

Détection et reconnaissance visuelle temps-réel d'objets d'une catégorie (piétons, voitures, etc...)

+ Présentation de l'outil « SEVILLE »
(Semi-automatic VISuaL LEarning)

Fabien Moutarde
Centre de Robotique (CAOR)
MINES ParisTech (ENSMP)

Fabien.Moutarde@mines-paristech.fr

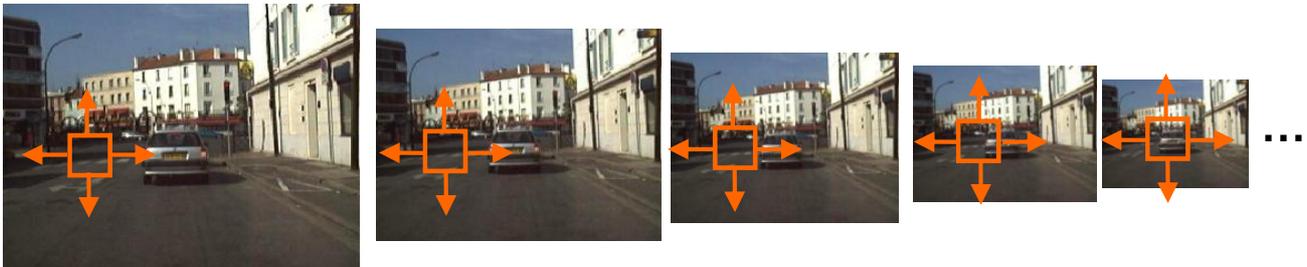
Détection et reconnaissance visuelle d'objets (piétons, voitures,...)



- Composant-clef pour beaucoup d'Aides à Conduite (ADAS), e.g. :
 - détection de véhicule précédent pour ACC
 - détection piétons pour Evitement Collision
- Forte contrainte de vitesse de calcul pour appli "voiture intelligente" : **traiter au moins ~10 frames/seconds**

Détection multi-résolutions avec un classifieur unique (« window-scanning »)

- Pour chaque image traitée de la vidéo :
 - construire une pyramide de ~12 images par sous-échantillonnages
 - scanner chaque image de la pyramide avec une fenêtre de détection de taille fixe (e.g. 36x36 pixels pour la détection de vue arrière de voiture)
 - plusieurs dizaines de milliers d'images correspondant à des sous-fenêtres de tailles et positions diverses dans l'image initiale



- avec un unique classifieur, évaluer pour chacune de ces images si elle est correctement centrée sur un objet du type cherché (e.g. vue arrière d'une voiture)

→ Besoin uniquement d'un classifieur objet_cherché/autre pour images de taille fixée (e.g. 36x36 pixels)

Apprentissage pour la reconnaissance visuelle de catégories d'objets

Piétons



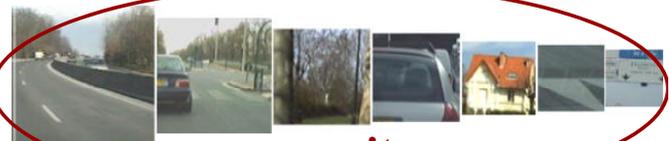
« non-piétons »



Voitures (vue arrière)

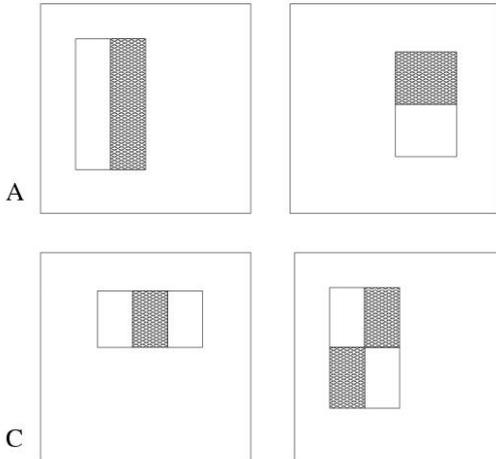


« non-voitures »



Apprentissage de classifieurs sur des bases d'images extraites de vidéos

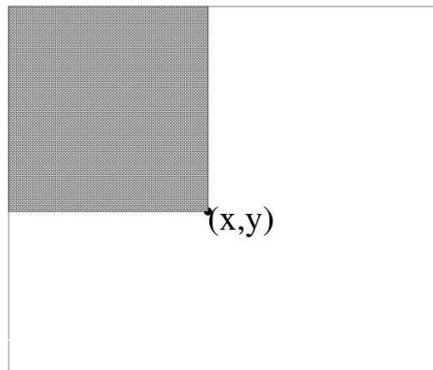
The Viola & Jones features for object detection



- 3 rectangular feature types:
 - *two-rectangles feature types* (horizontal/vertical)
 - *three-rectangles feature type*
 - *four-rectangles feature type*

Using a 24x24 pixel base detection window, with all the possible combination of horizontal and vertical location and scale of these feature types the full set of features has 45,396 features (and ~10 times more in a 32x32 window)

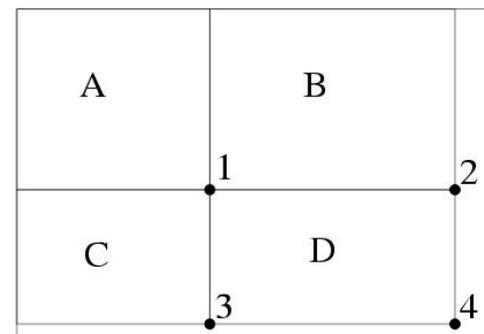
Integral image



- **Definition:** The *integral image* at location (x,y) , is the sum of the pixel values above and to the left of (x,y) , inclusive.
- It can be computed in one single pass with nb_pixels additions.

Using the integral image representation one can compute the value of any rectangular sum in constant time.

For example the integral sum inside rectangle D we can compute as: $ii(4) + ii(1) - ii(2) - ii(3)$

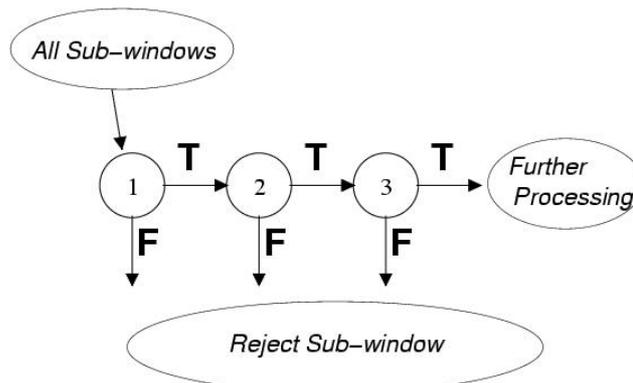


AdaBoost as feature selector

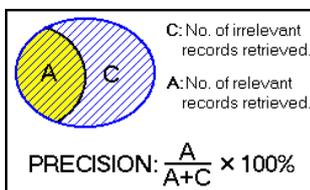
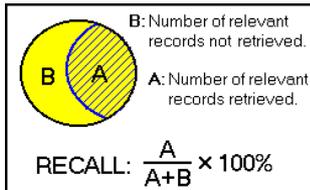
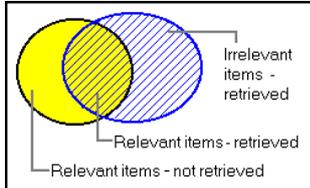
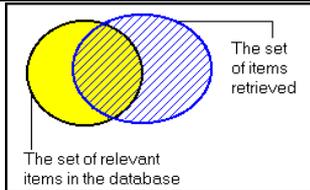
- Given a feature set and labeled training set of images one can apply number of machine learning techniques.
- Recall however, that there is 45396 features associated with each image sub-window, hence the computation of all features is computationally prohibitive.
- **Hypothesis:** A combination of only a small number of these features can yield an effective classifier.
- **Challenge:** Find these discriminant features.
- Use AdaBoost by testing all the features:
 - Learning is slow
 - But detection will be fast ! (well, no so....)

Speed-up through the Attentional Cascade

- Simple, boosted classifiers can reject many negative sub-windows while still detecting all positive instances
- Series of such simple classifiers can achieve good detection performance while eliminating the need for further processing of negative sub-windows.



Recall and Precision



- **Recall:** The percentage of the total relevant documents in a database retrieved by your search.

If you knew that there were 1000 relevant documents in a database and your search retrieved 100 of these relevant documents, your recall would be 10%.

- **Precision:** The percentage of relevant documents in relation to the number of documents retrieved.

If your search retrieves 100 documents and 20 of these are relevant, your precision is 20%.

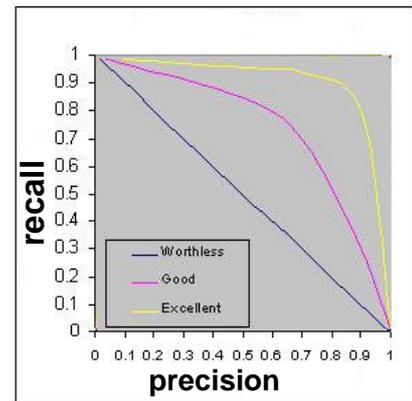
Recall and precision formulas

	predicted as positive	predicted as negative
positive	TP	FN
negative	FP	TN

$$\begin{aligned} \text{Recall} \\ \text{Sensitivity} \\ \text{True Positive rate} \end{aligned} = \frac{\text{Nb of } \underline{\text{correct}} \text{ positive predictions}}{\text{Nb of } \textit{real} \text{ positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

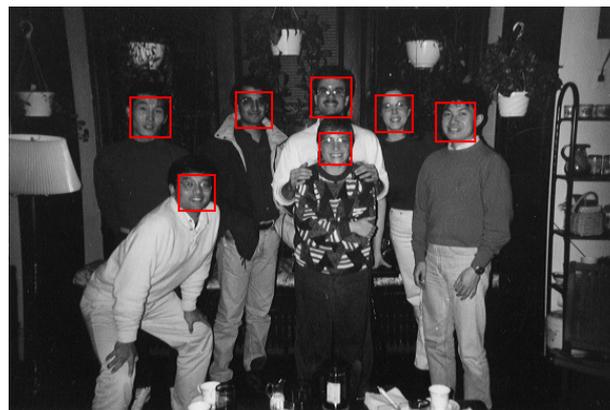
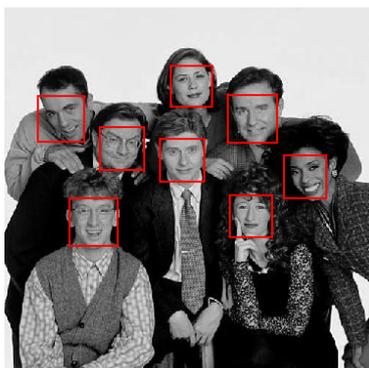
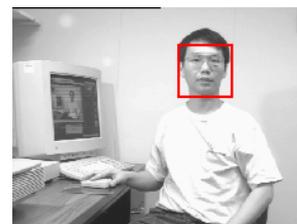
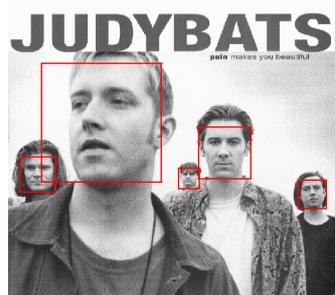
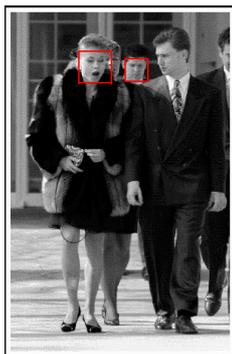
$$\begin{aligned} \text{Precision} \\ \text{(wrt positives)} \end{aligned} = \frac{\text{Nb of } \underline{\text{correct}} \text{ positive predictions}}{\text{Nb of positive } \textit{predictions}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- C1 predicts better than C2 if C1 has better recall *and* precision than C2
- There is a trade-off between recall and precision



→ Compare precision-recall curves!

**For numeric comparison (or if curves cross each other),
Area Under Curve (AUC)**

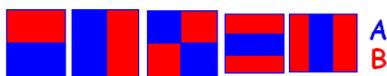


- The paper by Viola & Jones presents general object detection method which is illustrated on the face detection task.
- Using the integral image representation and simple rectangular features eliminate the need of expensive calculation of multi-scale image pyramid.
- Using AdaBoost gives a general technique for efficient feature selection.
- A general technique for constructing a cascade of homogeneous classifiers is presented, which can reject most of the negative examples at early stages of processing thereby significantly reducing computation time.
- A face detector using these techniques is presented which is comparable in classification performance to, and orders of magnitude faster than the best detectors known before.

BUT: limitation in the type of features used (essentially capture vertical/horizontal contrasts, contrast-sensitive, ...)

Main families of boosting Weak Classifiers

- Haar-like (Viola-Jones) = most commonly used features



if $|SumPixels(A) - SumPixels(B)| > Threshold$ then True
else False

- 👍 Relatively fast computation with integral image
- 👎 Mostly based on horizontal/vertical contrasts

Some work showed improved results with extended feature set [Treptow & Zell, CEC'2004]

- HOG (Histogram of Oriented Gradient) – based features

[Zhu et al., CVPR'2006, Mitsubishi] [Pettersson et al., IV'2008, NICTA]

- 👍 More detailed/discriminative information
- 👎 Tricky to make it fast enough
- 👎 Not so good results on object classes with too shallow gradients

- Pixel-pairs comparisons

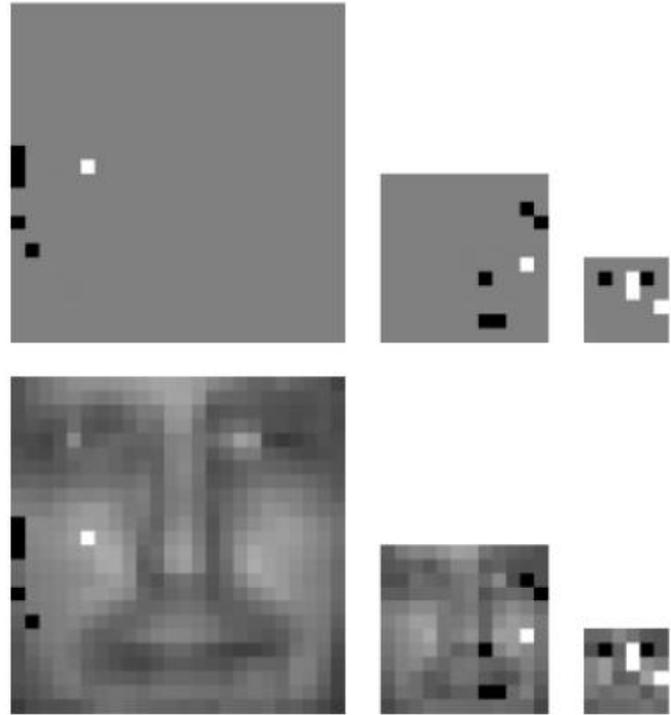
[Baluja et al., ICIP'2004, Google/CMU] [Leyrit et al., IV'2008, LASMEA]

- 👍 Extremely low computation time
- 👎 Less discriminative → more WC, or more complex classif required

- Control-points features [CAOR/Mines ParisTech work since 2004]

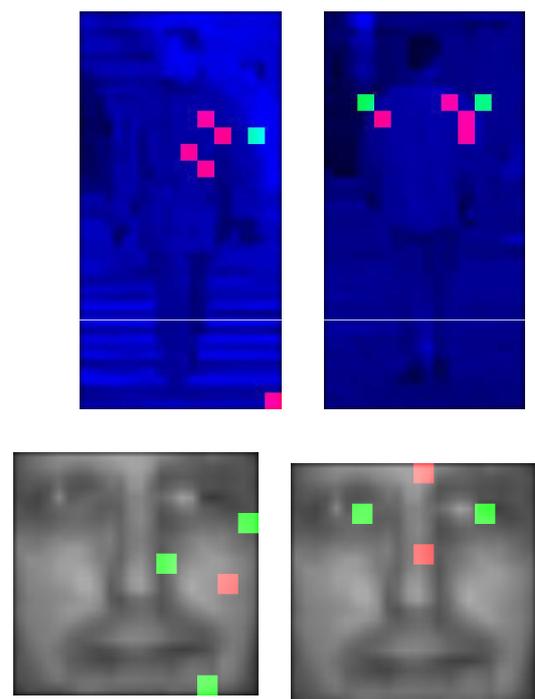
Les “control-point” features (inventés au CAOR/Mines ParisTech !!)

- Analyse imagette dans une des 3 résolutions parmi : full, half, quarter
- Deux ensembles de points (6 maxi par ensemble)
- Chaque point est un pixel (à la résolution choisie de l’imagette)
- Répond “OUI” si TOUS les points “blancs” ont des valeurs de pixel supérieures à TOUTES celles des “noirs”



Avantages des “control-points”

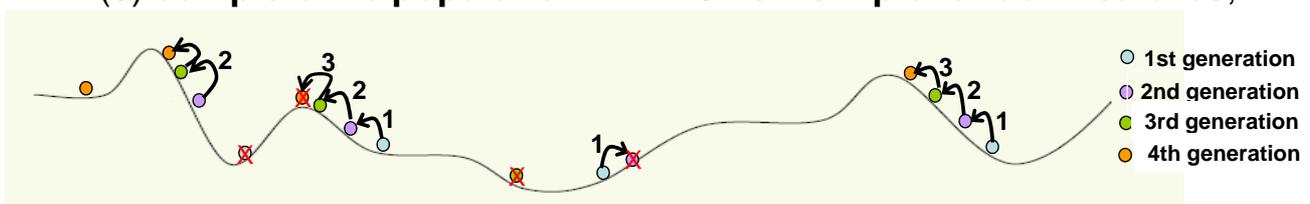
- Plus indépendant de la luminosité que features utilisant des différences de somme de valeur de pixels (pas besoin de normalisation préalable de l’histogramme)
- Calculable TRES efficacement : seulement 3-4 opérations d’accès aux pixels de l’imagette (2 fois plus rapide que features type “ondelettes Haar” utilisés par Viola&Jones)
- Utilise peu de mémoire auxiliaire



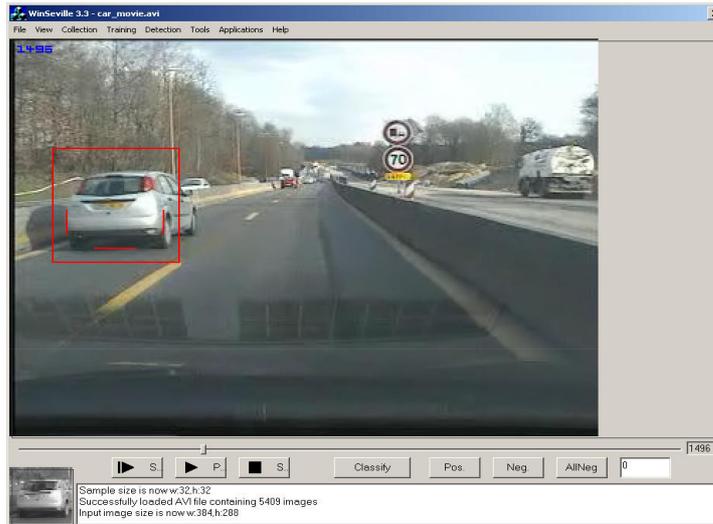
- **Control-Points features = huge family**
(> 10^{35} possible different CP-features !)
==> exhaustive search absolutely impossible
- **Dependance of CP-feature error on database with CP-feature parameters wildly variable**
(very chaotic « fitness landscape »)

Evo-HC: hybrid evolutionary / Hill-Climbing search Weak-Learner

- **start with a first generation of 70 random simple features** (i.e. with only 2 positive and 2 negative points);
- **at each generation:**
 - (1) **select the 30 best features** of the previous generation (those with lowest weighted error on the training set);
 - (2) **apply to those 30 best features a Hill-Climbing consisting in applying to each of them a maximum of 5 successive specifically-designed "mutations"**, each mutation being cancelled if it did not improve the feature;
 - (3) **complete the population with 40 new simple random features;**



- **stop the algorithm when there has been no improvement during 40 consecutive generations** → best feature of the last generation is the one selected for this adaBoost round.



**Application pour faciliter
la mise au point par apprentissage
de classifieurs pour la détection visuelle d'objets
(développée au CAOR/ENSMP par Yotam Abramson)**

Techniques assemblées dans SEVILLE

- Possibilité de sélection de sous-images à inclure comme exemples positifs ou négatifs dans une base
- Apprentissage par boosting sur base d'imagettes :
 - les classifieurs élémentaires sont des « control-point » features (voir plus loin) ;
 - Le « weak-learner » (l'algo appliqué à chaque « round » de boosting pour choisir un nouveau feature) est une recherche de type algorithme évolutionniste.
- Possibilité de tester sur une vidéo (aussi bien image par image que « en continu ») un détecteur obtenu
- Facilités pour utiliser les « faux positifs » afin de compléter la base

- Pour effectuer un apprentissage, il faut constituer une base d'exemples
- C'est souvent une étape longue et fastidieuse !!
- Dans le cas du "boosting" on a plutôt intérêt à assurer un "écart de marge" maximum entre négatifs et positifs → "biaiser" la base pour inclure plus particulièrement des exemples "difficiles à classer"

Algorithme évolutionniste comme "weak learner"

Opérateurs d'évolution :
 Pour le moment, seulement des "mutations", définies spécifiquement pour ce problème (ajout, suppression, ou déplacement de 1 des points du feature, changement de résolution, changement de canal de couleur, ...)

```

C:\users\lyotam\UnitedWorkspace\LearnAdaBoost\Debug\LearnAdaBoost.exe
Starting. MAX_CONTROL_POINTS=6, dimensions (24,48)
Dividing the examples: 2/3 training, 1/3 validation
Divided 0 new files
Divided 0 new files
..... Loaded directory 146 positive
..... Loaded directory 2592 negative
Loaded directory 0 unlabelled
Read validation images (146 positive, 2592 negative)
..... Loaded directory 271 positive
..... Loaded directory 5212 negative
Loaded directory 0 unlabelled
Read training images (271 positive, 5212 negative)

**** Start training
Starting AdaBoost run.
AdaBoost step 0
Calculating weights...
Just to verify: sum weights = 1.000000
Weights calculated. Starting weak learner...
Minimum error = 0.446790, 0th time (119/271 pos, 3478/5212 neg)
Minimum error = 0.380784, 0th time (145/271 pos, 3666/5212 neg)
Minimum error = 0.366617, 0th time (178/271 pos, 3179/5212 neg)
Minimum error = 0.346621, 0th time (210/271 pos, 2772/5212 neg)
Minimum error = 0.346621, 1th time (210/271 pos, 2772/5212 neg)
Minimum error = 0.346621, 2th time (210/271 pos, 2772/5212 neg)
Minimum error = 0.337210, 0th time (133/271 pos, 4351/5212 neg)
Minimum error = 0.331573, 0th time (132/271 pos, 4429/5212 neg)
Minimum error = 0.325877, 0th time (168/271 pos, 3796/5212 neg)
Minimum error = 0.324483, 0th time (123/271 pos, 4676/5212 neg)
Minimum error = 0.324483, 1th time (123/271 pos, 4676/5212 neg)
Minimum error = 0.304230, 0th time (145/271 pos, 4464/5212 neg)
Minimum error = 0.304230, 1th time (145/271 pos, 4464/5212 neg)
Minimum error = 0.304230, 2th time (145/271 pos, 4464/5212 neg)
Minimum error = 0.304230, 3th time (145/271 pos, 4464/5212 neg)
Minimum error = 0.286427, 0th time (212/271 pos, 3361/5212 neg)
Minimum error = 0.286427, 1th time (212/271 pos, 3361/5212 neg)
Minimum error = 0.286427, 2th time (212/271 pos, 3361/5212 neg)
Minimum error = 0.286427, 3th time (212/271 pos, 3361/5212 neg)
Minimum error = 0.286427, 4th time (212/271 pos, 3361/5212 neg)
Minimum error = 0.27726, 0th time (172/271 pos, 4221/5212 neg)
Minimum error = 0.267887, 0th time (183/271 pos, 4112/5212 neg)
Minimum error = 0.267887, 1th time (183/271 pos, 4112/5212 neg)
Minimum error = 0.267887, 2th time (183/271 pos, 4112/5212 neg)
Minimum error = 0.267887, 3th time (183/271 pos, 4112/5212 neg)
    
```

- **Ouvrir une séquence vidéo.**
- **Collecter manuellement au moins 1 ou 2 douzaines d'exemples positifs VARIÉS ET "CENTRES", ainsi que 2 ou 3 douzaines d'exemples négatifs VARIÉS**
- **Lancer un premier apprentissage.**



Etiquetage semi-automatique pour enrichissement **BIAISÉ** de la base

- **Le programme affiche ce que le détecteur courant considère comme "positif" (rectangles en surimpression sur l'image).**
- **L'utilisateur transforme les fausses détections en exemples négatifs (et ajoute "manuellement" les "détections ratées").**



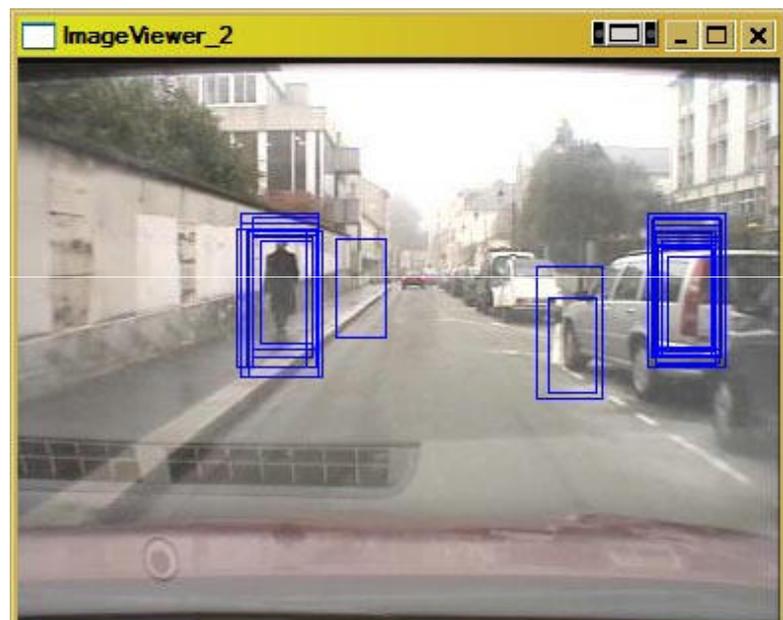
- Utiliser tous les exemples (base enrichie) pour lancer à nouveau l'apprentissage

➔ Les résultats du nouveau détecteur sont un peu meilleurs



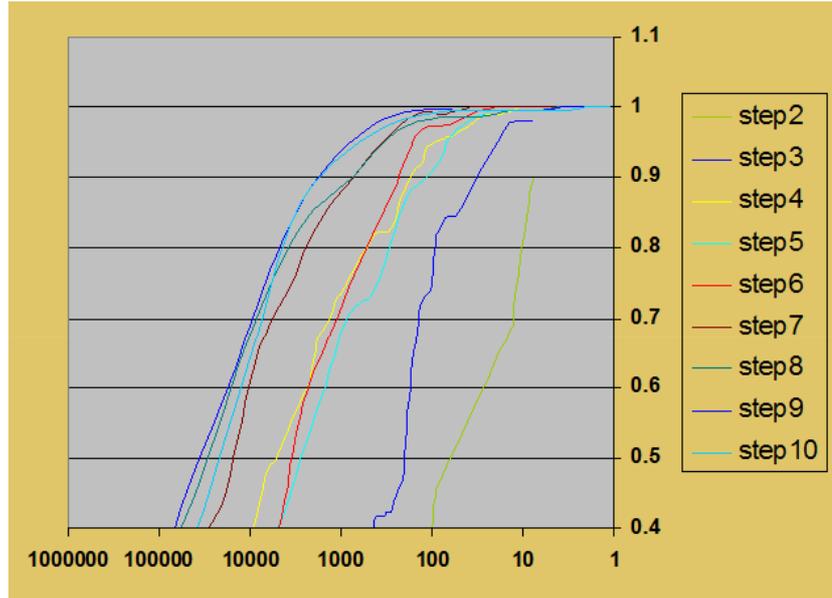
Cycles suivants

- Utiliser le nouveau détecteur sur d'autres plans fixes de la même séquence.
- Collecter de nouveaux exemples (principalement négatifs) toujours à l'aide des résultats du détecteur courant.
- Relancer l'apprentissage...



Amélioration au fil des cycles d'enrichissement de la base avec SEVILLE

- Au fur et à mesure des cycles répétés “apprentissage + enrichissement de la base”, les performances s'améliorent.



courbes de ROC successives sur un ensemble *indépendant* de validation

Back-viewed car detection in urban area



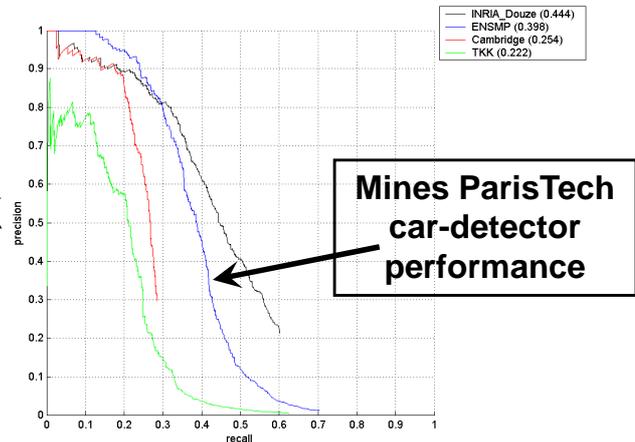
**~ 95% car
detection rate**
when
false detection rate
below 1/40,000
(i.e. < 1 false detection
per image frame)
process 3-10 fr/s
(320x240 videos)
on standard laptop

Comparison with state-of-the-art visual object detection

Participation to the Pascal VOC (Visual Object Class) challenge 2006

We focused essentially on car detection task (but *including lateral views* as well as front and rear views)

2nd best result on car detection task



Connected-Control-Points variant

Exactly same principle, but limit pixels configurations (and weak-learner search space !) by enforcing that all pixels be connected (at least by a corner)

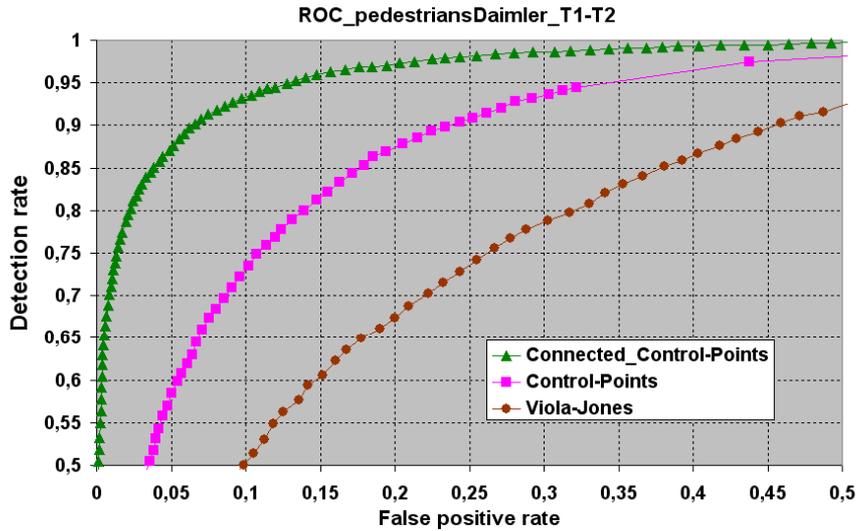


Typical connected-Control-Points selected during Adaboost training

For comparison, typical Adaboost-selected Haar features

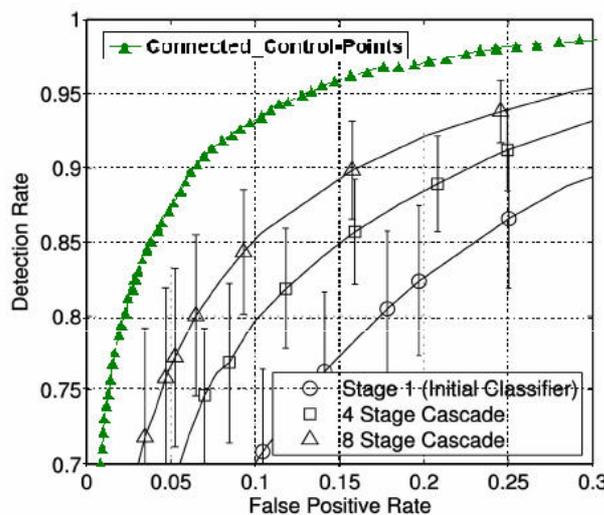


Results on Daimler pedestrians database



Boosted classifiers with connected-CP features *perform much better* than those with standard CP features, and even much better than those with Haar-like Viola-Jones features (all detectors with one single layer of 2000 weak-classifiers)

Comparison with published results on same pedestrians database



Our boosted classifiers with 2000 connected-CP features clearly outperform best results attainable with Haar-like Viola-Jones cascade reported in [MG2006], while executing at <math><0.4\text{ms/sample}</math> as well