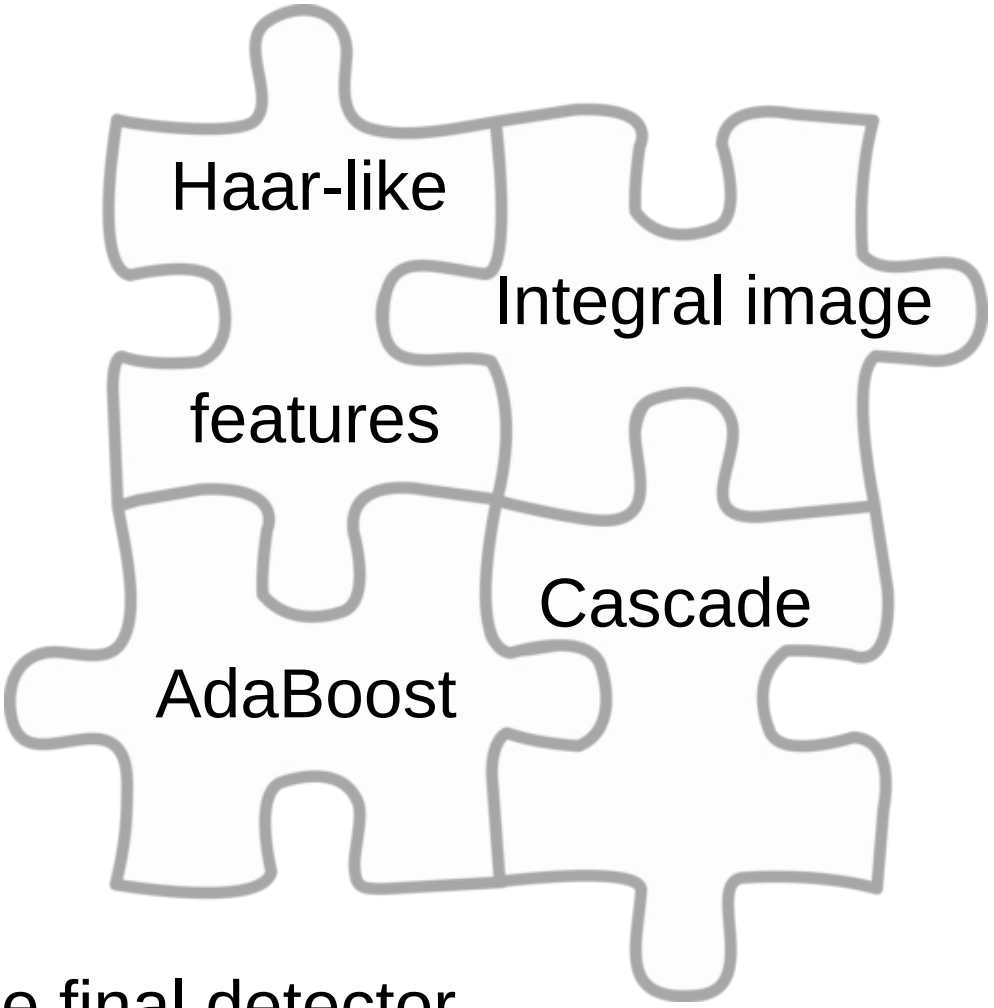
Real-time face detection with Haar cascades

Nora Kassner

Outline

- The Algorithm:

  2001 by Paul Viola & Michael Jones

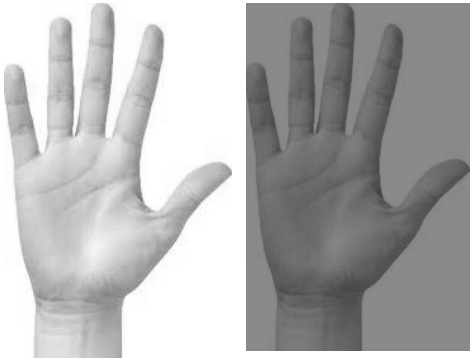


- Training & the final detector
- OpenCV

Choice of features

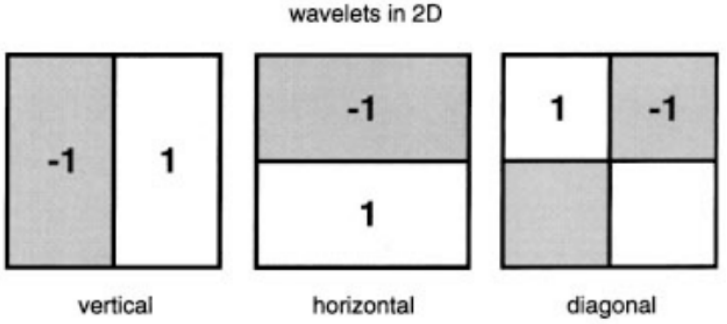
- How many features should be used?
- What makes a good feature?

- Pixels



- Histograms

- Haar-like features: local, oriented intensity differences

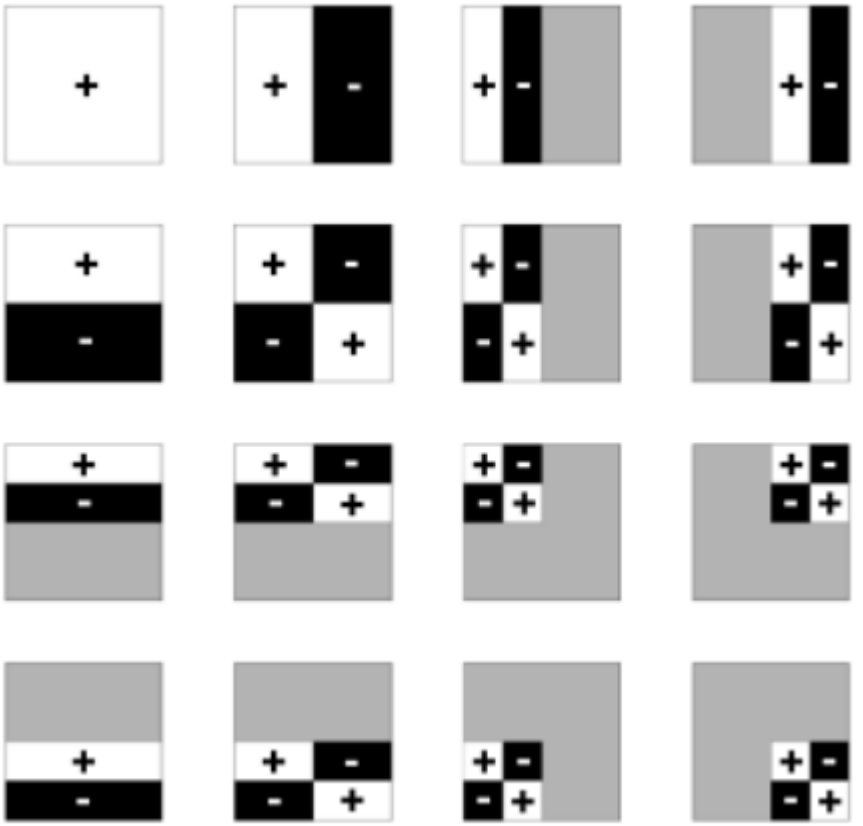


Haar wavelets

Original-Matrix

10	14	7	5
16	12	13	19
14	12	1	5
8	2	3	3

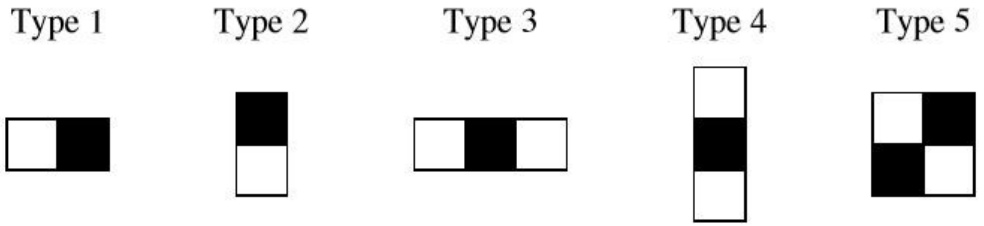
9	2	1	-1
3	-1	-1	0
-3	2	-2	2
2	2	-1	-1

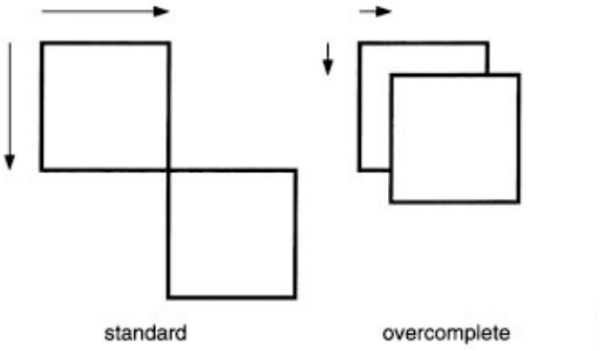


Zweidimensionale Basisfunktionen für ein 4×4-Bild

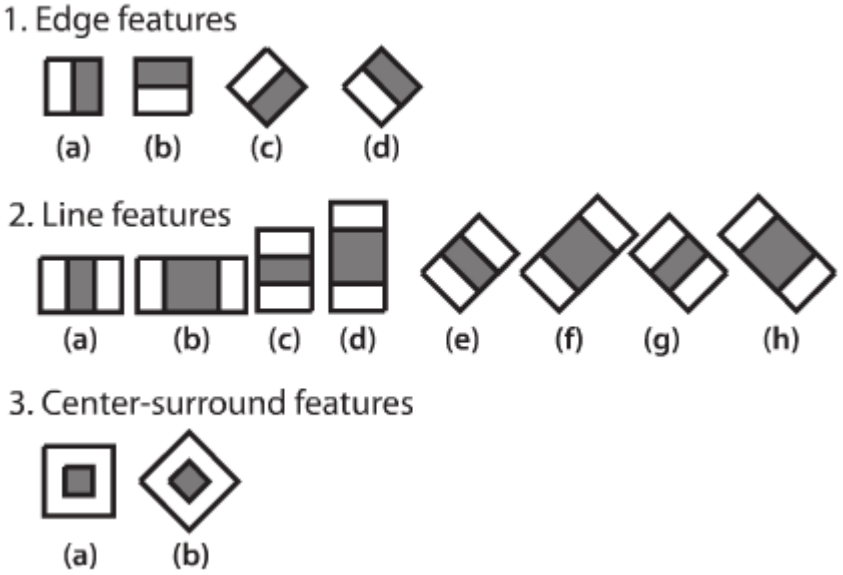
http://www.tilman.de/uni/ws05/scivis/2d-transformation.html

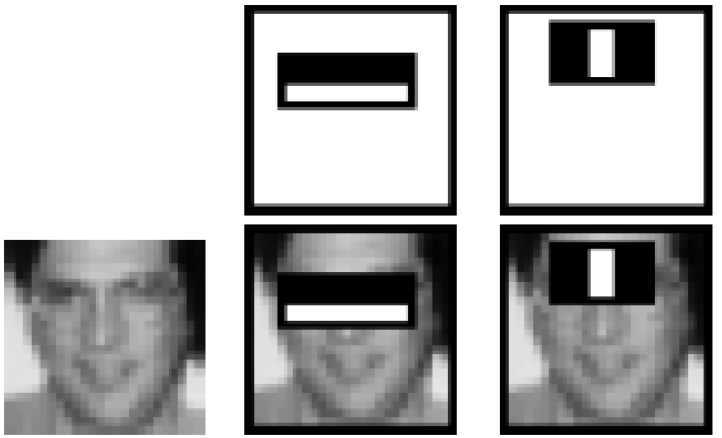
Haar-like features





→ 160.000 features per 24x24 px window





→ Features encode knowledge

→ Sensitive to edges, bars, simple structure

Integral Image

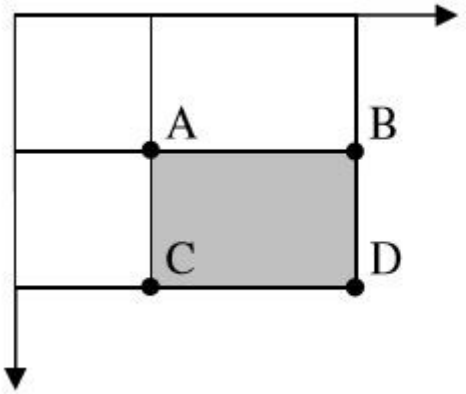
→ Very fast

1	1	1
1	1	1
1	1	1

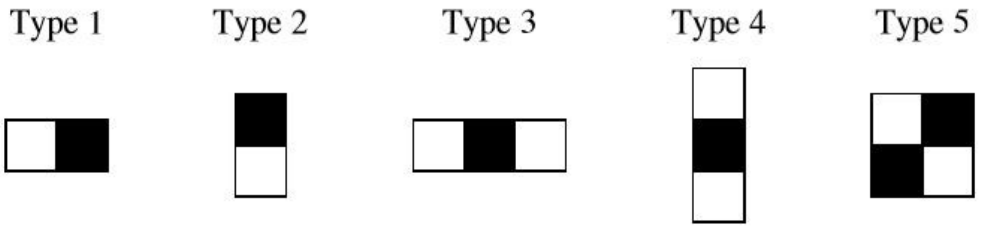
Input image

1	2	3
2	4	6
3	6	9

Integral image



Sum of grey rectangle = D - (B + C) + A



Number of array references: 6 8 9

AdaBoost

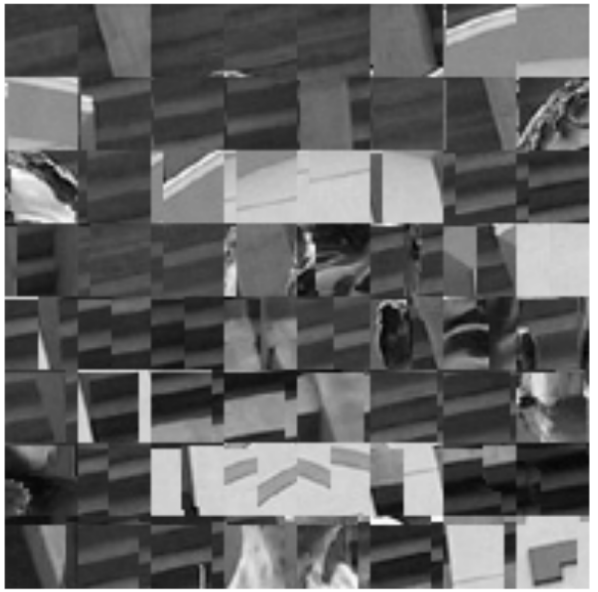
→ narrowing down number of features to only a few useful ones

- Weak classifier: perform at least better than random:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) > p\theta \\ 0 & \text{otherwise} \end{cases}$$

- Combining weak classifiers in a weighted sum to form a strong classifier:

$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$



AdaBoost

- Given examples images $(x_1,y_1),...,(x_n,y_n)$ where $y_1=0,1$ for negative and positive examples.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_1=0,1$, where m and l are the numbers of positive and negative examples.
- For $t=1,...,T$:

1) Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

2) Select the best weak classifier with respect to the weighted error:

$$\varepsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|$$

3) Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t and θ_t are the minimizers of ε_t.

4) Update the weights:

$$w_{t+1,i} = w_{t,i} \beta^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$

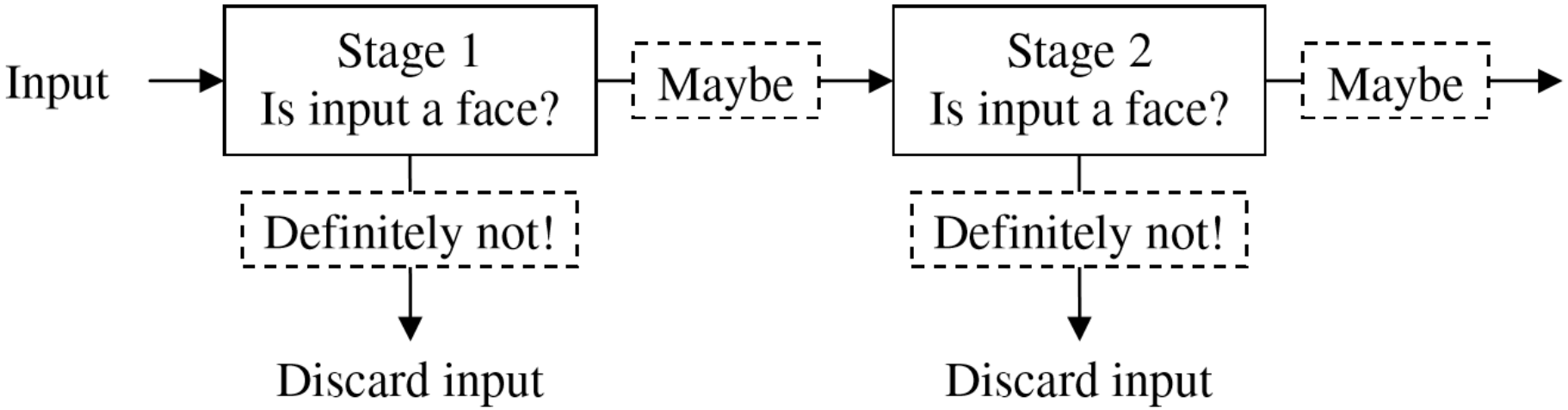
- The final strong classifier is:

$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

The cascade

→ Focus of attention



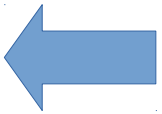
Training the cascade

- AdaBoost

  → minimize false negative

- Parameters:

  - # stages
  - # features per stage
  - Threshold of each stage



Select:

- Max. false positive / stage
- Min. true positive / stage
- Target overall false positive

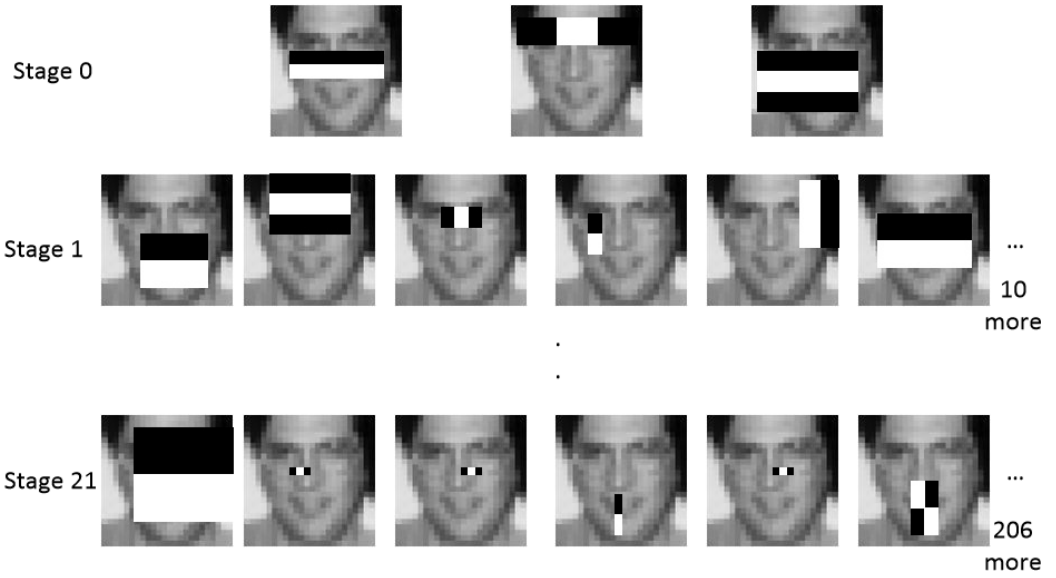
See how the cascade looks like:
http://www.makematics.com/research/viola-jones/

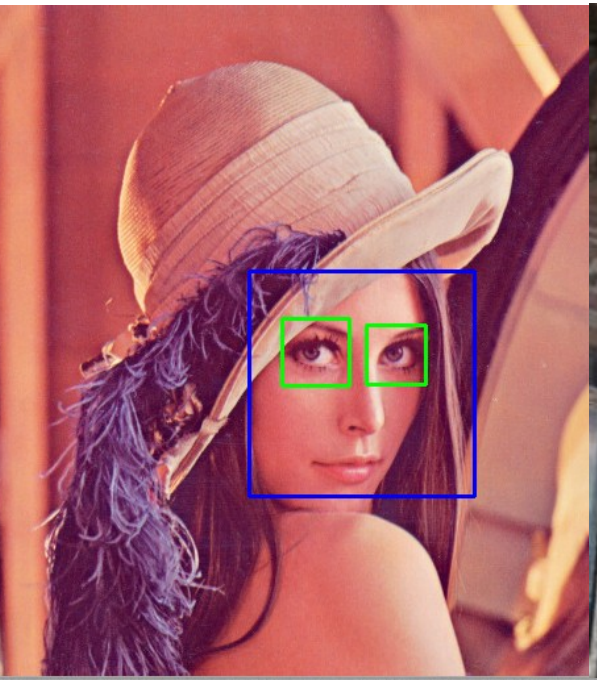
Training the Cascade

- User selects values for f, the maximum acceptable false positive rate per layer and d, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate, F_{target}.
- P = set of positive examples
- N = set of negative examples
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$
  - $i \leftarrow i + 1$
  - $n_i = 0$; $F_i = F_{i-1}$
  - while $F_i > f \times F_{i-1}$
    - $n_i \leftarrow n_i + 1$
    - Use P and N to train a classifier with n_i features using AdaBoost
    - Evaluate current cascaded classifier on validation set to determine F_i and D_i.
    - Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects F_i)
  - $N \leftarrow \emptyset$
  - If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N

The final detector

- 6000 features
- 38 stages
- Input parameters:
  - Cascade containing features
  - Starting scale
  - Starting delta
  - Scale increment
- 15 frames/s





OpenCV

- Free for academic & commercial use
- Link installation instruction: http://docs.opencv.org/master/d9/df8/tutorial_root.html
- C++, C, Python and Java interfaces
- Supports Windows, Linux, Mac OS, iOS and Android

Stuff you can do with it:

https://www.youtube.com/watch?v=oJAl9Yd3kNo

https://www.youtube.com/watch?v=8h9vU1pnNZA

Do it yourself!

```
nora@NorasT520 ~/OpenCV/TrainCars $ ls
bg.txt  cars.info  neg  pos
nora@NorasT520 ~/OpenCV/TrainCars $ opencv_createsamples -info cars.info -num 500 -w 48 -h 24 -vec cars.vec
```

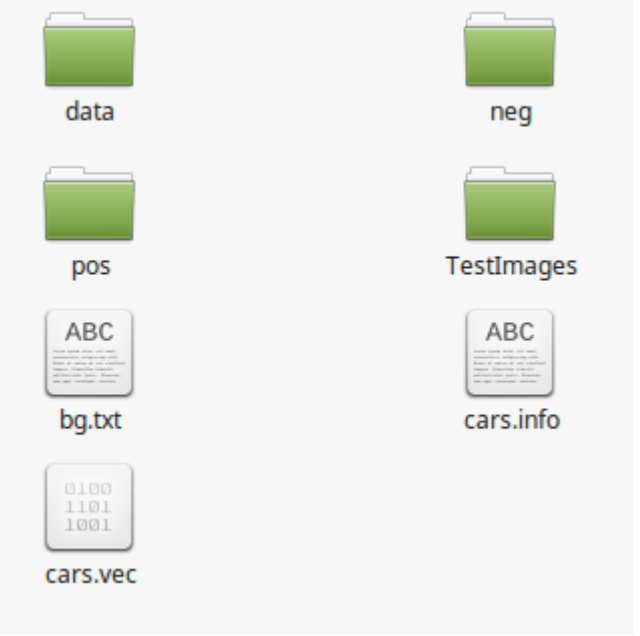
```
nora@NorasT520 ~/OpenCV/TrainCars $ opencv_traincascade -data data -vec cars.vec -bg bg.txt -numPos 500 -numNeg 500
-numStages 10 -w 48 -h 24
```

bg.txt

```
/home/nora/OpenCV/Training/background/image_0001.jpg
/home/nora/OpenCV/Training/background/image_0002.jpg
/home/nora/OpenCV/Training/background/image_0003.jpg
/home/nora/OpenCV/Training/background/image_0004.jpg
```

cars.info

```
pos/pos-532.pgm 1 0 0 100 40
pos/pos-166.pgm 1 0 0 100 40
pos/pos-76.pgm 1 0 0 100 40
pos/pos-193.pgm 1 0 0 100 40
pos/pos-0.pgm 1 0 0 100 40
```

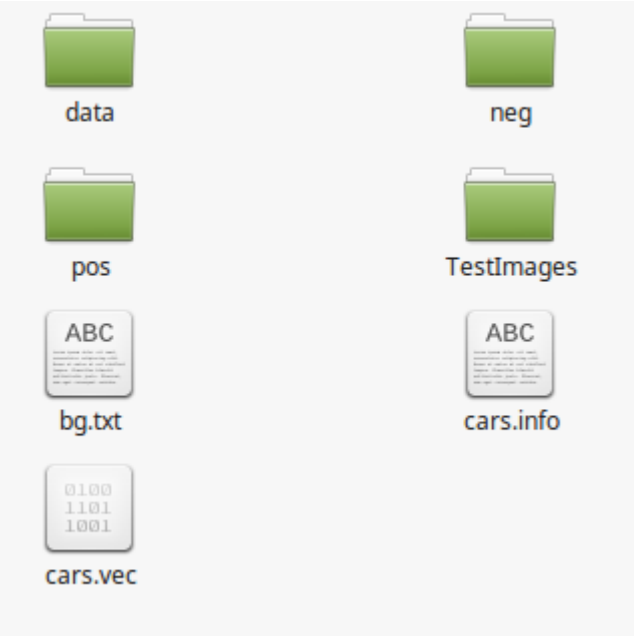


Do it yourself!

```
nora@NorasT520 ~/OpenCV/TrainCars $ ls
bg.txt  cars.info  neg  pos
nora@NorasT520 ~/OpenCV/TrainCars $ opencv_createsamples -info cars.info -num 500 -w 48 -h 24 -vec cars.vec
```

```
nora@NorasT520 ~/OpenCV/TrainCars $ opencv_traincascade -data data -vec cars.vec -bg bg.txt -numPos 500 -numNeg 500
-numStages 10 -w 48 -h 24
```

http://imgur.com/gallery/QWQiBYU



Sources

- P. Viola, M. Jones: Rapid Object Detection using a Boosted Cascade of Simple Features, Conference Paper in Computer Vision and Pattern Recognition, 2001, Vol.2.
- R. Lienhart, J. Maydt: An Extended Set of Haar-like Features for Rapid Object Detection, Conference Paper in Proceedings / ICIP ... International Conference on Image Processing 1, 2002, Vol. 1.
- O. Jensen: Implementing the Viola-Jones Face Detection Algorithm, Master thesis 2008, Technical University of Denmark, Informatics and Mathematical Modelling.
- A. Barczak, F. Dadgostar: Real-time hand tracking using a set of cooperative classifiers based on Haar-like features, Res. Lett. Inf. Math. Sci., 2005, Vol. 7.
- C. Papageorgiou, T. Poggioa: Trainable System for Object Detection, International Journal of Computer Vision, 2000, Vol. 38.1.
- http://docs.opencv.org/master/index.html
- https://www.cs.auckland.ac.nz/~rklette/CCV-CIMAT/pdfs/B27-Haar_VJ_AB.pdf
- https://en.wikipedia.org/wiki/Haar_wavelet
- http://www.makematics.com/research/viola-jones/
- http://www.tilman.de/uni/ws05/scivis/wavelet-transformation.html