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

Imran Nazir

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.

Introduction – Boosting

- 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 $(x_1, y_1), \ldots, (x_N, y_N)$
- 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 – AdaBoosting

- 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, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  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 $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- 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

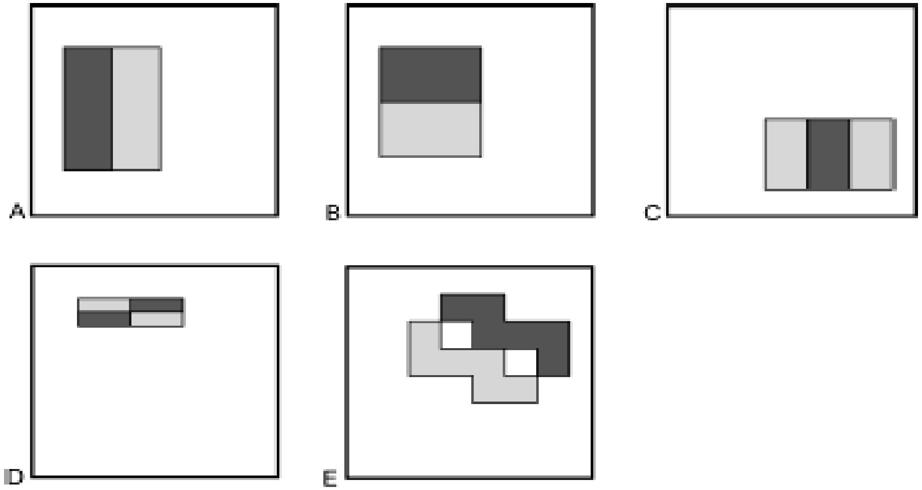
Introduction

- Integral Image

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

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

- Rectangular sum can be computed in four array references.



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:

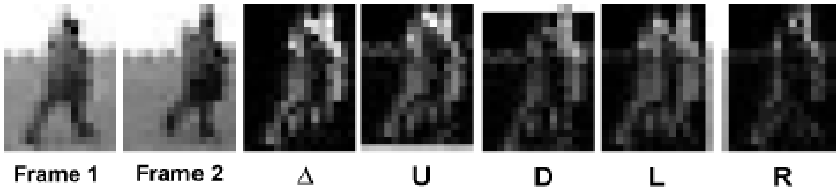
$$\Delta = abs(I_t - I_{t+1})$$

$$U = abs(I_t - I_{t+1} \uparrow)$$

$$L = abs(I_t - I_{t+1} \leftarrow)$$

$$R = abs(I_t - I_{t+1} \rightarrow)$$

$$D = abs(I_t - I_{t+1} \downarrow)$$



Detection of Motion

- Filter for direction

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

S = { *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

$$f_k = r_k(S)$$

- Appearance Filter

$$f_m = \phi(I_t)$$

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 are used to make the motion velocity scale invariant

$$\Delta^l = abs(I_t^l - I_{t+1}^l)$$

$$U^l = abs(I_t^l - I_{t+1}^l \uparrow)$$

$$L^l = abs(I_t^l - I_{t+1}^l \leftarrow)$$

$$R^l = abs(I_t^l - I_{t+1}^l \rightarrow)$$

$$D^l = abs(I_t^l - I_{t+1}^l \downarrow)$$

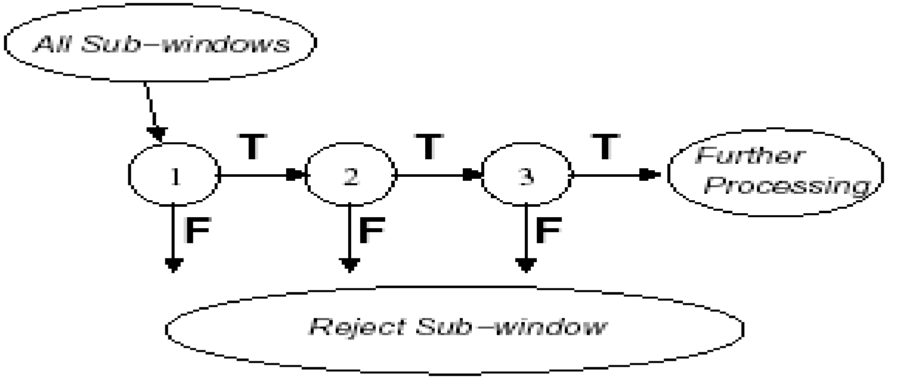
Training Process

- 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 features as well as α and β of each feature.

- Output is a classifier consisting of linear combination of the selected feature.

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.

Experiments & Results

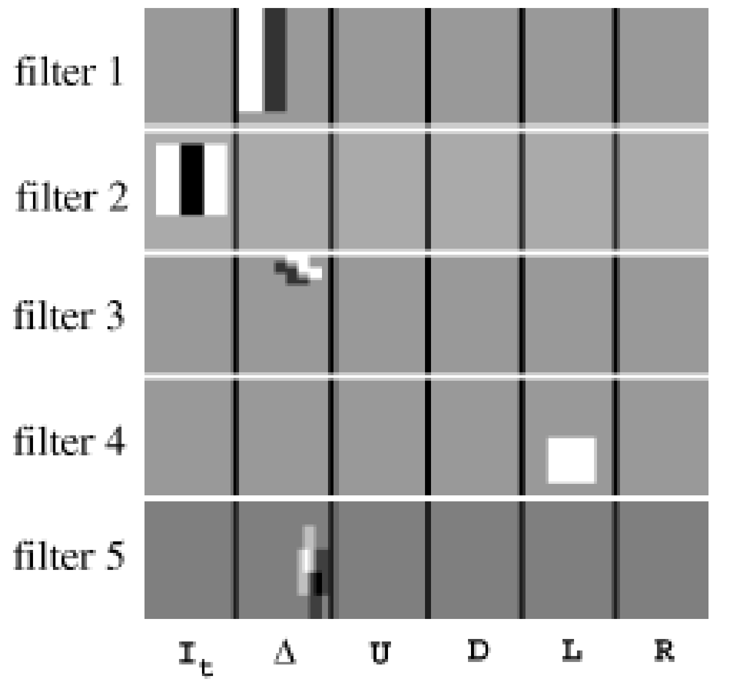
- Dynamic detector trained on consecutive frames using positive and negative examples.

Experiments & Results

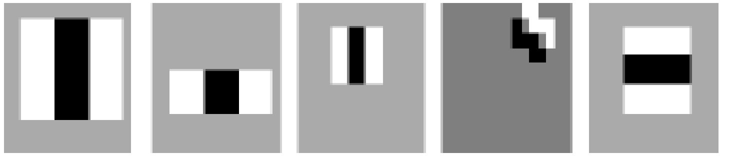
- Each classifier in the cascade trained using the positive and false positives.
- The detection threshold of the classifier is adjusted so that the false negative rate is very low.
- Static pedestrian detector trained in the same way.

Experiments & Results

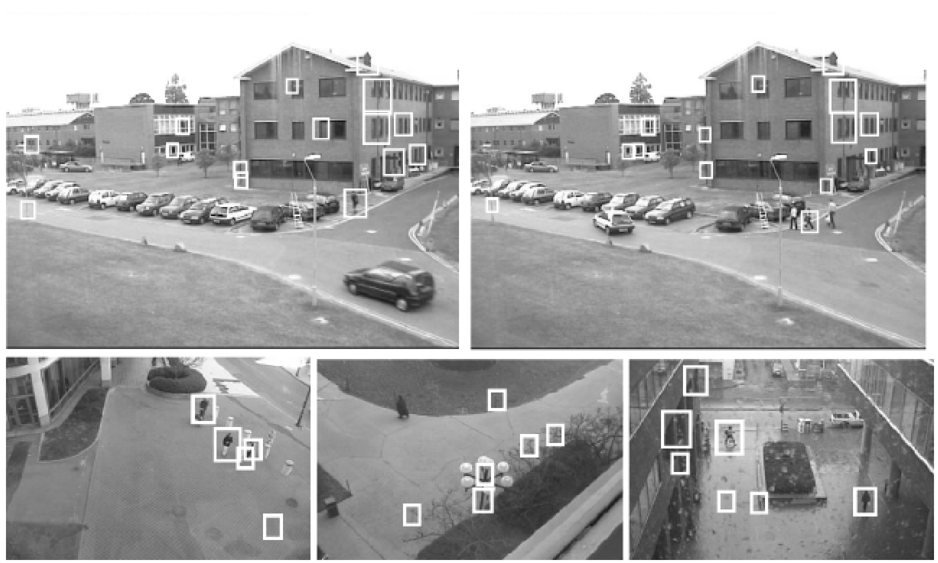
- Filters learned for the dynamic detector



- Filters learned for static detector.



Experiments & Results



Conclusion

- Integrates intensity information with motion information.

- Work well under low resolution images under difficult conditions.

- Does not detect occluded or partial human figures.