Comparing Boosted Cascades to Deep Learning Architectures for Fast and Robust Coconut Tree Detection in Aerial Images

VISAPP2018, 27-29 January 2018

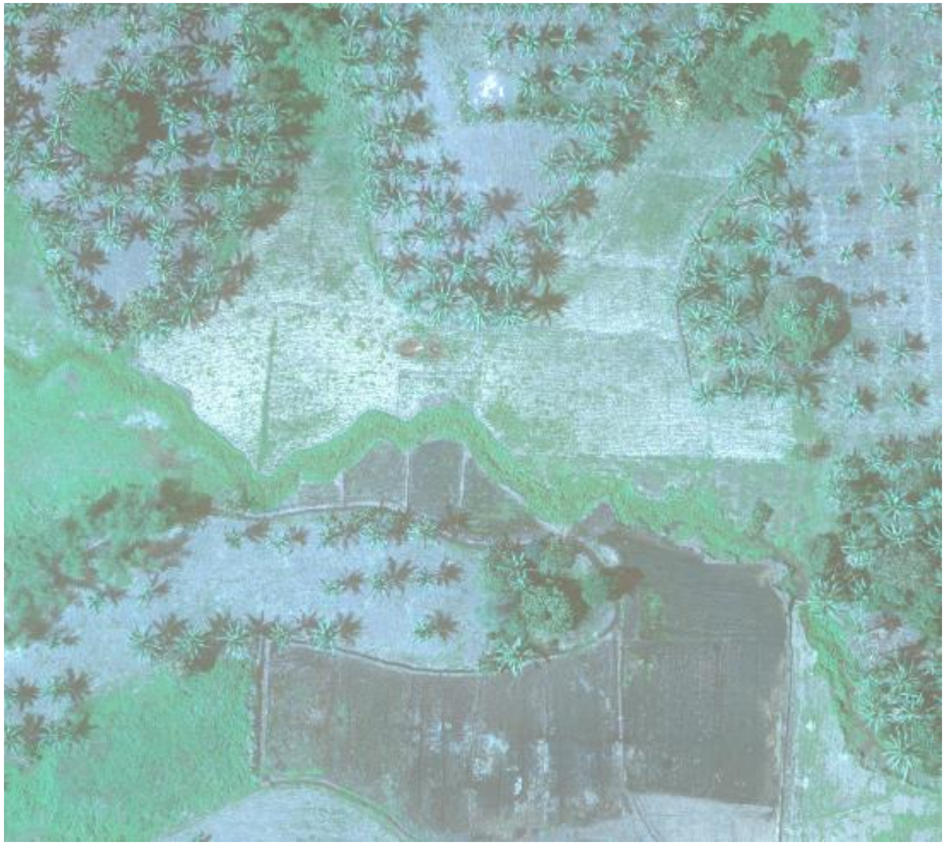
Steven Puttemans*, Kristof Van Beeck* and Toon Goedemé

Introduction

- Project in cooperation with Dutch company
  - ➔ Airborne mapping and surveying

- Farm and crop inspection
  - Crop counting, predict crop productivity
  - Crop performance, early detection of health problems
- Land use
  - Locations for expansion
  - Planning of land use, planting pattern, height differences
- Environmental analytics (predict erosion, flood risks,…)

Introduction

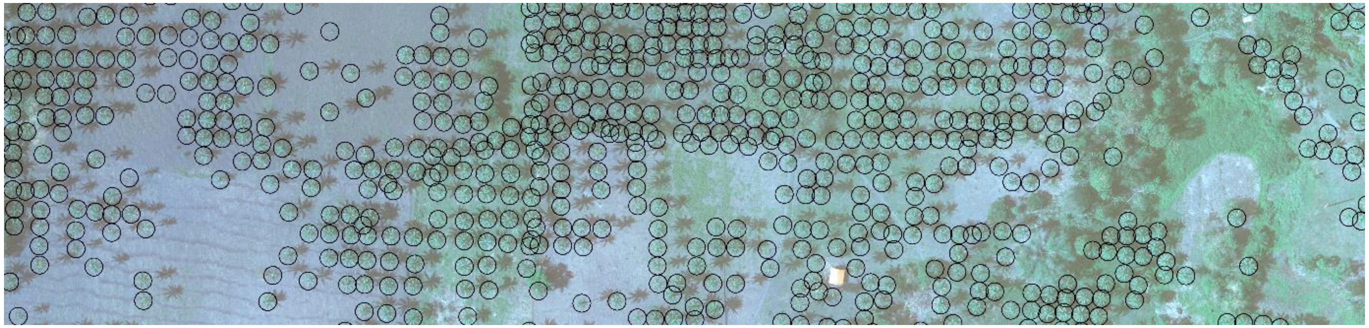


- **Our goal:** generate statistics on the number of coconut trees from these aerial images

KU LEUVEN

Introduction

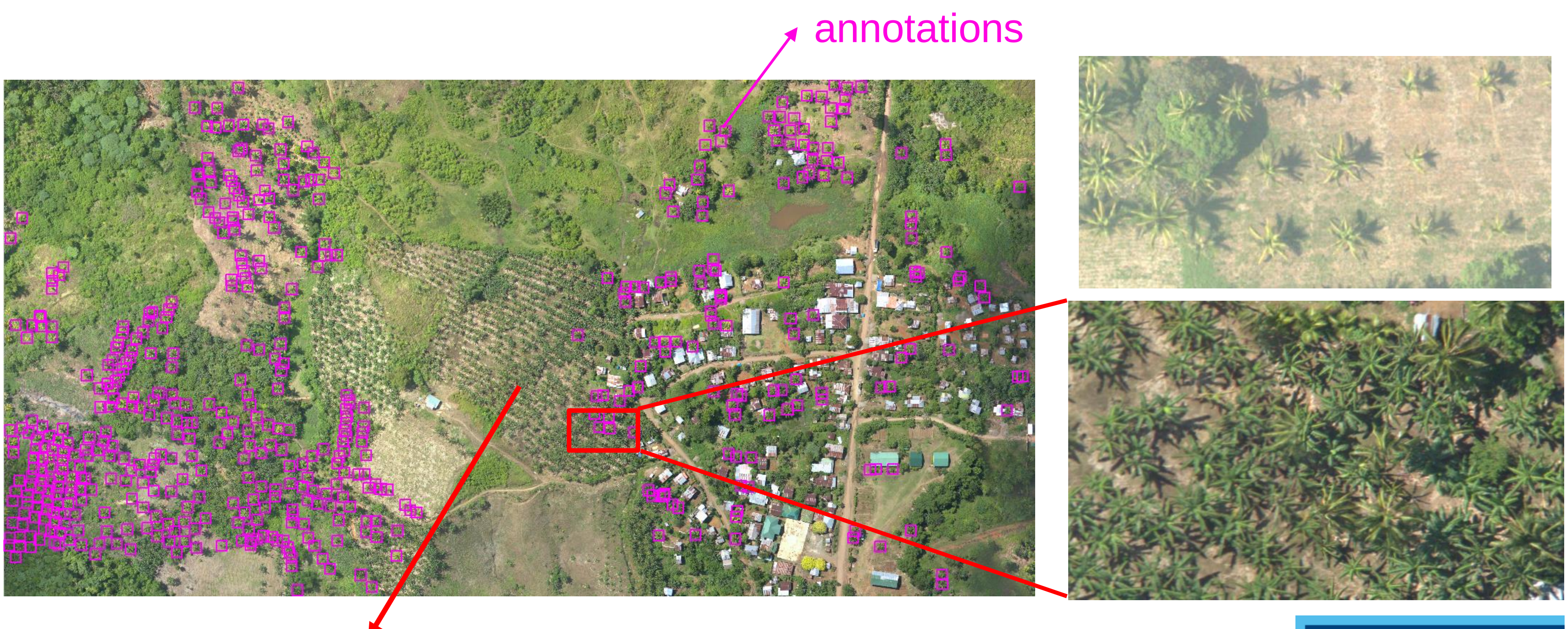
- Currently, this is done manually
  - Human annotators click coconut tree centers
    - ➔ Circle with predefined average diameter (fixed flying height)



  - Cumbersome, time-consuming and expensive
  - Avoid error and annotation bias: label same image with multiple annotators
  - Mistakes (forget trees, select wrong locations, …)

Challenges

- Perfect task to automate! Simple object detection task?
- Challenges:
  - Different vegetations, coconut trees in between other very similar vegetation, occluded under trees, not always strict pattern, different stages of growth,…



No coconut trees (look similar!)

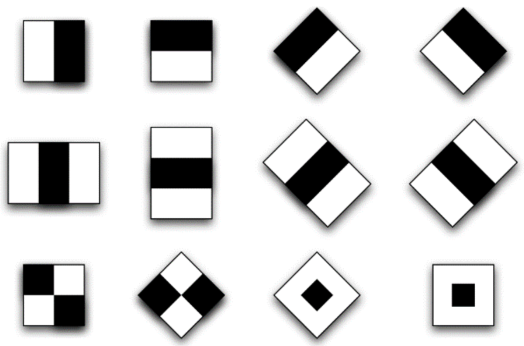
Approach

- Goal of this work: compare different object detection methodologies for reliable coconut tree counting

- Tailored towards ease-of-use for companies
- Accuracy, runtime, training time, number of training images,…

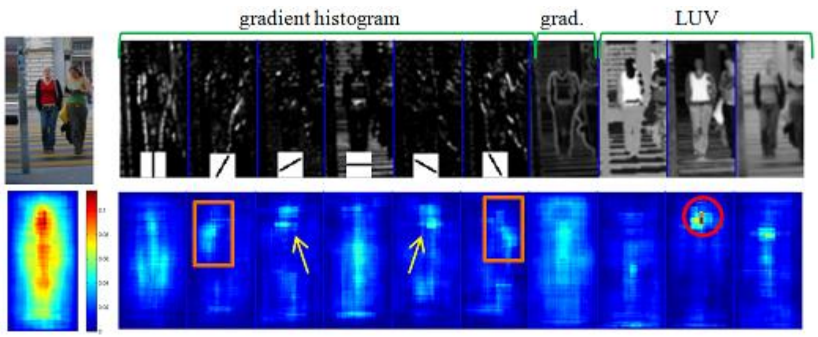
- We compare:
  - More *traditional* cascade classifier object detectors
  - With deep-learned object detectors

Related work



- Boosted cascade of weak classifiers
  - ➔ Viola & Jones (2001): Haar wavelets + AdaBoost
  - ➔ Early rejection of non-object patches, integral images

  +: Simple, fast -: no color, low accuracy?
  - ➔ Often improved with scene constraints and application specific constraints



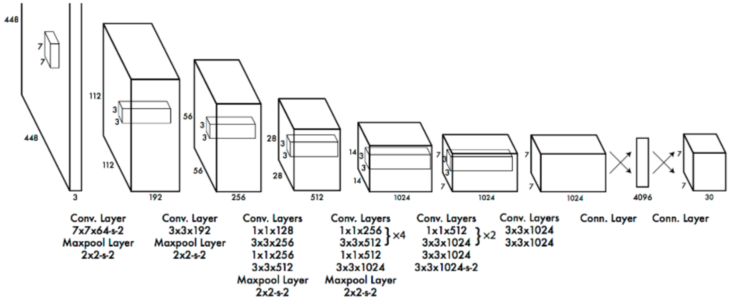
- ICF (Dollar et al., 2009)
  - ➔ Multiple features & color
  - ➔ Extension to ACF (2014): rectangles + approx. features

  +: Higher accuracy -: slower?

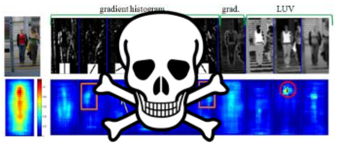
KU LEUVEN

Related work

- New trend since 2015: deep learning
  - Enormous datasets, drop in GPU hardware cost
  - Pre-trained nets AlexNet (2012), DenseNet (2014), ResNet (2016) ➔ top accuracy on ImageNet
  - From classification nets to detection: multi-scale sliding window ➔ computationally expensive
  - Region proposal networks –two parts which need to be tuned
  - Current trend: single-pass detectors SSD (2016), Yolo9000 (2017)
  - Real-time performance: 120 FPS @ VGA resolution



- Are V&J and ACF dead?



KU LEUVEN

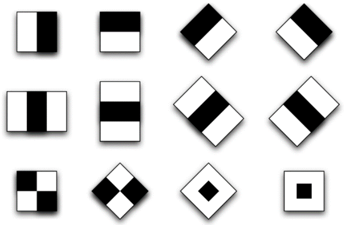
Dataset and frameworks

- A single 10.000 x 10.000 pixel image, RGB format
- Coconut trees: 100 x 100 pixels
- 3798 annotations

- Frameworks:
  - V&J: OpenCV3.2
  - ACF: internal C++ framework
  - InceptionV3: Tensorflow
  - C/CUDA darknet framework
    - Darknet19 & Densenet201

KU LEUVEN

Approaches with boosted cascades

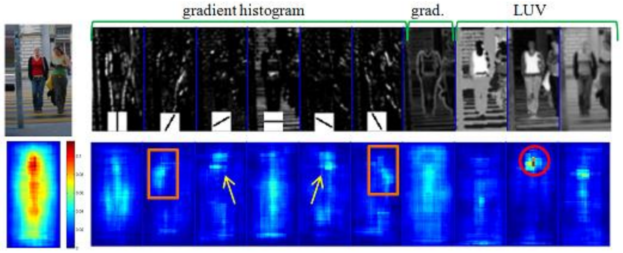


- First approach: V&J, 2001
    - Using LBP (Ahonen et al., 2004)
    - No color information (convert to grayscale images)
    - No obvious separation between coconut and background
        - ➔ otherwise first color transformation (e.g. solar panels)
    - Training: split image in four parts, train on top left, test others parts
    - Increase number of pos/neg samples for each model
    - Data augmentation: randomly flipping patches around vertical/horizontal axes
    - Single depth binary decision trees

	#pos	#neg	#weak	#feats
Model 1	1000	2500	16	126
Model 2	1000	5000	15	123
Model 3	1000	10000	15	142
Model 4	2000	8000	16	221

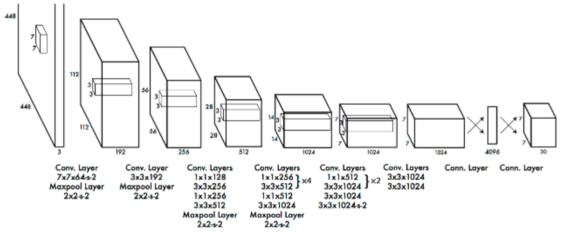
Approaches with boosted cascades

- Second approach: ACF, 2014
  - Add multiple channels and color
  - Initially trained on top left corner
  - ACF uses a lot more negatives
  - Not able to sample enough from top left corner
  - Split dataset: upper (1.741 positives) and lower half (1.914 positives)
  - Up to 150.000 negative patches



KU LEUVEN

Approaches with deep learning



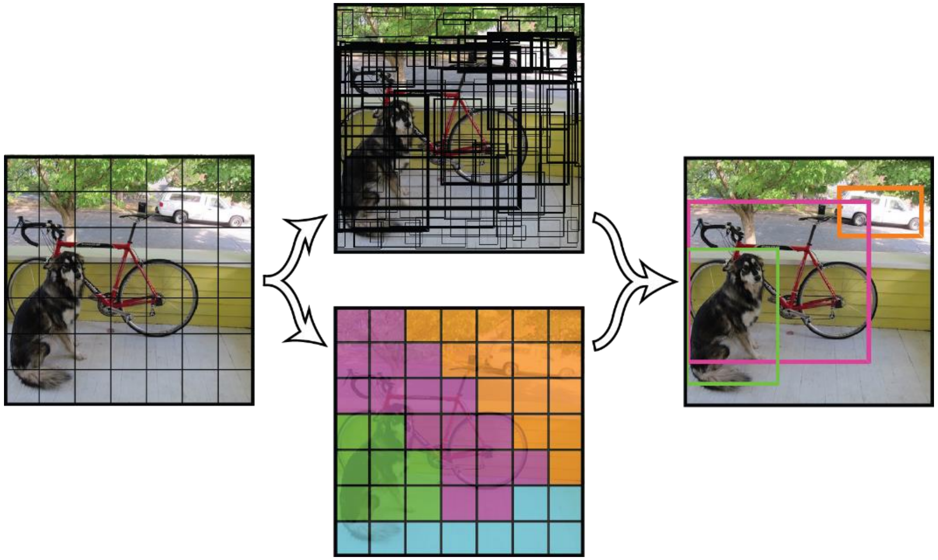
- Third approach: Deep learning, 2014
  - Most likely better accuracy
  - At which cost? Training time? Ease-of-use?

- Training with limited positives in three manners:
  - Learn a complete new deep network
    - ➔ Not advised, try to see what's possible
  - Freezing (n-1) layers, only retrain final layer
    - ➔ Transfer learning, only limited data required
    - ➔ Only works if new data relates to data of which initial model was trained
  - Fine-tuning weights of all layers
    - ➔ Again, limited training data needed
    - ➔ More flexible, new fine-tuned features for specific task

KU LEUVEN

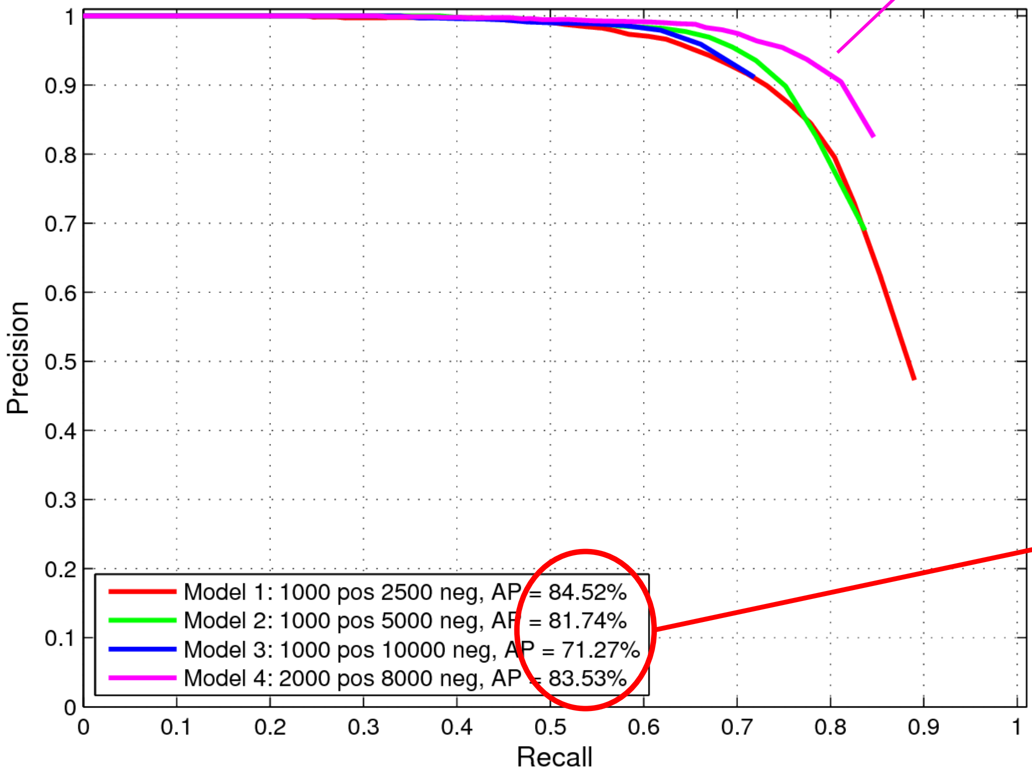
Approaches with deep learning

- We also tried a single-pass network (YoloV2)
  - Much faster than multi-scale sliding window
  - Coarse grid-based region proposals
    - ➔ Not able to cope with dense object packed scenes
    - ➔ In our case, objects close together and slightly overlapping
    - ➔ Final output detections cover multiple object instances



Results

- V&J



Not possible to generate more points with OpenCV

Even with limited training examples, still good accuracy (P=90%, R=80%)

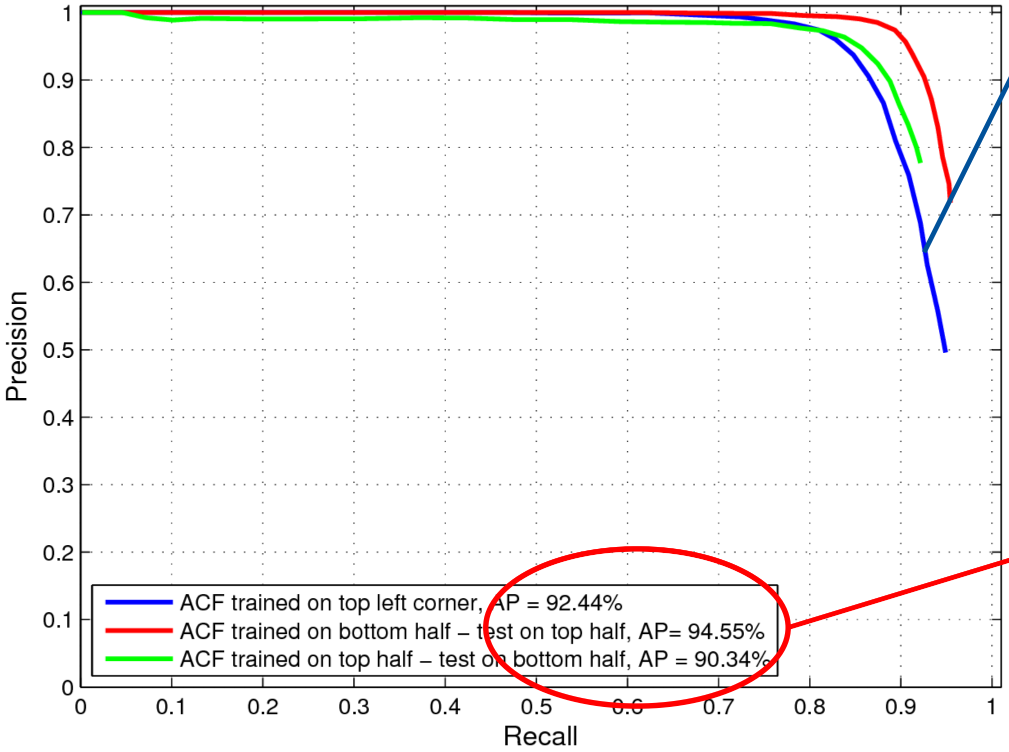
Influence of amount of training data

- Training time: 2 hours CPU only, evaluation: 10 minutes (10.000 x 10.000, Intel Xeon E5-2687W – 3.10 GHz)

KU LEUVEN

Results

- ACF



Model not optimal, trained on top left corner

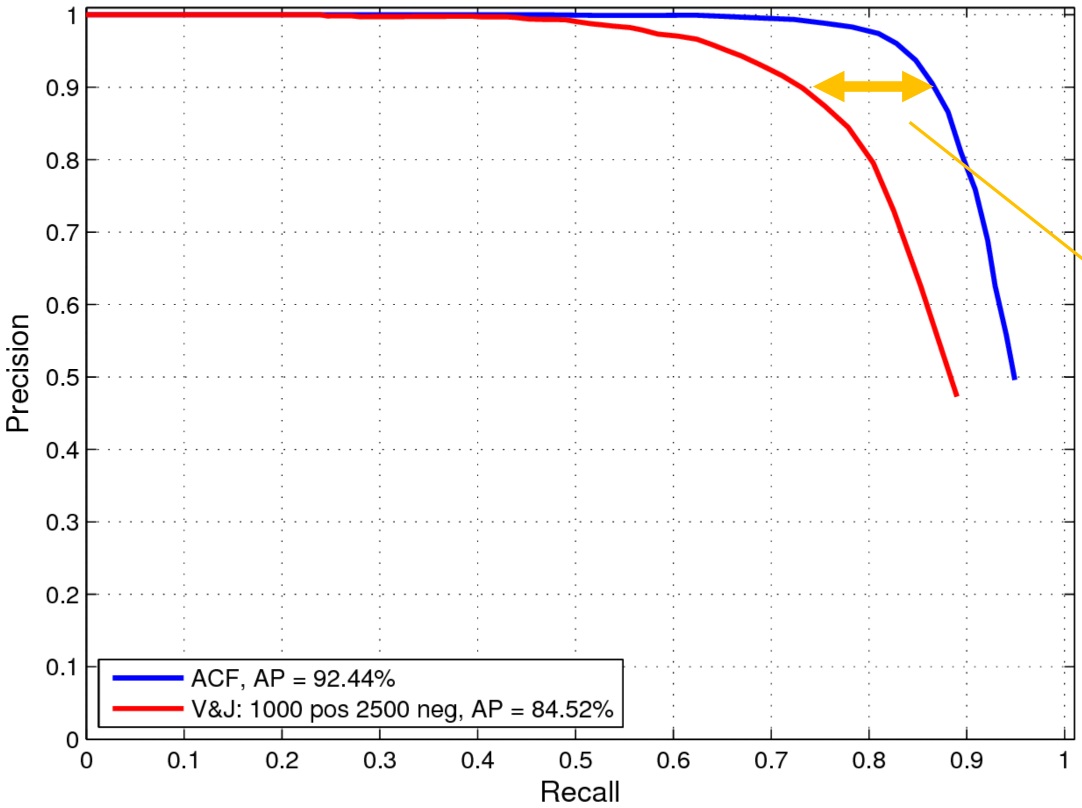
Uses color information, already much better (P=96%, R=90%)

Influence of training/test data

- Training time: 30 minutes CPU only, evaluation: 5 minutes (10.000 x 10.000, same hardware)

Results

- V&J versus ACF, both trained on top left corner



For same precision,
recall improves +- 12%

KU LEUVEN

Results

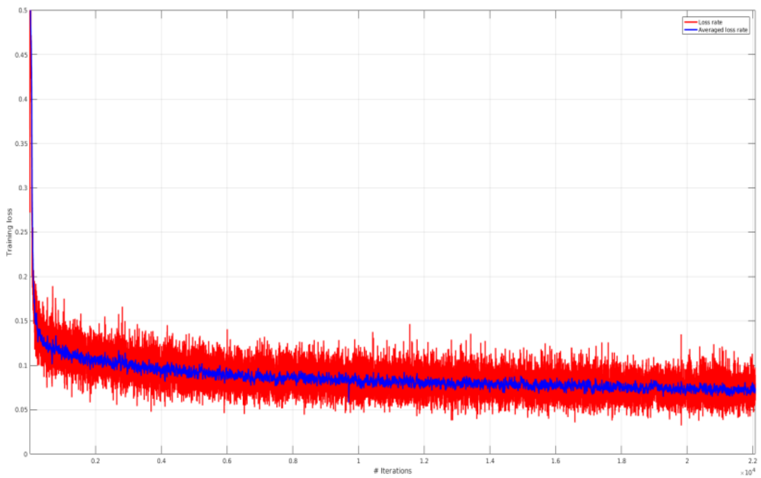
- Deep learning: classification networks
  - Train complete model from scratch
    - ➔ Model seems to converge (loss rate lowers)
    - ➔ Top-1 accuracy of 33% (two classes: coconut / background)
  - Transfer learning with frozen layers
    - ➔ InceptionV3 in TensorFlow, 75 positive examples / 75 background examples
    - ➔ Top-1 accuracy of 77%
    - ➔ Compare with boosted cascade: evaluation at pixel level: P=75%, R=52%
  - Transfer learning by fine tuning layers
    - ➔ Darknet19 and Densenet201
    - ➔ Trade-off between accuracy and inference time

Results

- Transfer learning by fine tuning layers
  - ➔ Darknet19: 10.000 iterations, Top-1 accuracy of 95.2%
  - ➔ Densenet201: 20.000 iterations, Top-1 accuracy of 97.4%
  - ➔ Training takes multiple hours (24h for Darknet19)

Darknet19

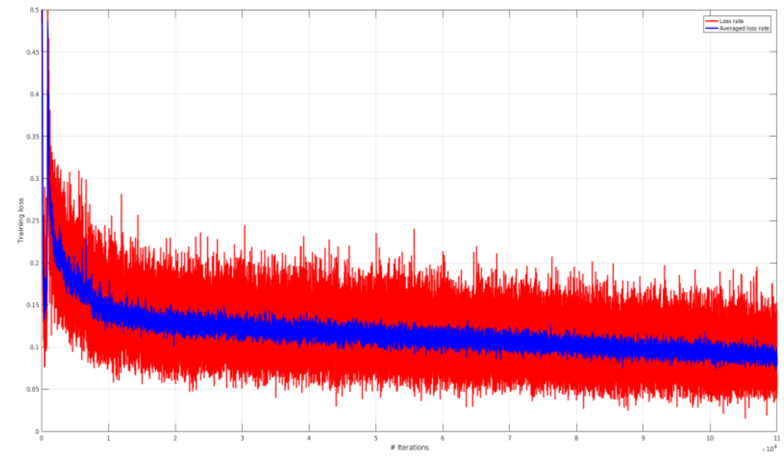
Loss-rate



Iterations

Densenet201

Loss-rate



Iterations

Results

- Deep learning: execution speeds
  - Classification on NVIDIA TitanX
    - ➔ Darknet19: 100x100 pixel patches: 265 FPS
    - ➔ Densenet201: 52 FPS
    - ➔ Memory footprint only 400MB
  - Detection: multi-scale not needed
    - ➔ Sliding window evaluated over different step sizes
    - ➔ Achieves excellent accuracy of P=97.31%, R=88.85%

step	#patches	Darknet19	Densenet201
5px	3.924.361	4h	20h30m
25px	157.609	9m5s	50m20s
50px	39.601	2m30s	12m35s

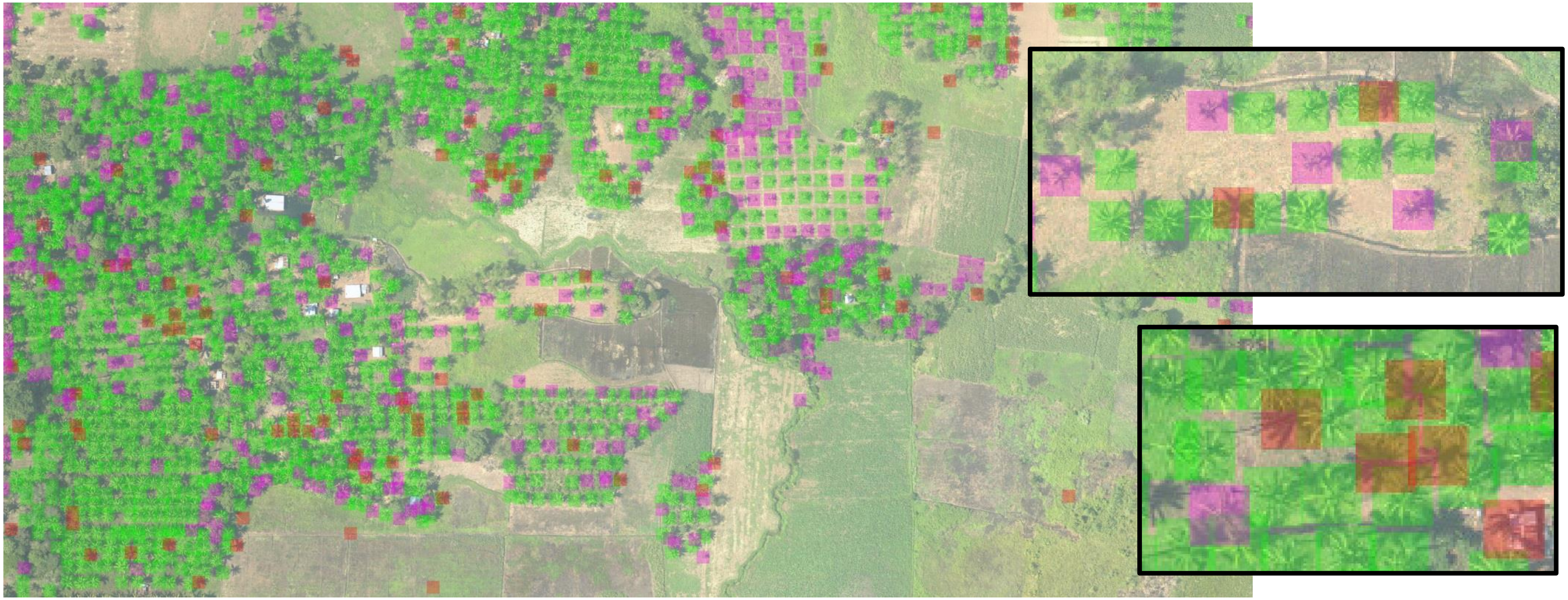
V&J: 10 min
ACF: 5 min

KU LEUVEN

Visual results: V&J

Model	Precision	Recall	Train	Infer
V&J	90.64%	81.12%	2h	10m
ACF	90.55%	86.43%	30m	5m
DN19	97.31%	88.58%	24h	2m30s

Green, TP – Red, FP – Magenta, FN



➔ High FP rate, especially on shadows (no color information)
➔ Several FN (smaller trees)

KU LEUVEN

Visual results: ACF

Model	Precision	Recall	Train	Infer
V&J	90.64%	81.12%	2h	10m
ACF	90.55%	86.43%	30m	5m
DN19	97.31%	88.58%	24h	2m30s

Green, TP – Red, FP – Magenta, FN

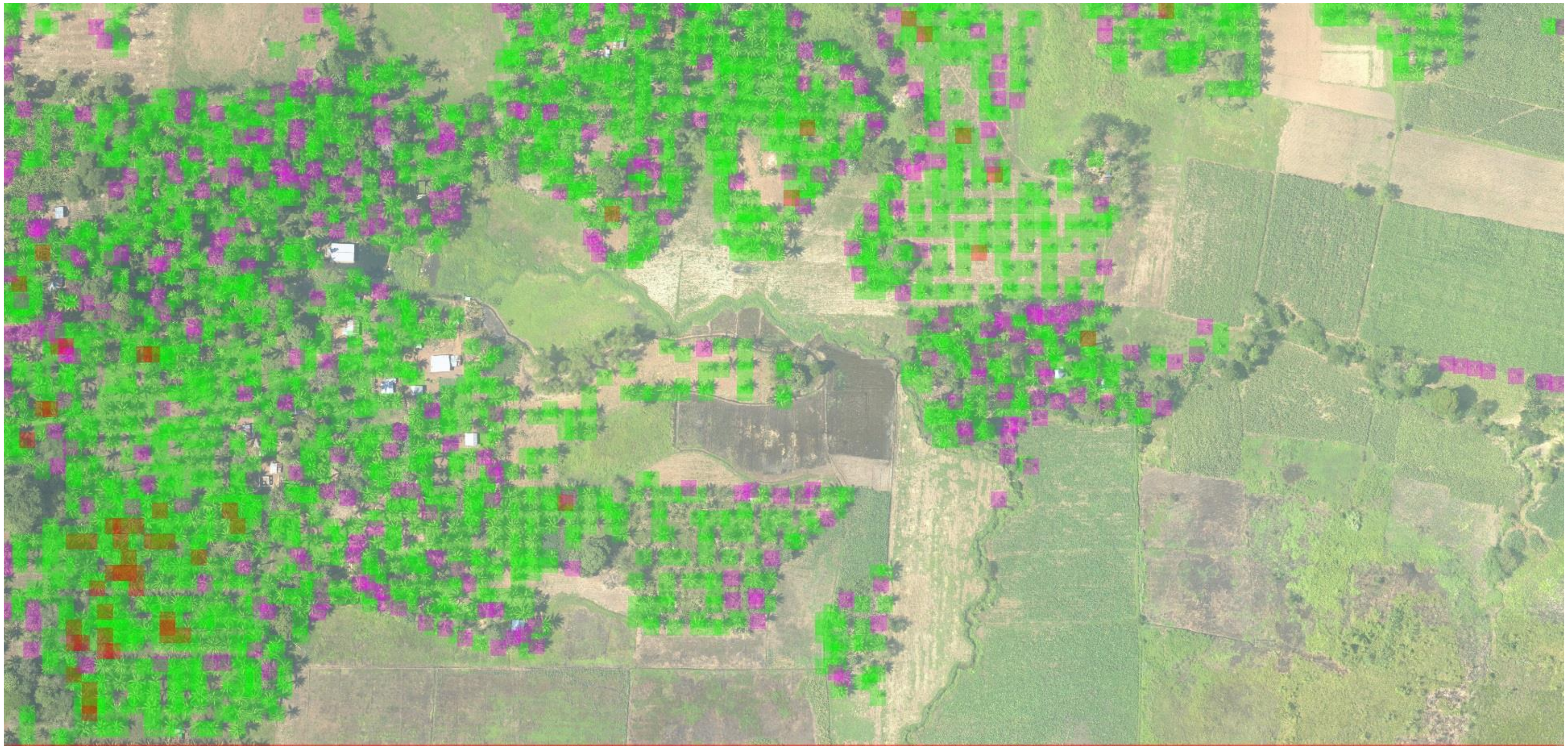
- ➔ About equal amount of FP: no shadows but in between trees
- ➔ Higher recall (less FN) – FN again on smaller trees ➔ train separate model?

KU LEUVEN

Visual results: DL

Model	Precision	Recall	Train	Infer
V&J	90.64%	81.12%	2h	10m
ACF	90.55%	86.43%	30m	5m
DN19	97.31%	88.58%	24h	2m30s

Green, TP – Red, FP – Magenta, FN



- ➔ Almost no FP
- ➔ Again FNs: train separate model? – reduce step size (50px here)?

Conclusion

- Evaluated the capability of older boosted cascaded object detectors and deep learning for coconut tree detection

- Best cascaded: 94.56% AP, 5-10 min evaluation
- Best deep learning: 97.4% Top-1 accuracy, 2m30 – 4h evaluation

- Are VJ & ACF dead? 

- Accuracy of ACF slightly lower than DL
- Evaluation time: depends on step size
- Training time and required hardware BIG difference (ACF wins)

ACF is dead, long live deep learning!

Otherwise, long live ACF (if hardware is an issue)

Future work

- Combine region proposal networks with deep learning
  - Lower number of candidate patches

- Combine both deep learning and boosted cascades
  - Use principle of boosted cascaded where the weak classifiers are built using small convolutional neural networks

Questions?

- Thank you for your attention!
- Contact:
  - http://www.eavise.be/
  - kristof.vanbeeck@kuleuven.be
  - steven.puttemans@kuleuven.be
  - toon.goedeme@kuleuven.be

EAVISE

Embedded & Applied Vision Engineering