Detecting Pedestrians Using Patterns of Motion and Appearance

Paul Viola
Microsoft Research
viola@microsoft.com

Michael J. Jones
Mitsubishi Electric Research Labs
mjones@merl.com

Daniel Snow
Mitsubishi Electric Research Labs
snow@merl.com

Alex J. Aved
2/14/2005

TOC

- Overview
- AdaBoost
- Introduction
- Detection of Motion
- Training Process
- Experiments & Results

Objective

- Machine Learning approach for pedestrian detection that maximizes detection accuracy and minimizes computation time.
- Detection at very low resolution taking advantage of motion and appearance information.

Overview

- Use machine learning to construct a detector from a large number of training examples.
- Work directly with images to detect instances of potential objects.
- Use AdaBoost to select a subset of features and construct a cascade of classifiers.

Overview – Approach

- Direct detection of pedestrians
  - Intensity features
  - Motion energy features
- Feature selection determines optimal balance
- Based on Viola/Jones detector (fast)
- Overall system is very simple
  - Large feature set
  - Training data
  - Detection requirements
  - No tracker
  - No alignment

Introduction – Boosting

AdaBoost – Adaptive boost: Freund & Schapire, AT&T Labs, 1996

- Consider example of a gambler allowing his agents to make bets on his behalf.
- Make a program that predicts accurately the winner of races.
- How to combine many rules-of-thumb into an accurate prediction rule.
- Boosting is to produce very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb.

Introduction – Boosting

- Booster is provided with a set of labeled training examples $(x_1, y_1), \ldots, (x_N, y_N)$
- Where *yi* is the label associated with instance *xi* ; *xi* is observable data & *yi* is the outcome.
- On each round **t = 1,2 , … ,T**, the booster devices a distribution Df over the set of examples, and requests a weak hypothesis (or rule-of-thumb) ht with low error with respect to Df.
- The distribution of Df specifies the relative importance of each example for the current round.
- After T rounds, the booster must combine the weak hypothesis into a single prediction rule.

Introduction – AdaBoost.M1

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

- Superior error bound
- Does not require prior knowledge about the accuracy of the hypothesis

Introduction – AdaBoost.M1

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

Introduction – AdaBoost.M1

- For $t = 1, \ldots, T$:
  1. Normalize the weights,

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

     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:

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

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

Introduction – AdaBoost.M1

- The final strong classifier is:

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

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

Note: log(..) refers to the natural log, "ln"

Theorem 10 *Suppose the weak learning algorithm* **WeakLearn**, *when called by* **AdaBoost.M1**, *generates hypotheses with errors* $\epsilon_1, \ldots, \epsilon_T$, *where* ϵ_t *is as defined in Figure 3. Assume each* $\epsilon_t \leq 1/2$. *Then the error* $\epsilon = \Pr_{i \sim D}[h_f(x_i) \neq y_i]$ *of the final hypothesis* h_f *output by* **AdaBoost.M1** *is bounded above by*

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t (1 - \epsilon_t)}.$$

AdaBoost.M1 Summary

- The algorithm maintains a set of weights w^t over training examples
- Weights used to generate hypothesis h(t). Using h(t), a new weight vector w^(t+1) is generated
- After T iterations, a final hypothesis is output, combining the outputs of the T weak hypothesis using a weighted majority vote

Introduction

- Algorithm inputs:
  - Feature set
  - Scale for training data
  - Scales for detection

Note: the exhaustive set of rectangle features is larger than 180,000

(Values for other various parameters determined dynamically by AdaBoost)

Introduction

- Integral Image

  The Image at location *x , y* contains sum of the pixels above and to the left of *x, y,* inclusive.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y')$$

Can be calculated in one pass over image:

$$\begin{aligned} s(x,y) &= s(x,y-1) + i(x,y) \\ ii(x,y) &= ii(x-1,y) + s(x,y) \end{aligned}$$

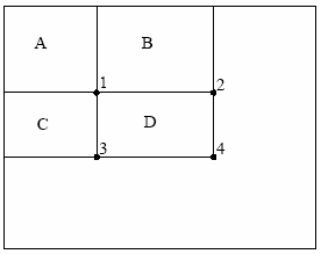
"Integral Image" similar to "summed area tables" in graphics.

F. Crow. Summed-area tables for texture mapping. SIGGRAPH 1984

Introduction

•A 2-rectangle feature can be computed in 6 array references;

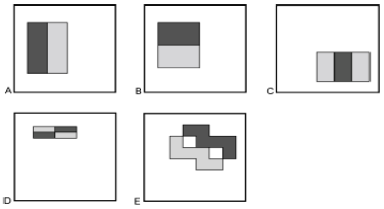
3 rectangle feature in 8 arr. ref's; 4-rec in 9 ref's.



2-rec feature: difference between regions

3-rec feature: center – sum of outer

4-rec feature: difference between diagonal pairs



Sum of pixels within rec. D (above) can be computed with 4 array references:

Val(2)=A+B, Val(3)=A+C,

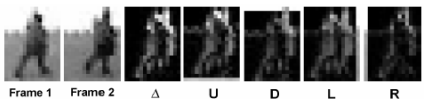
Val(4)=A+B+C+D, so

Sum(D)= 4 + 1 – (2 + 3)

Detection of Motion

- Based on simple rectangle filters.
- Features operate on the difference between pairs of image in time.
- Motion filters operate on 5 images:

$$\Delta = abs(I_t - I_{t+1})$$
$$U = abs(I_t - I_{t+1} \uparrow)$$
$$L = abs(I_t - I_{t+1} \leftarrow)$$
$$R = abs(I_t - I_{t+1} \rightarrow)$$
$$D = abs(I_t - I_{t+1} \downarrow)$$



Regions where the difference is large correspond to motion.

Rectangle filters computed on these images in addition to original image patch

Detection of Motion

- Filter for direction

$$f_i = r_i(\Delta) - r_i(S)$$

S is one of { *U* , *L* , *R* , *D* }

$r_i()$ is a rectangular sum

- Filter for motion shear

$$f_j = \phi_j(S)$$

Compares sum within the same motion image

- Measuring the magnitude of the motion

$$f_k = r_k(S)$$

Single box sum in detection window

- Appearance Filter

$$f_m = \phi(I_t)$$

Rectangular filter, operates on first input image

Integral image used for evaluating filters.

Detection of Motion

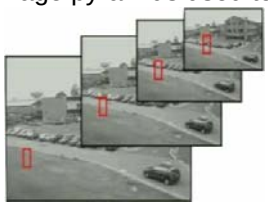
- Classifier

$$C(I_t, I_{t+1}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} F_i(I_t, \Delta, U, L, R, D) > \theta \\ 0 & \text{otherwise} \end{cases}$$

- Feature

$$F_i(I_t, I_{t+1}) = \begin{cases} \alpha & \text{if } f_i(I_t, \Delta, U, L, R, D) > t_i \\ \beta & \text{otherwise} \end{cases}$$

- Image pyramids used to achieve: motion velocity scale invariant

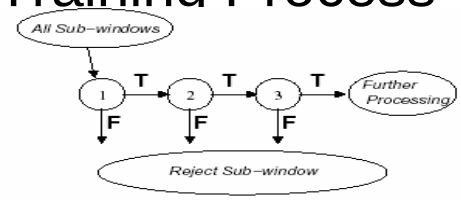


$$\Delta^t = abs(I_t^t - I_{t+1}^t)$$
$$U^t = abs(I_t^t - I_{t+1}^t \uparrow)$$
$$L^t = abs(I_t^t - I_{t+1}^t \leftarrow)$$
$$R^t = abs(I_t^t - I_{t+1}^t \rightarrow)$$
$$D^t = abs(I_t^t - I_{t+1}^t \downarrow)$$

Training Process

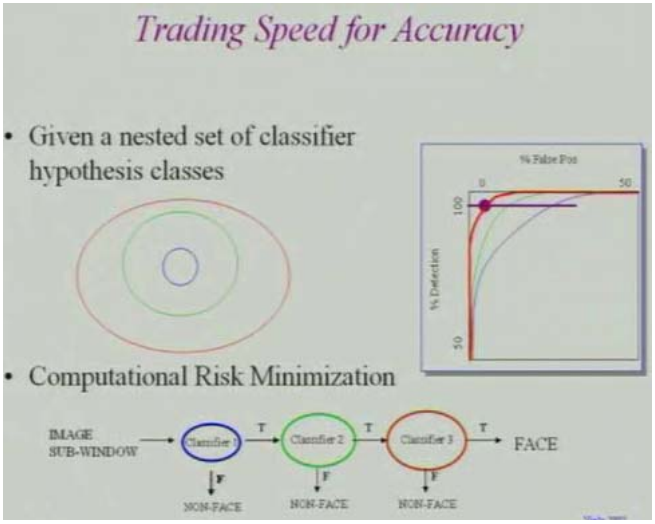
- Given a feature set and training set of positive and negative image, AdaBoost is used to select a subset of features
- AdaBoost picks the optimal threshold for each feature as well as α and β of each feature ("train the classifier")
- Output is a classifier consisting of linear combination of the selected feature; balances intensity & motion information to maximize detection rate

Training Process



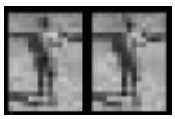
- Classifiers arranged in cascades.
- Each classifier trained by AdaBoost.
- Simple detectors with small number of features placed earlier in the cascade.
- Each stage decreases the false positives.
- Each stage trained by adding features until target detection and false positive rates are met.
- Think of it as a "focus of attention" mechanism

Training Process



Experiments & Results

- Static detector
  - Trained with static image patches
  - Does not use motion information
- Dynamic detector
  - Trained on consecutive frame pairs
  - Motion information



$$f_i = r_i(\Delta) - r_i(S)$$

Experiments & Results

- Dynamic detector trained on consecutive frames using positive and negative examples.

Experiments & Results

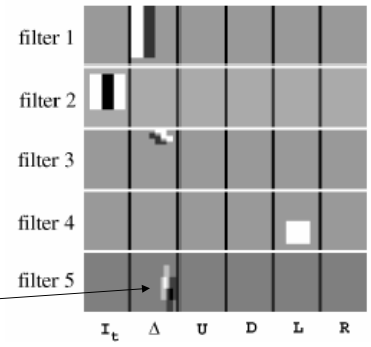
•Each classifier in the cascade trained using the positive and false positives.

•The detection threshold of the classifier is adjusted so that the false negative rate is very low.

•Static pedestrian detector trained in the same way.

Experiments & Results

- Filters learned for the dynamic detector

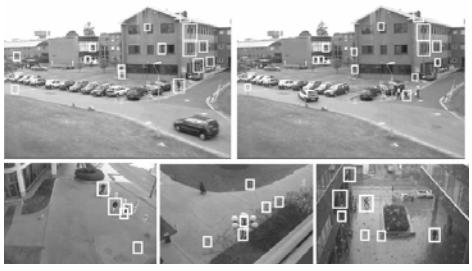


Alex's guess: this detects legs moving between frames

- Filters learned for static detector.

Experiments & Results

Dynamic detector



Static detector

Experiments & Results

Dynamic

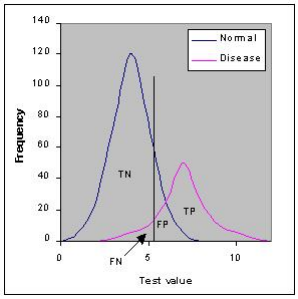
Static

Conclusion

- Integrates intensity information with motion information.
- Work well under low resolution images under difficult conditions.
- Does not detect occluded or partial human figures.

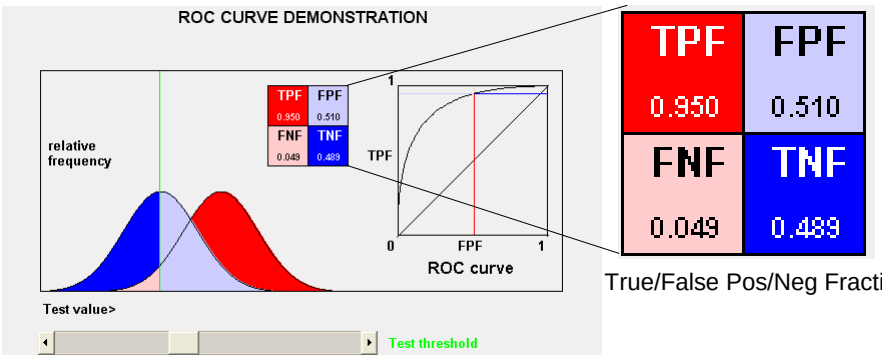
- This presentation was developed based upon the PowerPoint presentation by Imran Nazir
- http://www.research.microsoft.com/iccv_2003/

Receiver Operating Characteristic



*It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
*The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
*The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.
*The slope of the tangent line at a cutpoint gives the likelihood ratio (LR) for that value of the test.
*You can check this out on the graph above. Recall that the LR for T4 < 5 is 52. This corresponds to the far left, steep portion of the curve. The LR for T4 > 9 is 0.2. This corresponds to the far right, nearly horizontal portion of the curve.
*The area under the curve is a measure of test accuracy.

Receiver Operating Characteristic



True/False Pos/Neg Fraction

http://www.anaesthetist.com/mnm/stats/roc/

Conclusion

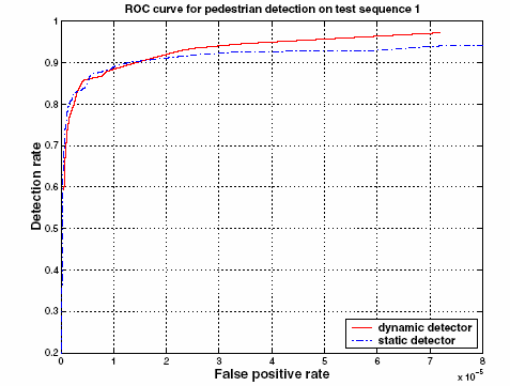


Figure 8: ROC curve on test sequence 1 for both the dynamic and static pedestrian detectors. Both detectors achieve a detection rate of about 80% with a false positive rate of about 1/400,000 (which corresponds to about 1 false positive every 2 frames for the 360x240 pixel frames of this sequence).

Conclusion

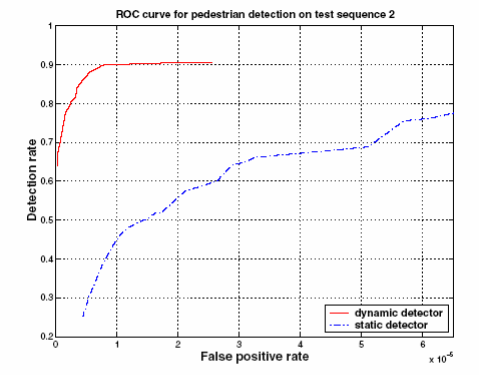


Figure 9: ROC curve on test sequence 2 for both the dynamic and static pedestrian detectors. The dynamic detector has much greater accuracy in this case. At a detection rate of 80%, the dynamic detector has a false positive rate of about 1/400,000 while the static detector has a false positive rate of about 1/15,000.