Implementing Histogram of Oriented Gradients on a Parallel Vision Processor

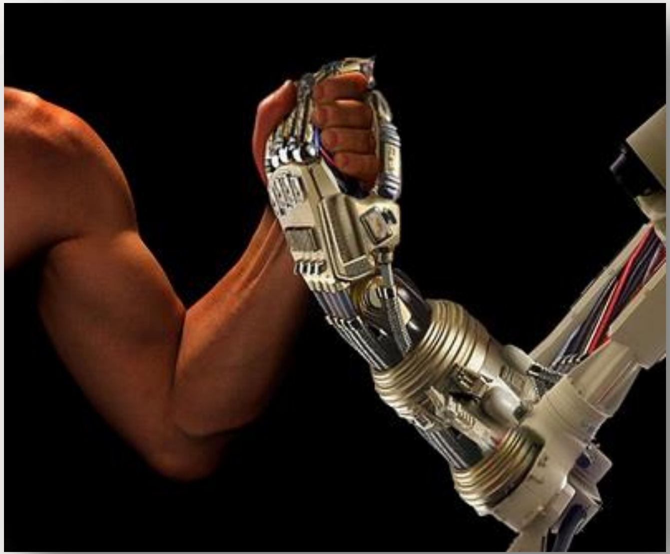
Marco Jacobs

May 29, 2014

The Challenge: Make Our Phones, Cars, Etc. Smarter Than Us

- 50% of the brain is used for vision
- Body uses 100W
- Brain consumes 20W
- → about 10W for vision analysis

- Challenge: beat the human
- Seems really hard, but can focus on specific areas:
  - Build machines that are faster, safer, cheaper, last longer, more accurate, etc.



Object Detection — Key Vision Algorithm

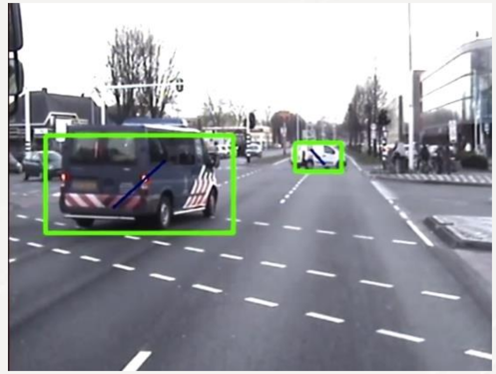
- Object detection/recognition
  - "That's a person"
  - "That's a car"
- Dalal & Triggs, "Histograms of Oriented Gradients for Human Detection", INRIA (France), 2005
- Seminal paper: "100x accuracy increase in object detection"

Computer Vision: Algorithms and Applications

Richard Szeliski

Object Detection — Sample Applications

automotive:
pedestrian detection



automotive:
vehicle detection

surveillance:
perimeter detection

research:
animal detection

industry:
object inspection

search:
image categorization

Step 1 — Training the Classifier, Offline

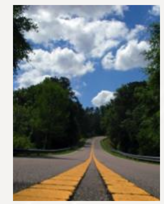
Normalized, fixed resolution images with yes/no annotations

Extract feature vector (HOG)

Learn binary classifier (SVM)

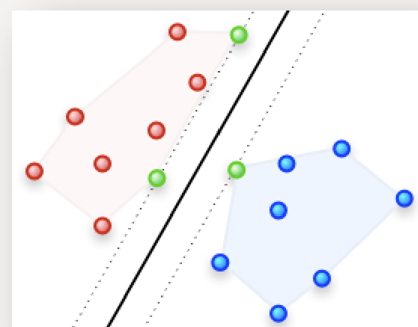
Object yes / no

pedestrian



no pedestrian
→ resample

false positives ← retrain



SVM classifier

Step 2 – Object Detection, Real-time

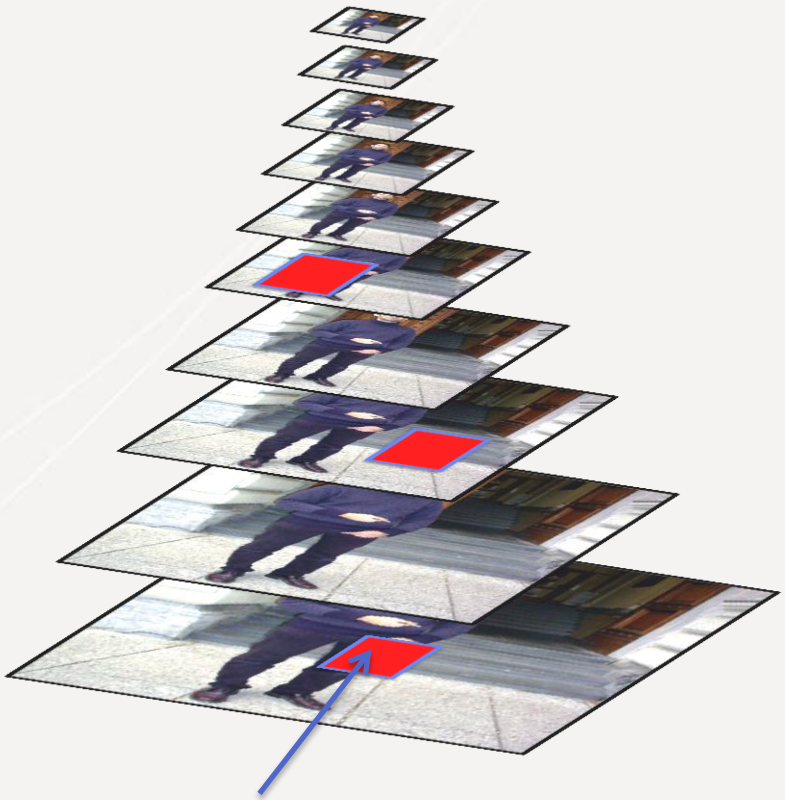
Scan image at different scales and locations

Extract features over window (vector)

Run SVM classifier (object yes/no)

Fuse multiple detections

Final object detected



Detection window

videantis

Training – INRIA people dataset

- Variety of poses
- Variable appearance / clothing
- Complex background
- Unconstrained illumination
- Occlusions and different scales

- Main assumption:
    - clearly visible mostly upright people

Step 2 – Object Detection, Real-time

Grayscale input image

Generate multiscale pyramid

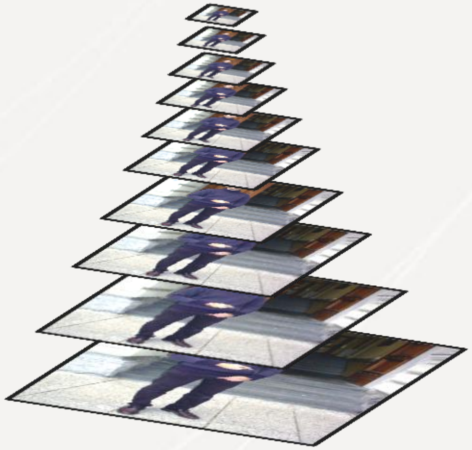
Gamma normalization

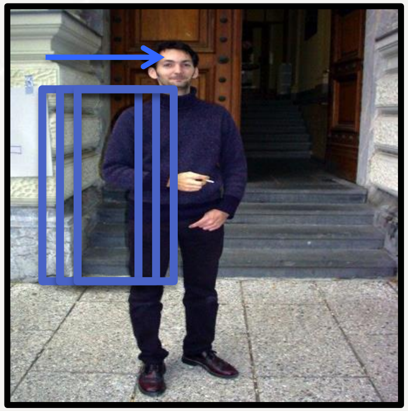
Gradient calculation
(calculate angle and magnitude)

Histogram per block

SVM per window position

Non-max suppression





videantis

Step 2 – Object Detection, Real-time

Grayscale input image

Generate multiscale pyramid

Gamma normalization

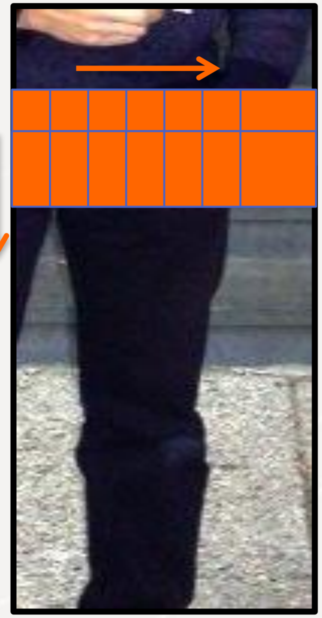
Gradient calculation
(calculate angle and magnitude)

Histogram per block

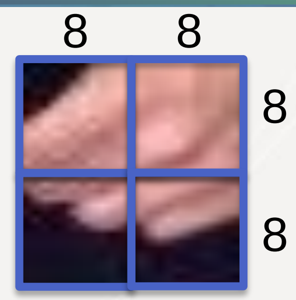
SVM per window position

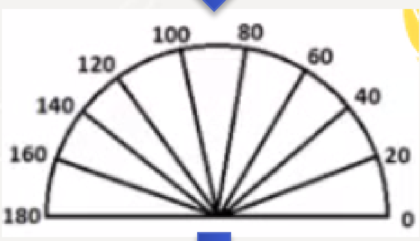
Non-max suppression

64x128 pixels

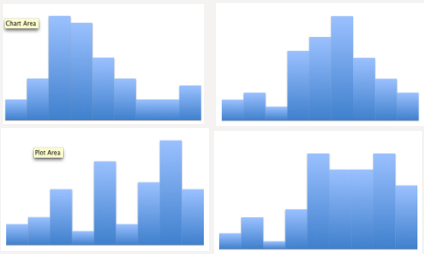


7*15=105
16x16 positions





Gradient direction & magnitude



4 histograms of 9 cells

Per 64x128 window: feature vector of 105x4x9=3780
Multiply by SVM vector → object detected yes/no

HOG in Combination With Feature Detect & Track

- HOG compute complexity is ~10x optical flow (for full frame rate and resolution)
- To reduce complexity, can locate features inside detected object window and track these across frame
- Can also calculate the direction of the object
- Significantly reduces processor load

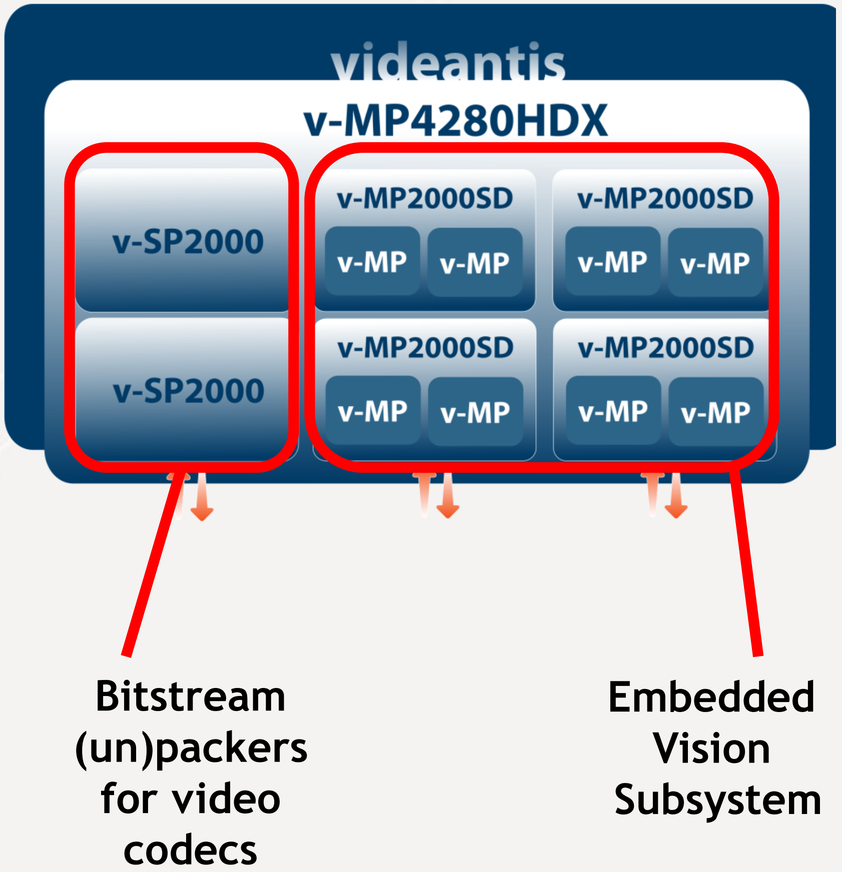
frame 1 | frame 1 | frame 2 | frame 3 | frame 4

HOG | Feature detect | Feature track | Feature track | Feature track

Videantis v-MP4000HDX Architecture

Heterogeneous, scalable multi-core IP

- v-SP for bitstream parsing/ generation in video codecs
- v-MP for pixel-processing:
  - vision, video encoding, decoding, image processing
- Each v-MP is VLIW & SIMD with own DMA
- v-MP4280HDX delivers:
  - 8 x ~25.6 GOPS per v-MP at 800MHz, total >200 GOPS
  - Less than 2mm^2 in 28nm



Architecture Trade-offs for Vision Algos

	Host CPU	GPUs	Imaging DSPs	v-MP4000HDX
ILP: VLIW or superscalar	**Superscalar** (Superscalar is expensive in HW)	**Varies, not disclosed Needs CPU**	**4-issue** >2 issue VLIW causes NOPs and requires loop unrolling	**2-issue VLIW** Right trade off
SIMD	**128-bit** requires second pipeline, RF, etc.	**Very wide array** not used efficiently by block-based algos	**>128-bit SIMD** Wide SIMD can't be used efficiently by block-based algos	**64/128-bit** Right trade off for imaging and video
Multicore	**1-4 cores** but cache coherency introduces overhead	**Many cores, with many restrictions**	**1 core**	**1-8+ cores** Supports diverse algorithms Scales to low or high end apps
Processor frequency	**2GHz+** Long pipeline introduces hardware overhead	**~1GHz** Medium/long pipelines	**500MHz-1GHz** Medium pipeline	**500MHz-1GHz** Medium pipeline
Caches / DMA	Multi-level caches	Multi-level caches	No cache, single DMA	No cache, DMA per core

Seamless OpenCV Acceleration

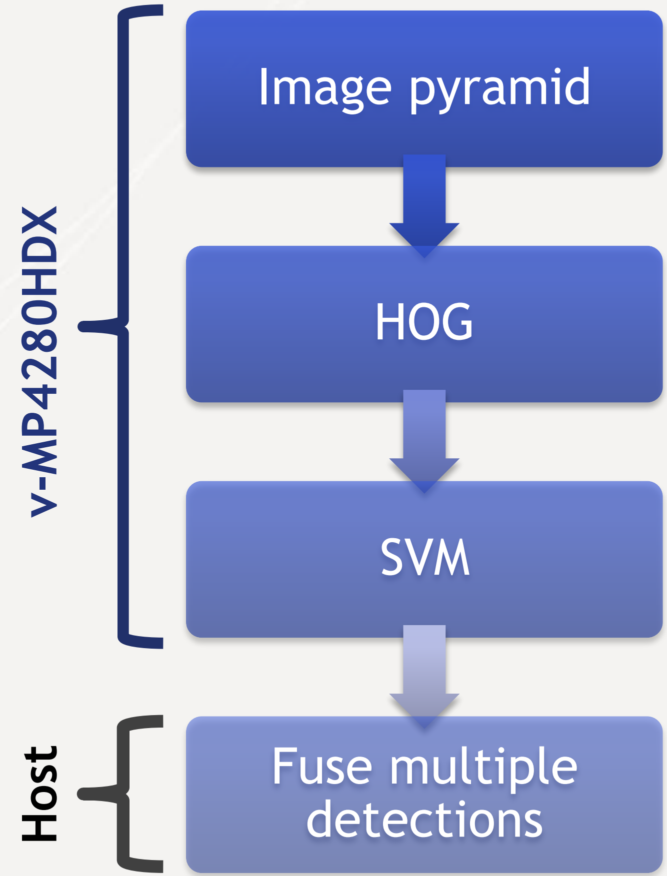
- “Lower-level” pixel processing processed on accelerator
- How to enable acceleration on v-MP4280HDX:
  - Replace all image data allocators

```
cvCreateMatHeader(…);
cvCreateData(…);
hog.detectMultiScale(…);
```

  - by new “shared memory” allocator

```
cvCreateMatHeader(…);
cvCreateDataOvl(…)
hog.detectMultiScale(…);
```

  - API internally takes care of moving data and processing onto accelerator
- “Higher-level” processing remains on host CPU for initial accelerated version

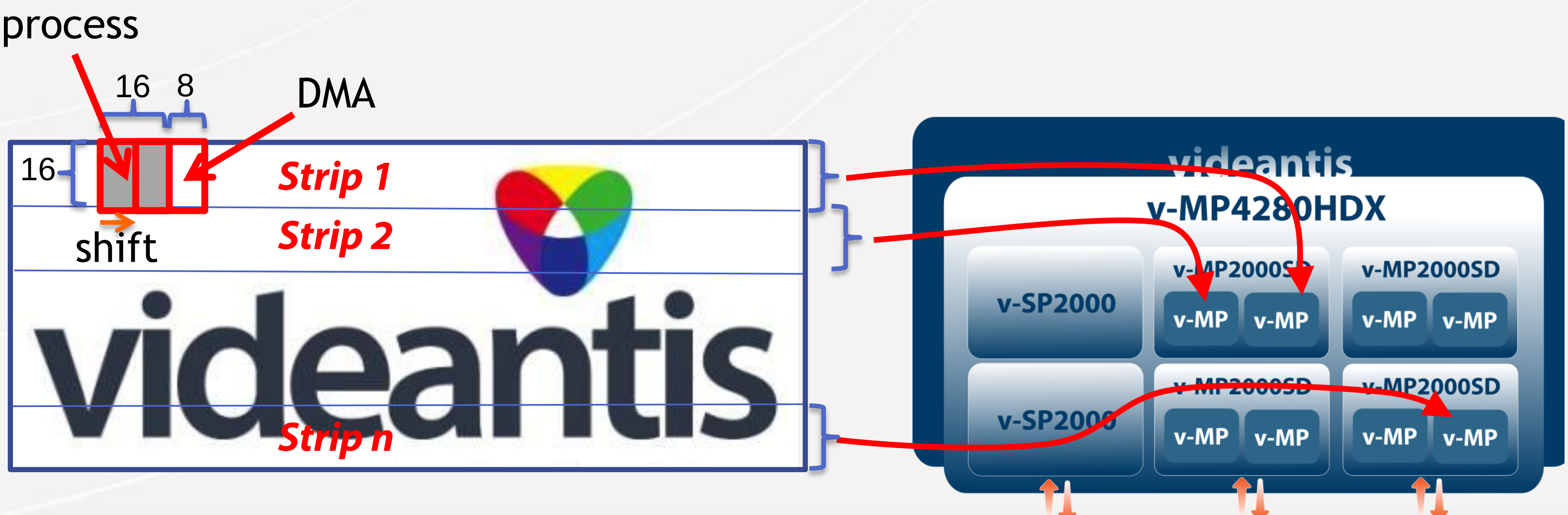


videantis

Mapping of HOG to v-MP4280HDX

Calculating HOG feature vectors in parallel:

- Each v-MP gets a slice of 16 pixels height
- Within the row, we calculate the HOG feature vector per 16x16 block
- We DMA in the next 8x16 block of data while the previous 16x16 block is processed



Mapping of SVM to v-MP4280HDX

Calculating the SVM dot products in parallel:

- We use the Daimler detector: 48x96 window versus 64x128 original Dalal and Triggs. The Daimler detector detects pedestrians that are smaller in view
- 4 histograms x 9 bins x 5x11 16x16 blocks, using 8-pixel overlap
- Process a column per v-MP. Keep the fixed SVM vector local to v-MP
- Process a sliding window in vertical direction, preload the next 5x 9x4 histograms

v-MP4280HDX HOG Performance and Power

- HOG in each image at 30 fps (each frame in video) or at 2 fps (for combination with tracking)

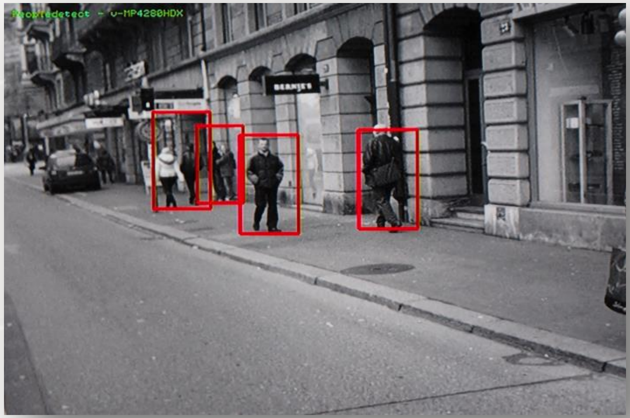
Resolution	v-MP cores @30 fps	Silicon area	Core power @30 fps	v-MP cores @2 fps	Core power @2 fps
VGA *	6 at 400MHz	$2.4mm^2$ 40nm LP	30 mW	1 at 160MHz	2 mW
720p	8 at 800MHz	$1.6mm^2$ 28nm HPM	40 mW	1 at 425MHz	2.7 mW

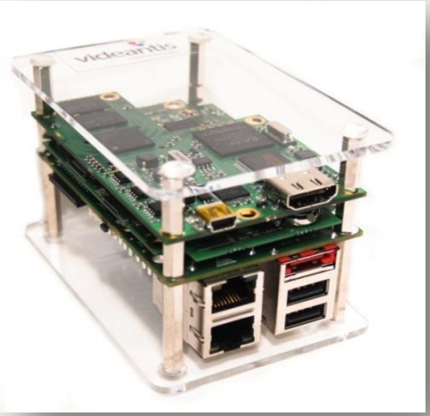
- 1.2GHz Cortex-A9 ARM runs VGA at ~1fps
- Performance v-MP4280HDX compared to ARM: 135x at same frequency
- Power v-MP4280HDX compared to ARM: >400x lower

* performance and power measured on videantis 40nm silicon

Conclusions

- HOG is a key algorithm for object detection
  - ~90% detection rate with 10^{-4} false positives per window
- Computationally demanding algorithm, ~10x more complex than feature detection or optical flow
- The algorithms can be implemented efficiently at high resolution while consuming low power on the videantis v-MP4000HDX vision processor





Please drop by our booth for a silicon demonstration

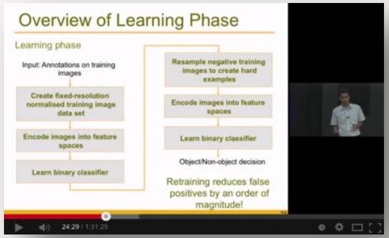
References — Videos

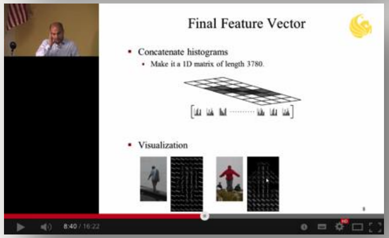
HOG:

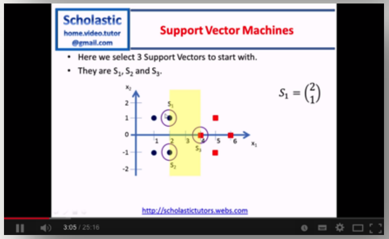
- Histogram of Oriented Gradients (HOG) for Object Detection in Images, Navneet Dalal
  - https://www.youtube.com/watch?v=7S5qXET179I
  - 19 mins: starts talking about HOG
- Histograms of Oriented Gradients, UCF Computer Vision Video Lectures 2012, Mubarak Shah
  - http://www.youtube.com/watch?v=0Zib1YEE4LU

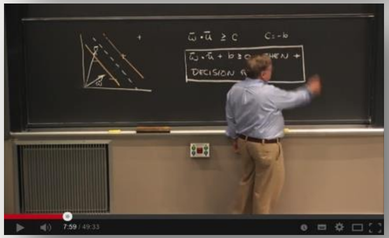
SVM:

- Support Vector Machines, Scholastic Home Video Tutor
  - https://www.youtube.com/watch?v=LXGaYVXkGtg
- Support Vector Machines, AI course Fall 2010, MIT
  - https://www.youtube.com/watch?v=_PwhiWxHK8o









Thank you

Marco Jacobs