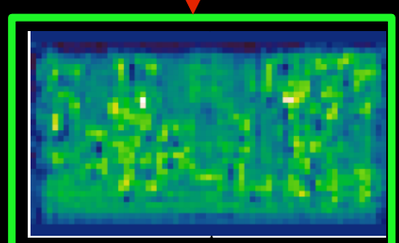


$\phi\{T_N(\mathbf{0})\}$

Part Responses

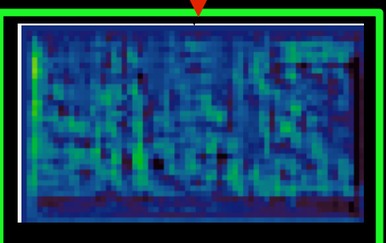


D_1



D_2

.....

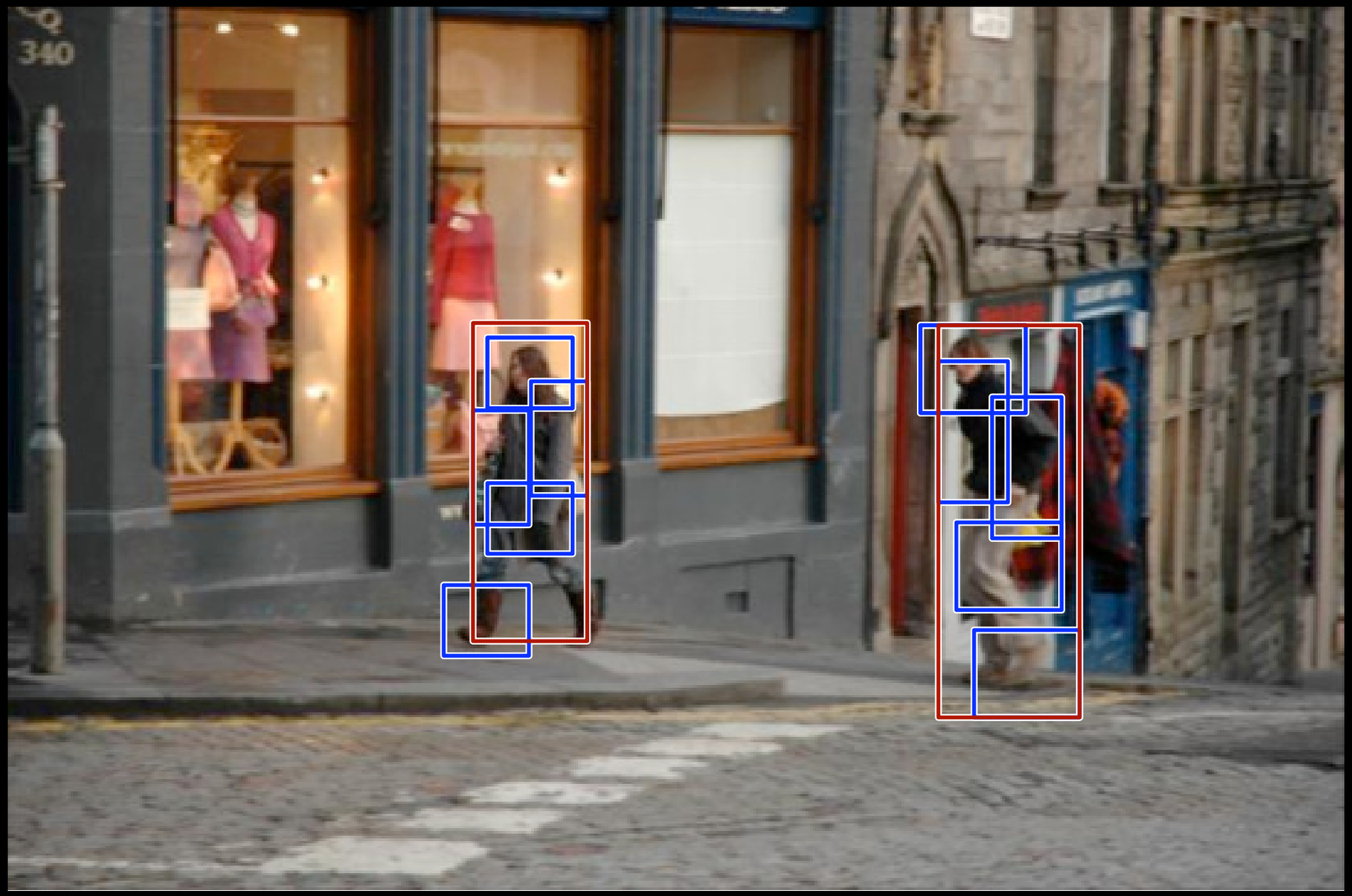


D_N

Color Encoding of Filter Responses

Felzenszwalb, Girshick, McAllester & Ramanan, 2010

Deformable Parts Models



Felzenszwalb, Girshick, McAllester & Ramanan, 2010

`Dlib` C++ for Computer Vision

- `Dlib` is a general purpose cross-platform C++ library designed using contract programming and modern C++ techniques.
- It is open source software and licensed under the Boost Software License.
- Code is platform independent (Windows, Linux, MAC OS X).
- Check out more details at the link - http://dlib.net/
- Very useful set of vision and learning tools.

Why `Dlib` is useful?

- What makes `Dlib` very cool, is its ability to train your own object detectors quickly and easily.
- This is hard to do in OpenCV as it relies on something called “Hard Negative Mining”.
- Requires setting tricky parameters, and can often takes hours/days to train a model.

Why `Dlib` is useful?

- What makes `Dlib` very cool, is its ability to train your own object detectors quickly and easily.
- This is hard to do in OpenCV as it relies on something called “Hard Negative Mining”.
- Requires setting tricky parameters, and can often takes hours/days to train a model.

Dlib - Make your own detector!

- Dlib - uses the well known HOG - SVM pipeline for object detection - Dalal & Triggs 2005.
- Does not rely on HNM, instead employs Structural Support Vector Machine (SVM).
- No need for negative training set, no messy parameters.
- Using this tutorial authors were able to learn a face detector in just a few minutes using Dlib.

Visualization of HOG Detector →

`Dlib` versus OpenCV

Red Box - `Dlib`

Blue Circles - `OpenCV`

Taken from http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html.

`Dlib` versus OpenCV

Red Box - `Dlib`

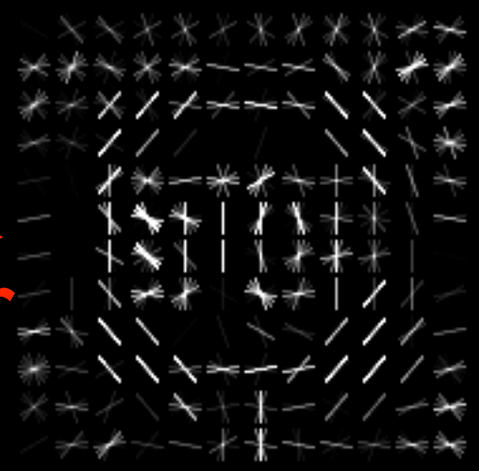
Blue Circles - `OpenCV`

Taken from http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html.

`Dlib` versus OpenCV

- Another example - 8 images of stop signs downloaded and labeled.
- `Dlib` was then used to create a HOG detector.

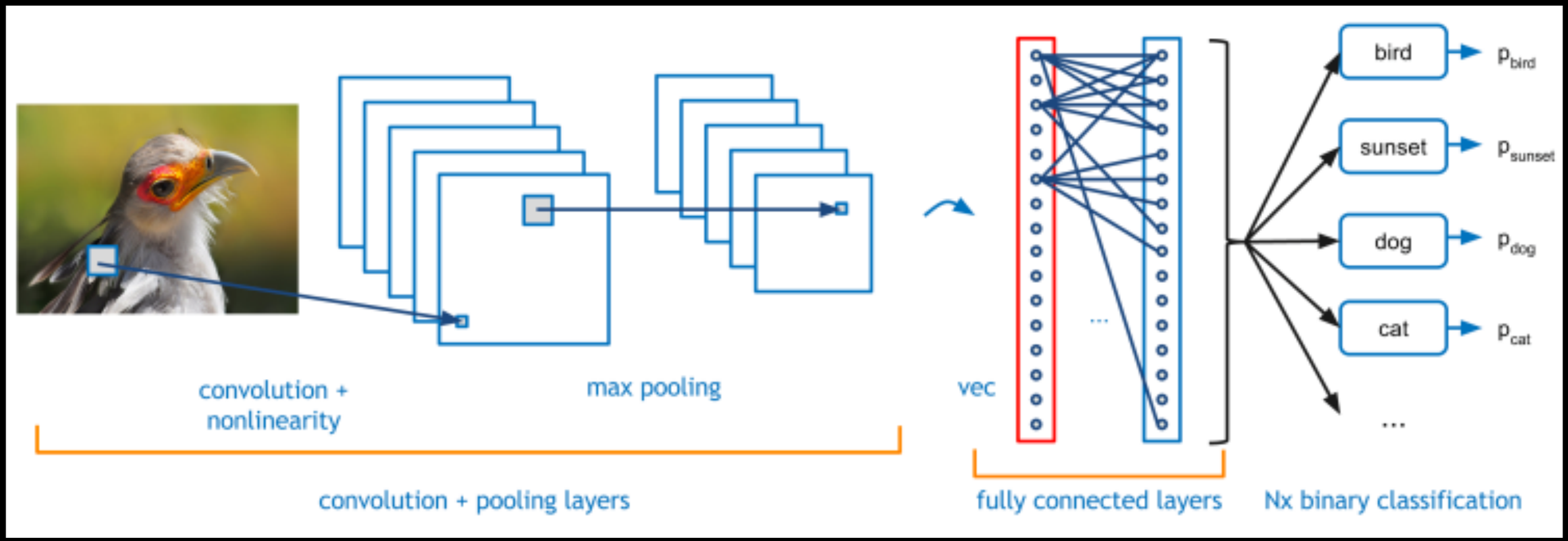
Visualization of HOG Detector →



Taken from http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html.

CAFFE

- Deep-learning has taken the vision world by storm.
- Essentially an extension of neural networks.
- State of the art for object detection (e.g Imagenet).
- CAFFE is one of the most popular packages.



Taken from - DIY Deep Learning for Vision.

CAFFE

- Open framework, models, and worked examples for deep learning.
- Pure C++ / CUDA architecture for deep learning.
- Command line, Python, MATLAB interfaces.
- Fast, well-tested code.
- Tools, reference models, demos, and recipes.
- Seamless switch between CPU and GPU.

Prototype

Train

Deploy

Taken from - DIY Deep Learning for Vision.

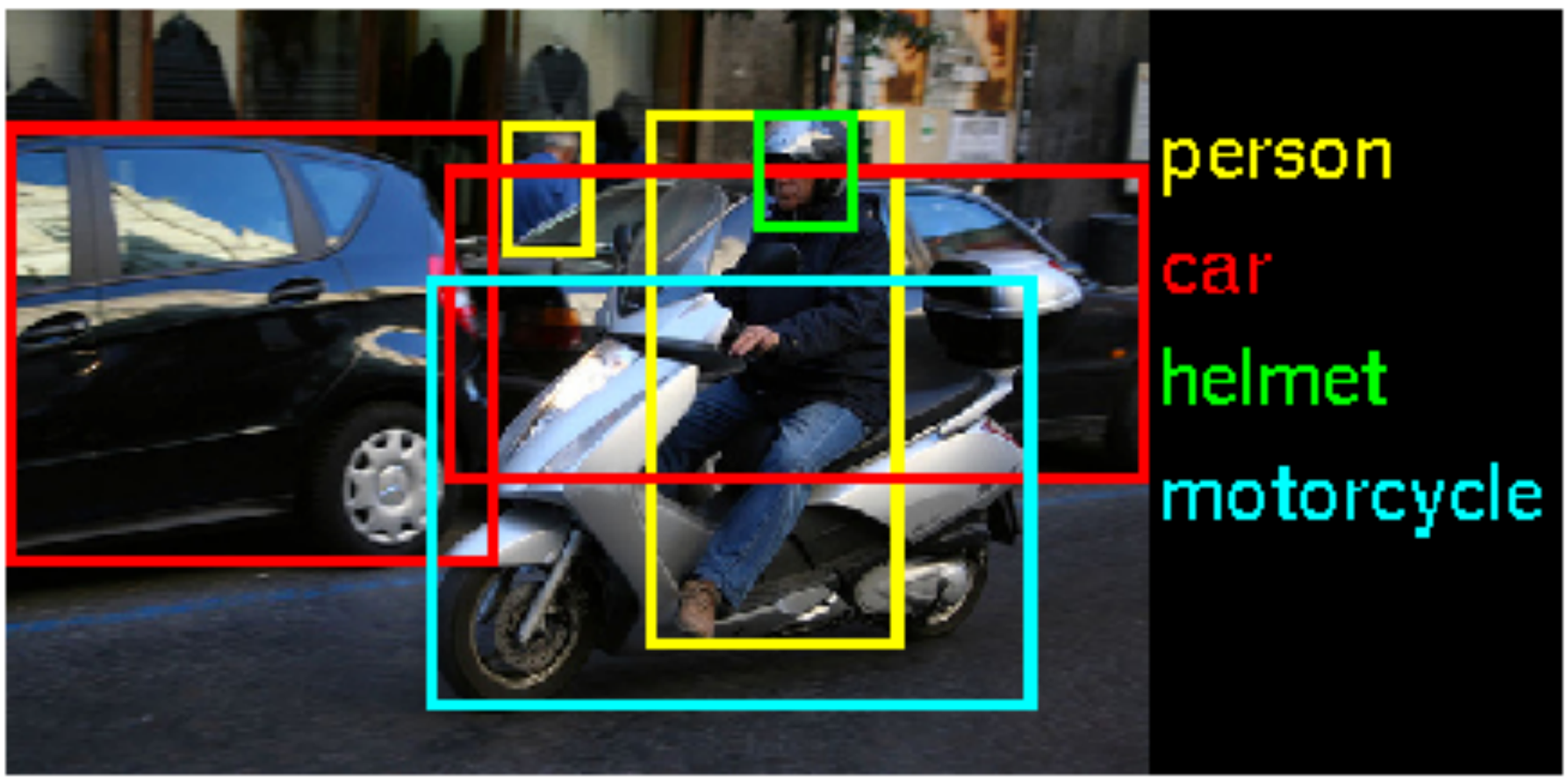
CAFFE

- Pre-trained models are openly available at the CAFFE Model Zoo.
- open collection of deep models to share innovation
  - VGG ILSVRC14 + Devil models in the zoo
  - Network-in-Network / CCCP model in the zoo
  - MIT Places scene recognition model in the zoo
  - help disseminate and reproduce research
  - bundled tools for loading and publishing models

Taken from - DIY Deep Learning for Vision.

CAFFE

- Developed in Berkley.
- Released under BSD-Clause 2 license.
- See here for very good tutorial on its use.
- Other packages in use - notably Torch 7 & MatConvNet.



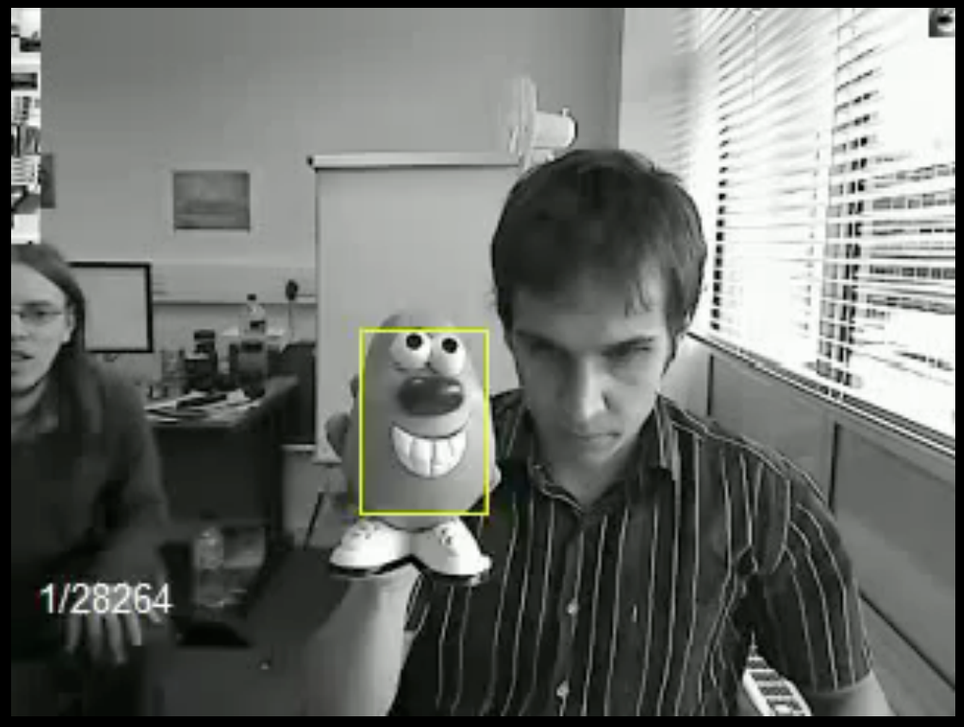
Today

- Motivation - Object Detection/Tracking
- Computer Vision as a “Black Box” - Considerations
- Detection
  - Computer Vision as a Service (VMX, Project Oxford, Clarifai).
  - OpenCV 3.0 (face and pedestrian detectors, what’s new?)
  - DLib C++ (create your own detector!!!)
  - Caffe (Deep Learning)
- Tracking
  - Correlation Filters (fast in tracking in just a few lines of code)
  - Predator (tracking an object efficiently/quickly)

Just a sampling - by no means a complete list!!!

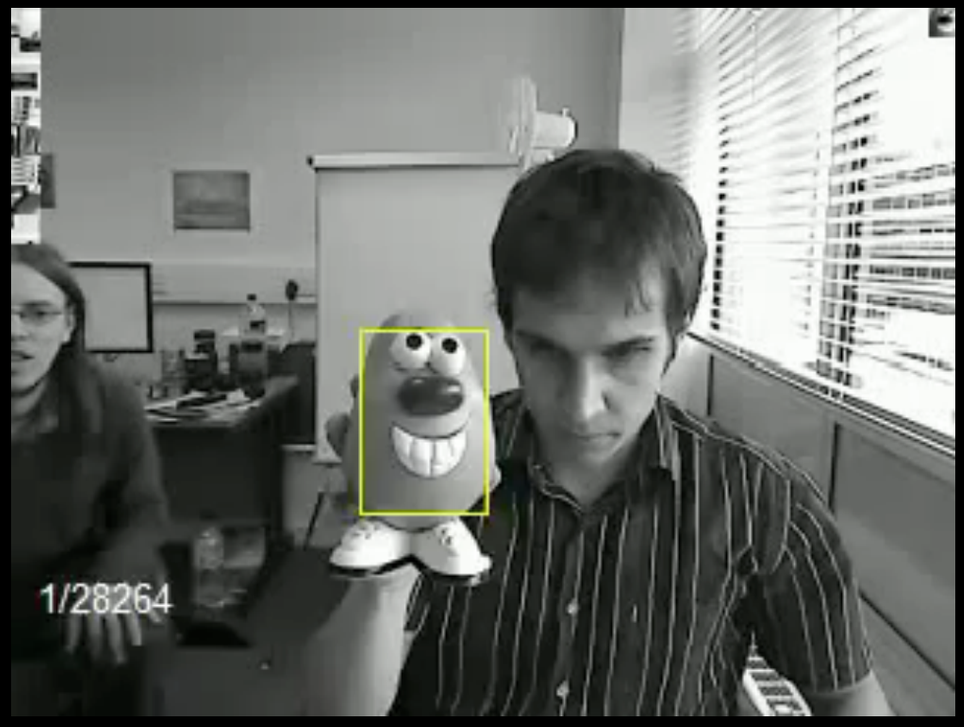
Predator Tracker

- Predator - or Track-Learning-Detection (TLD)
- Based on Kalal et al. ICCV 2011.
- Released under GPL v3.0.



Predator Tracker

- Predator - or Track-Learning-Detection (TLD)
- Based on Kalal et al. ICCV 2011.
- Released under GPL v3.0.



Predator Tracker

zk00006 / OpenTLD

Watch 183 S

Official source code for TLD

15 commits | 1 branch | 0 releases | 1 contributor

Branch: master | OpenTLD / +

Update README

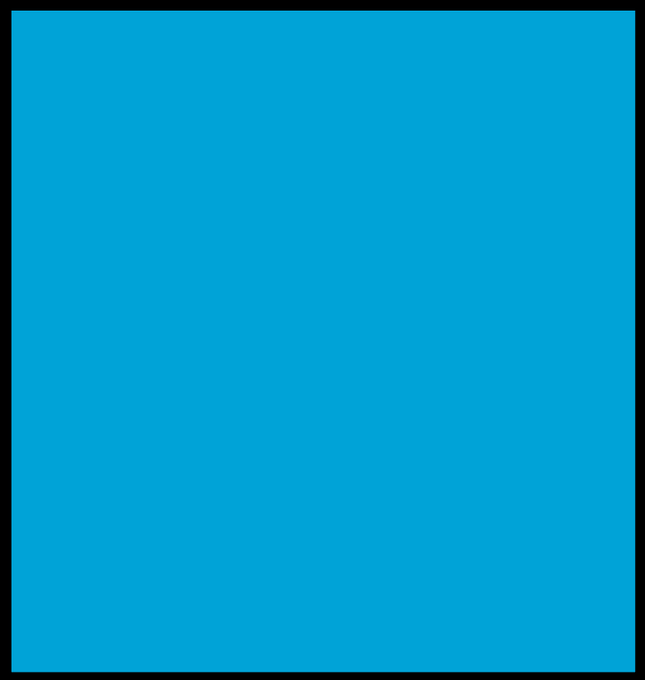
zk00006 authored on May 22, 2014 — latest commit 953e2df965

_input	first commit	5 years ago
_output	first commit	5 years ago
bbox	first commit	5 years ago
img	first commit	5 years ago
mex	first commit	5 years ago
other	first commit	5 years ago

Taken from https://github.com/zk00006/OpenTLD.

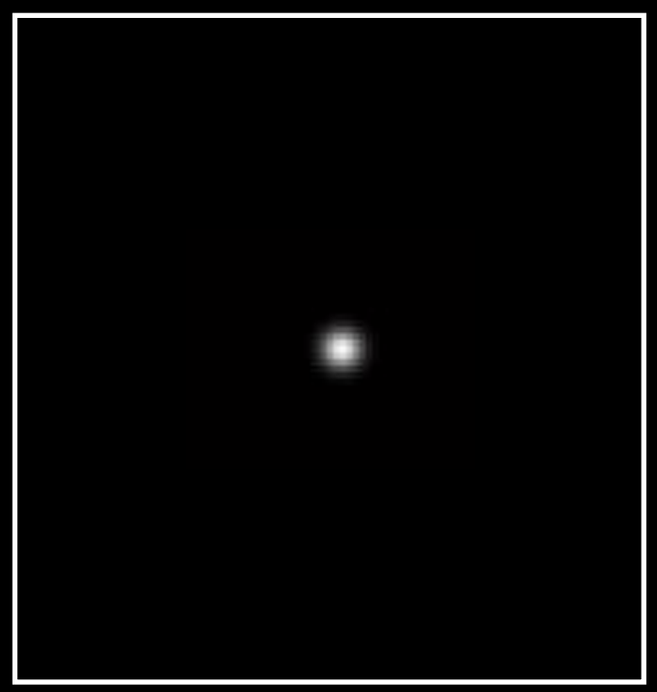
Correlation Filters

*



"unknown filter"
$\mathbf{h}$

"known signal" $\mathbf{x}$

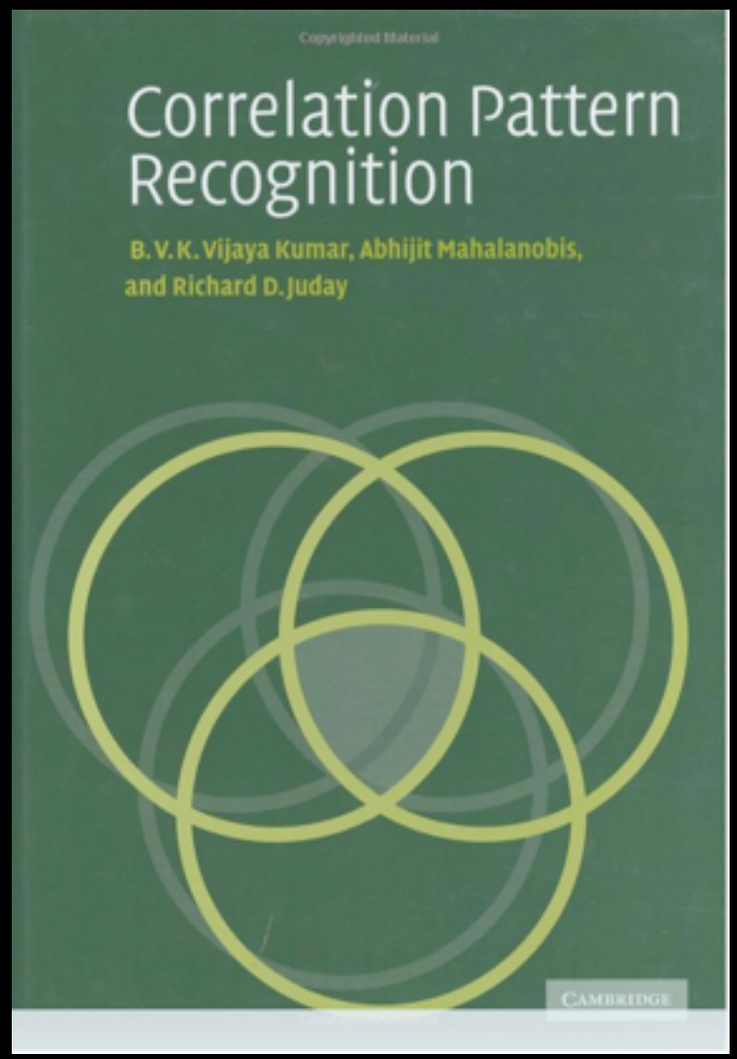


"known response" $\mathbf{y}$

Correlation Filters

- Can build a correlation filter from **5** lines of MATLAB code!!!

$$\hat{\mathbf{h}} = \hat{\mathbf{s}}_{xy} \circ^{-1} (\hat{\mathbf{s}}_{xx} + \lambda \mathbf{1})$$

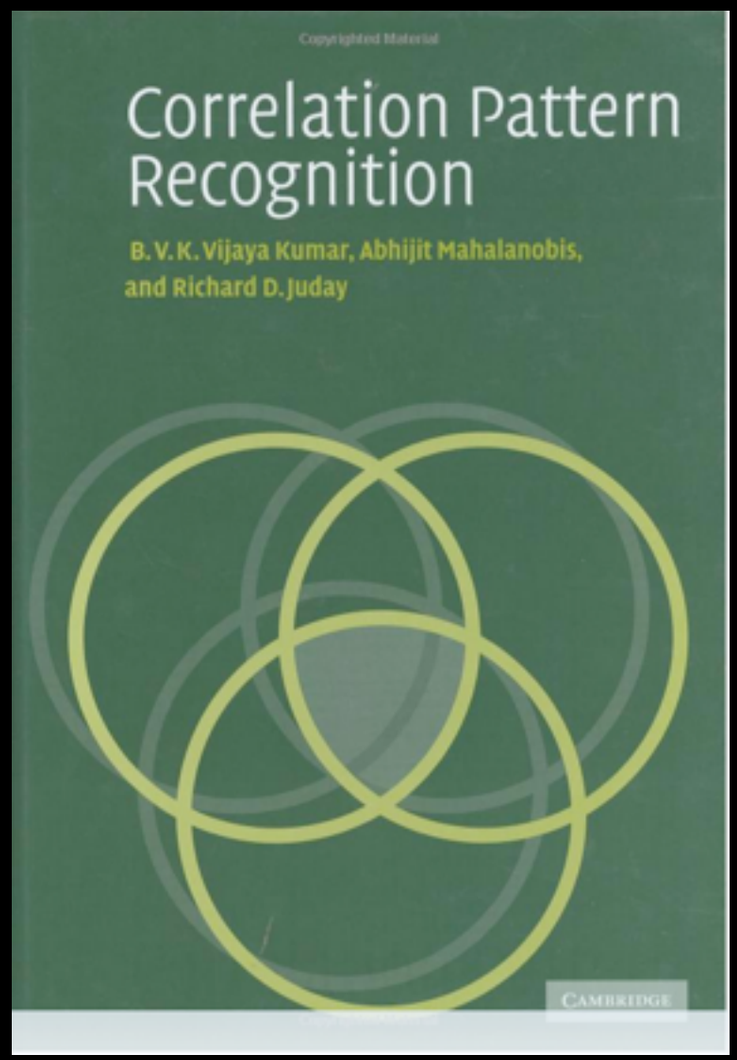


Correlation Filters

- Can build a correlation filter from **5** lines of MATLAB code!!!

$$\hat{\mathbf{h}} = \hat{\mathbf{s}}_{xy} \circ^{-1} (\hat{\mathbf{s}}_{xx} + \lambda \mathbf{1})$$

```
>> xf = fft2(x);
```

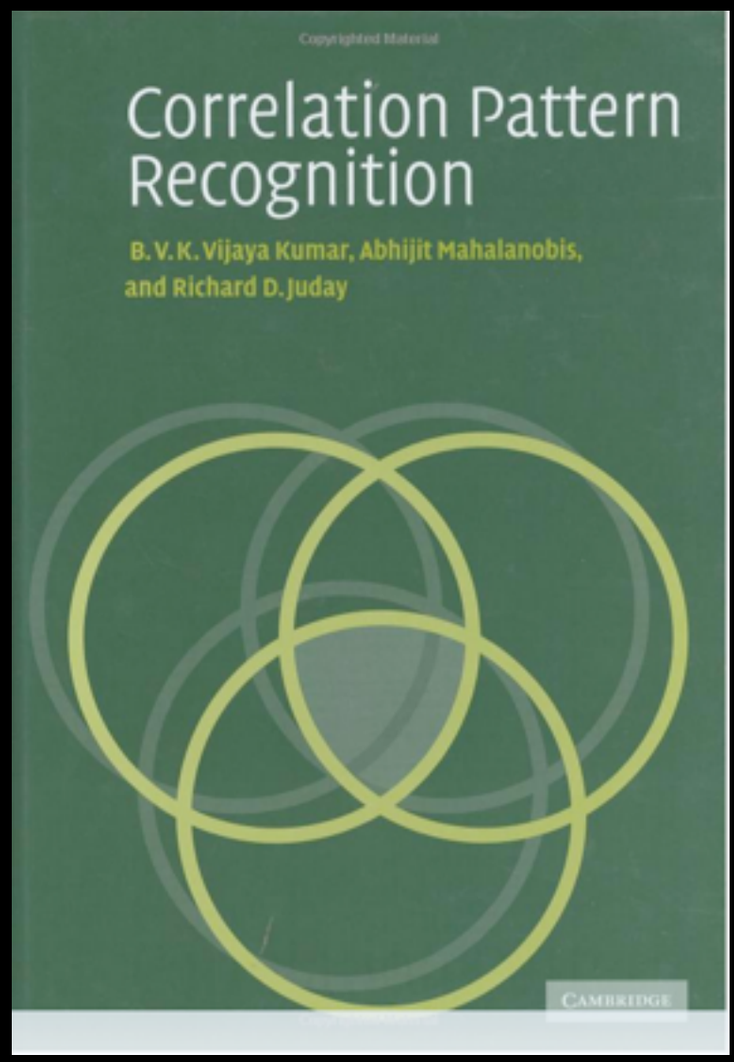


Correlation Filters

- Can build a correlation filter from **5** lines of MATLAB code!!!

$$\hat{\mathbf{h}} = \hat{\mathbf{s}}_{xy} \circ^{-1} (\hat{\mathbf{s}}_{xx} + \lambda \mathbf{1})$$

```
>> xf = fft2(x);
>> yf = fft2(y);
```

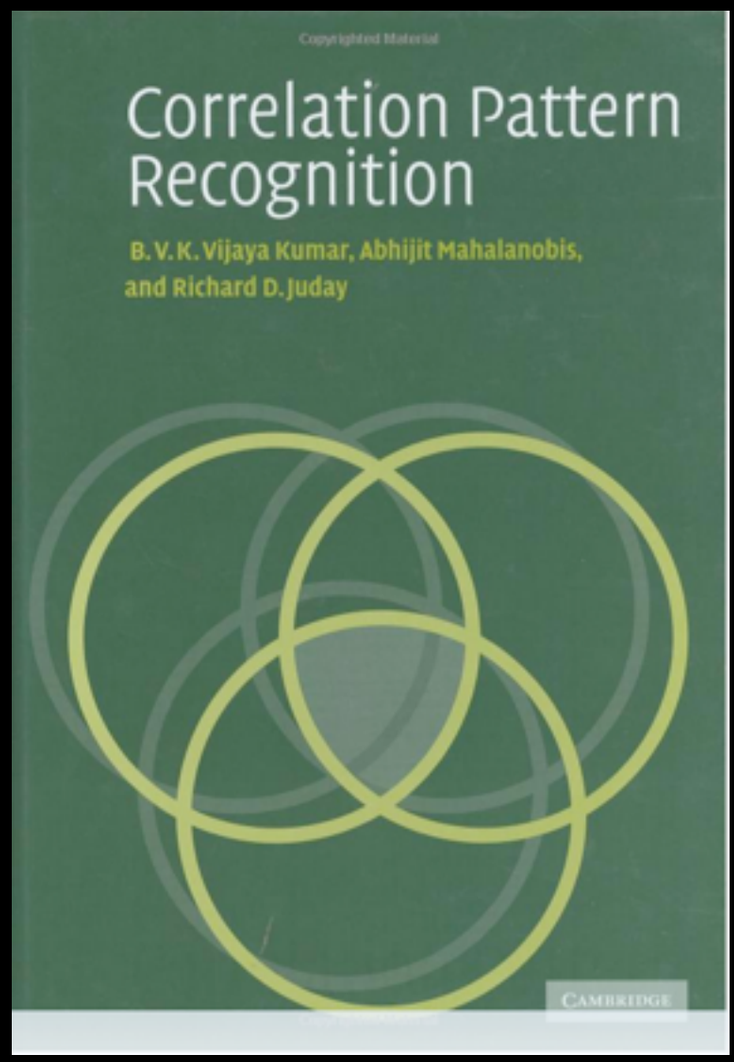


Correlation Filters

- Can build a correlation filter from **5** lines of MATLAB code!!!

$$\hat{\mathbf{h}} = \hat{\mathbf{s}}_{xy} \circ^{-1} (\hat{\mathbf{s}}_{xx} + \lambda \mathbf{1})$$

```
>> xf = fft2(x);
>> yf = fft2(y);
>> sxx = xf.*conj(xf);
```

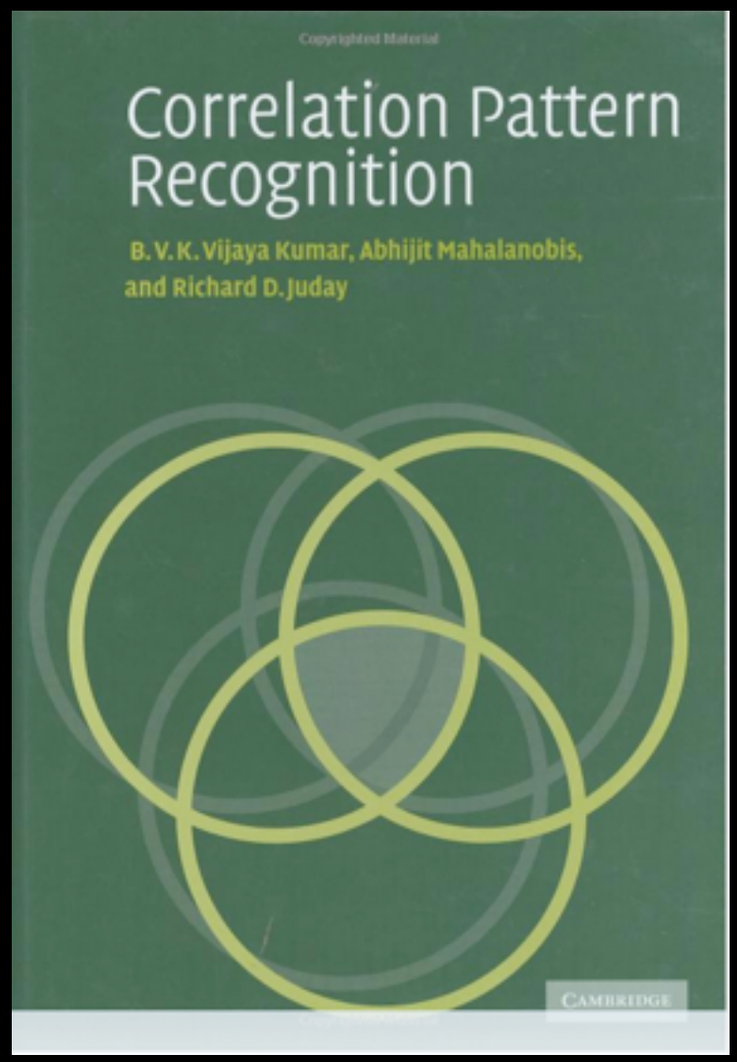


Correlation Filters

- Can build a correlation filter from **5** lines of MATLAB code!!!

$$\hat{\mathbf{h}} = \hat{\mathbf{s}}_{xy} \circ^{-1} (\hat{\mathbf{s}}_{xx} + \lambda \mathbf{1})$$

```
>> xf = fft2(x);
>> yf = fft2(y);
>> sxx = xf.*conj(xf);
>> sxy = xf.*conj(yf);
```

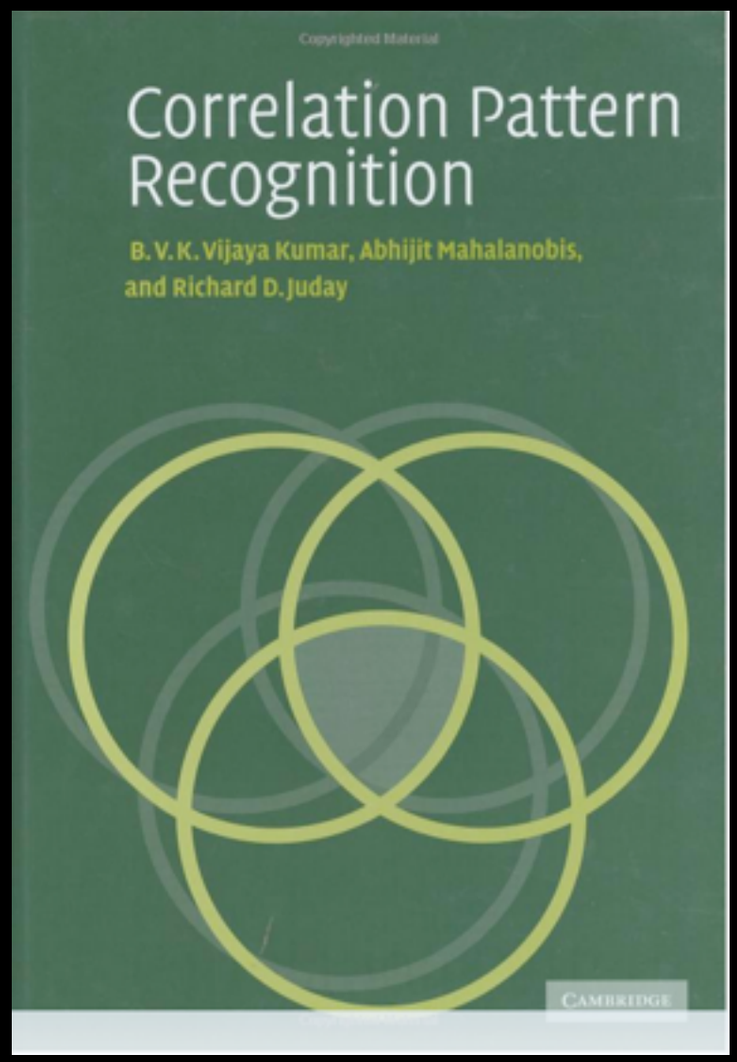


Correlation Filters

- Can build a correlation filter from **5** lines of MATLAB code!!!

$$\hat{\mathbf{h}} = \hat{\mathbf{s}}_{xy} \circ^{-1} (\hat{\mathbf{s}}_{xx} + \lambda \mathbf{1})$$

```
>> xf = fft2(x);
>> yf = fft2(y);
>> sxx = xf.*conj(xf);
>> sxy = xf.*conj(yf);
>> hf = sxy./(sxx + 1e-3);
```



Algorithm	Frame Rate	CPU
FragTrack[1]	realtime	Unknown
GBDL[19]	realtime	3.4 Ghz Pent. 4
IVT [17]	7.5fps	2.8Ghz CPU
MILTrack[2]	25 fps	Core 2 Quad
MOSSE Filters	669fps	2.4Ghz Core 2 Duo

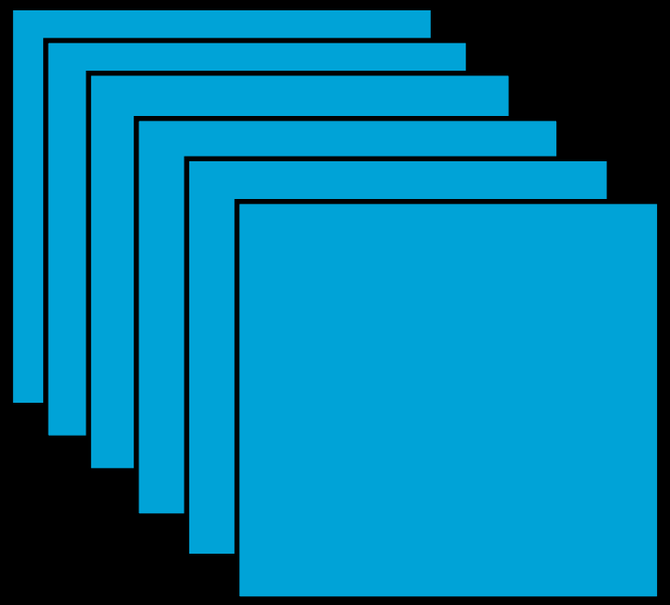
Taken from Bolme et al. CVPR 2010.

Algorithm	Frame Rate	CPU
FragTrack[1]	realtime	Unknown
GBDL[19]	realtime	3.4 Ghz Pent. 4
IVT [17]	7.5fps	2.8Ghz CPU
MILTrack[2]	25 fps	Core 2 Quad
MOSSE Filters	669fps	2.4Ghz Core 2 Duo

Taken from Bolme et al. CVPR 2010.

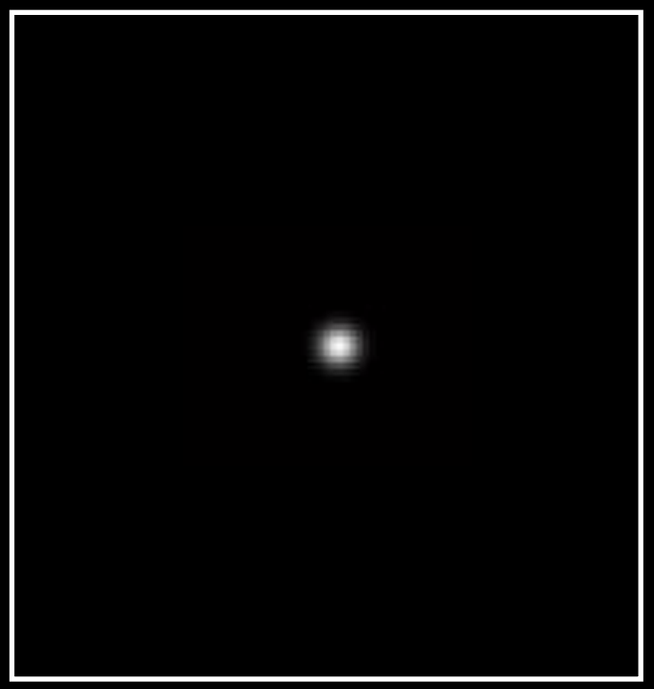
HOG - Correlation Filters

*



“unknown filter” $\mathbf{h}$

“known signal” $\mathbf{x}$

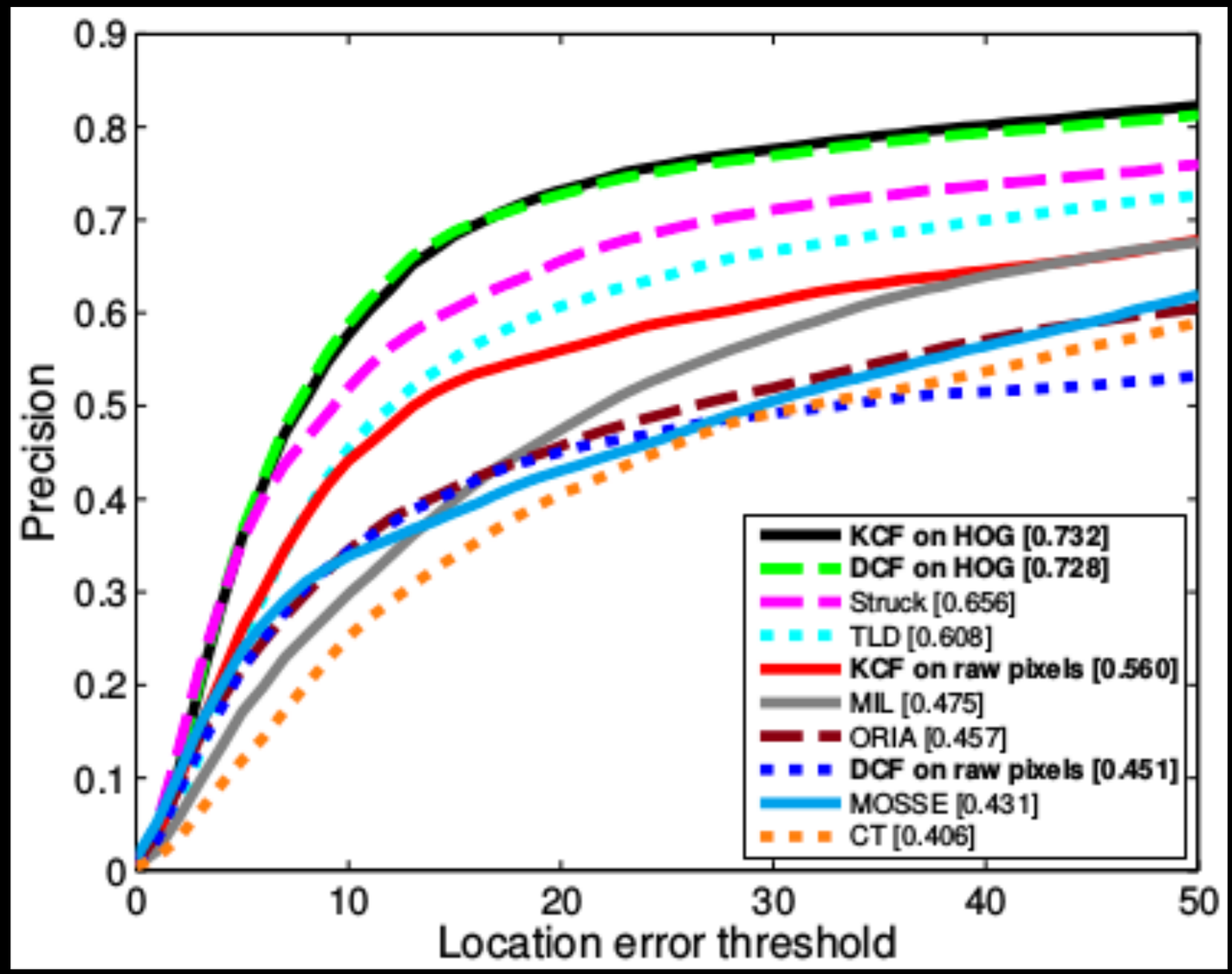


“known response” $\mathbf{y}$

Kiani, Sim and Lucey ICCV 2013.

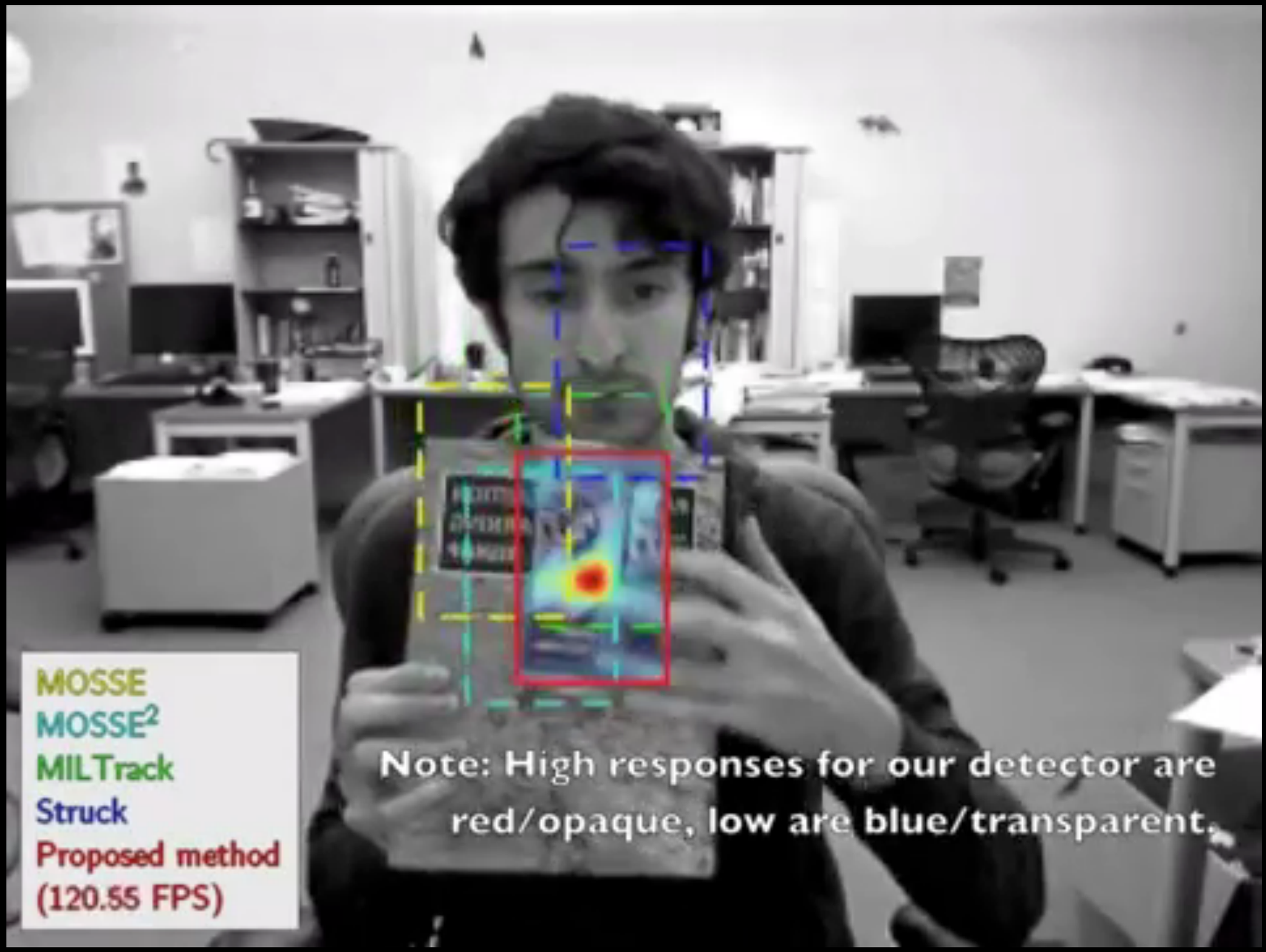
Kernelized Correlation Filters

- State of the art are “Kernelized Correlation Filters”.
- Code available in MATLAB and Python.



Henriques et al. PAMI 2015.

Kernelized Correlation Filters



Kernelized Correlation Filters