Human Detection in Images

C. V. Jawahar

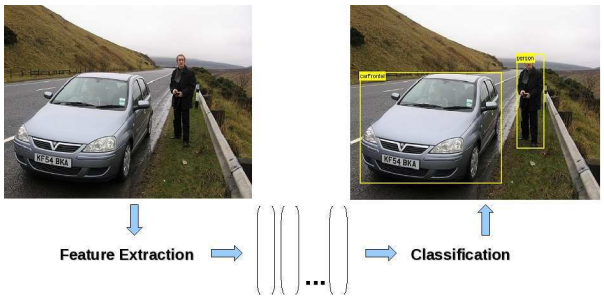
Center for Visual Information technology
International Institute of Information Technology
Hyderabad, India
http://www.iiit.ac.in/~jawahar

2014

Introduction to Object Detection

Given an image, task of object detection is to test whether the object of interest is present in the image, and if present, localize it with a simple bounding box like structure.

- Popular Approach: Classify (all) the windows and label.



Introduction to Object Detection

- Approaches for Classification
  - Object Detection Vs Scene Classification
  - Part-Based or Segmentation Based
  - Sliding Window: A detector window is scanned across the image at multiple scales and locations. Features extracted from each of these windows are assigned confidence scores using a classifier.
- Two basic ingredients: Feature Extraction and Classification
- Features: Haar Features, Histograms etc.
- Classifiers:
  - Neural Networks
  - Adaboost
  - SVMs
  - Random Forests

Human Detection in Images and Videos

Challenges

- Variable appearance/clothing.
- Wide variety of articulated poses.
- Presence of Complex and cluttered background.
- Self, mutual and other occlusions.
- Unconstrained illumination.
- Different scales.
- Occurrence of multiple objects at varying scale.

Human Detection in Images and Videos

Challenges

Human Detection in Images and Videos

Applications

- One of the most frequent object in videos which we capture.
- Visual surveillance, Security systems, Tracking.
- Pedestrian detection for smart cars/automobiles.
- Key for machines to interact intelligently and effortlessly with a human-inhabited environment.
- Virtual reality, film and media analysis.
- Activity based summarization and analysis of videos.
- Advanced user interfaces
- Motion analysis

Human Detection: State of The Art

- Papageorgiou et al in IJCV'00, describe a pedestrian detector based on a polynomial SVM using rectified Haar wavelets as input descriptors, with a parts (subwindow) based variant in PAMI'01.
- Ronfard et al (ECCV'02) build an articulated body detector by incorporating SVM based limb classifiers over 1st and 2nd order Gaussian filters in a dynamic programming framework similar to those of Felzenszwalb and Huttenlocher (CVPR'00) and Ioffe and Forsyth (IJCV'01).
- Viola et al (ICCV'03) build an efficient moving person detector, using AdaBoost to train a chain of progressively more complex region rejection rules based on Haar-like wavelets and space-time differences.
- Parts based binary orientation position histograms + adaBoost by Mikolajczyk et al (2004).

Human Detection: Dalal and Triggs

- A significant work in this direction is by Dalal and Triggs[1]
- Represent Human images with robust feature descriptors called HoG.
  - HoG is a "SIFT-like" feature, but computed densely.
- A linear Support Vector Machine is trained using these descriptors.
  - Linear is used to make the computations efficient.
- Sliding window based SVM detector is used; followed by a step on non-maximum suppression.

[1] Navneet Dalal and Bill Triggs, "Histogram of Oriented Gradients for Human Detection", CVPR 2005

Human Detection: Some Results

Human Detection: Some Results

Histogram of Oriented Gradients

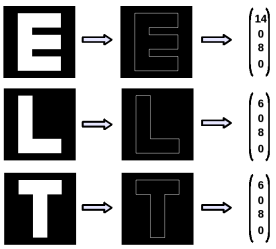
- Feature sets are based on dense and overlapping encoding of image regions using Histogram of Oriented Gradient (HOG) descriptors.
- Histogram of Oriented Gradient descriptors provide a dense indeed overlapping description of image regions.
- Extensively used set of features for the task of object detection.
- Recently HOG and it's variants have given state of art performance on many object classes.

HoG: An example of naive HOG

- Consider the problem of discriminating character images (say 'E' 'L' and 'T').
- We first compute the gradient image (as in edge detection) and build a histogram of gradient orientations.
- We quantize the gradients into (say) four orientation bins: $[0, \pi/4)$, $[\pi/4, \pi/2)$, $[\pi/2, 3\pi/4)$ and $[3\pi/4, \pi)$.
- And histogram is formed by voting to these bins. Let the vote be proportional to the gradient magnitude.
- We consider this histogram as the feature vector.

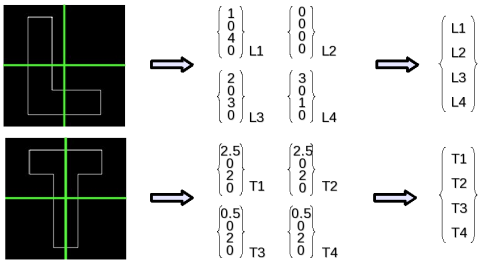
HoG: An example of naive HOG

- Input images, gradient images and feature vector are shown in the figure.
- This representation is able to discriminate E from L and T, but L and T have same feature vector.
- To be more discriminative, we need to add local information i.e. gradient orientations at different locations.



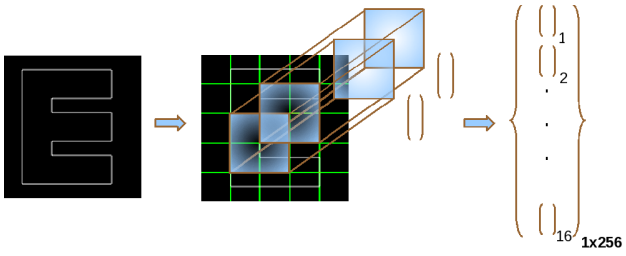
Utility of Local information

- Need to add local information i.e. gradient orientations at different locations.
- Divide image into 2X2 grid and combine 4 histograms to get a 16 dimensional feature vector.
- By adding local information now L and T can be discriminated.



Utility of Local information

- More discrimination might be required for more complex tasks.
- This can be done by adding more local information by finer subdivision of image.

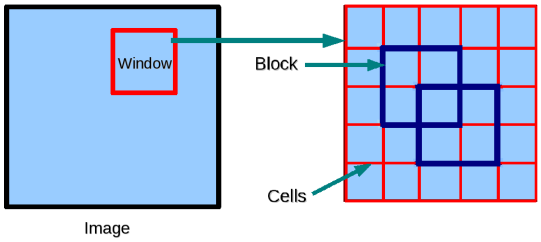


Role of Parameters

- If we use finer and finer sub-divsion of images, we are constraining the object appearence/variability spatially. An example of *Tradeoff* between invariance and discriminability.
- If we quantize the gradients into finer resolution, we are constraining the object appearence. Yet another example of *Tradeoff* between invariance (applicability under allowable transformations) and discriminability.
- It is important to "learn" these parameters from examples for a given problem.

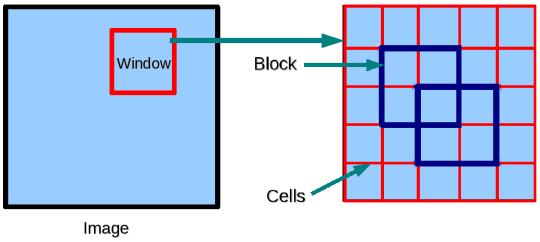
Definitions/Terminology

- *Windows* are the sub-images for which final HoG descriptor is computed (and classification is carried out, for object detection).
- Each window is divided into number of *Cells* of size $p \times p$ pixels.
- A *Block* is a grid of cells. Blocks could overlap.



Definitions/Terminology: The base resolution

- Navneet et al use 64×128 as *base window size*.
- Size of *cell* used is 8×8 pixels.
- *Block* is a 2×2 grid of cells.
- *Window* is a grid of overlapping *Blocks*.

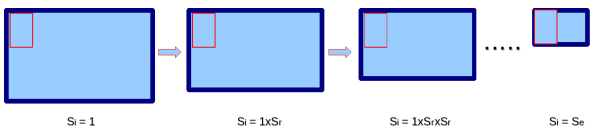


HoG Descriptor for a Window

- The number of orientation bins used in the paper (for human detection) is 9.
- And there is 50% overlap between two blocks, then what is the dimension of final HoG descriptor?
- There are $\frac{64}{8} = 8$ cells in each row and $\frac{128}{8} = 16$ cells in each column
- With 50% overlap we have 7 blocks per row and 15 blocks per column i.e. **105** blocks.
- Dimension of block descriptor = number of cells $\times$ Number of orientation bins $= 2 \times 2 \times 9 = \mathbf{36}$
- Dimension of final HOG descriptor $= 36 \times 105 = \mathbf{3780}$

The Structure of the Solution

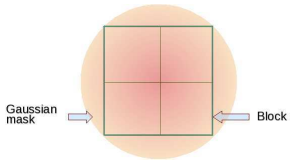
- Initialize
  - Start with $S_s = 1$; compute end scale, such that $min(\frac{W_i}{Wn}, \frac{H_i}{H_n})$ is 1,
    where W_i and H_i are image width and height, respectively and W_n and H_n are base width and base height of window.
- For each scale $S_i = [S_s, S_s S_r, .., S_n]$
  - rescale the input image using bilinear interpolation.
  - extract features for all the windows (by sliding in space).
  - assign detection score to them.



Typically $S_r = 1.05$, $W_n = 64$, $H_n = 128$

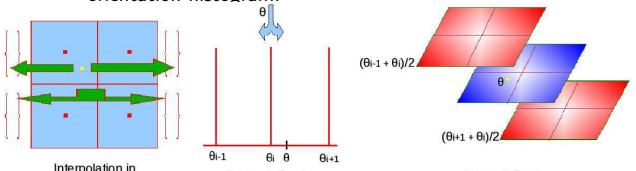
Computation of HOG descriptor: Weighted Voting

- The vote from a pixel is a function of it's gradient magnitude.
- To give more importance to the pixels which are *more* inside the block, pixels near the edges of the block are down-weighted.
- A Gaussian spatial window is applied to each pixel before accumulating orientation votes into cells.
- σ of Gaussian mask can be taken around $0.5 \times block\_width$.



Computation of HOG descriptor: Soft Voting and Interpolation

- *Soft assignment* of pixel votes is done using *Interpolation* in space and orientation:
  - The orientation bins are evenly spaced over the orientation range (π or 2π).
  - According to the position of pixel it's vote is distributed between 4 neighboring spatial bins (cells).
  - According to the orientation at the pixel, it's vote is distributed between two neighboring bins.
  - So, weight is distributed between 8 surrounding bins in spatial orientation histogram.



Computation of HOG descriptor: Normalization of Histograms

- Blocks are overlapped so that each cell response contribute to several components in the final descriptor, each normalized with respect to different block.
- This strong normalization is essential and improves performance.
- Normalization schemes for blocks:
  - L2-norm: $v \rightarrow \frac{v}{\sqrt{||v||_2^2+\epsilon^2}}$
  - L2-Hys: L2-norm followed by clipping and re normalizing.
  - L1-norm: $v \rightarrow \frac{v}{||v||_1+\epsilon}$
  - L1-sqrt: $v \rightarrow \sqrt{\frac{v}{||v||_1+\epsilon}}$

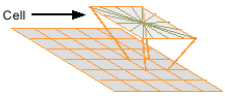
  where v is block descriptor.

Computation of HOG descriptor

- Input image is first preprocessed to normalize gamma and color.
- Gradient orientation at pixel (i,j):

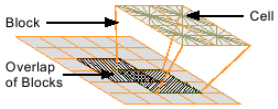
$$\theta(i,j) = \arctan(\frac{\delta y}{\delta x})_{(i,j)}$$

  Where δy and δx are the gradient values in vertical and horizontal gradient images respectively.
- Each pixel gives a *weighted* vote into a orientation bin based on the gradient orientation at that pixel.
- These votes are accumulated over *cells*.



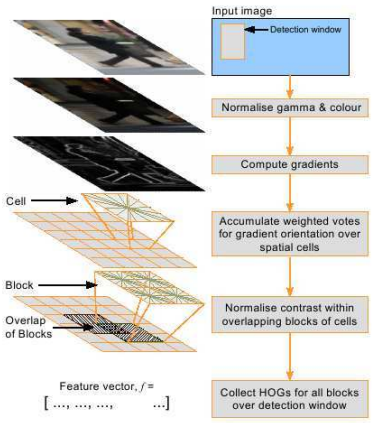
Computation of HOG descriptor

- Votes are interpolated between neighboring bin centers both in orientation and position to reduce aliasing.
- For effective local contrast normalization cells are grouped into larger spatial blocks and each block is normalized separately.



- The final HOG descriptor collects the descriptors from all the blocks of a dense overlapping grid of blocks.

Computation of HOG descriptor: Summary

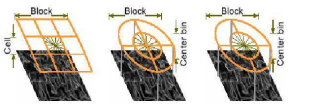


Two types of HOG descriptors

R-HOG Image window is represented as a dense (usually overlapping) grid of rectangular blocks and each such block is a grid of rectangular cells.

C-HOG In C-HOG descriptors, the cells are defined into grids of log-polar shape. The image window is covered by a dense rectangular grid of centers.

- At each center, we divide the local image patch into a number of angular and radial bins.
- Single center C-HOG has single circular central cell.



HoG

- Finally, HoG representation is calculated for a window.
- Representation is 3780 dimensional.

HoG

- Finally, HoG representation is calculated for a window.
- Representation is 3780 dimensional.
- Why naive HoG is not good enough?
- How do we find the right HoG/feature descriptor?

Review of SVM

Let N m-dimensional training inputs $\mathbf{x_i}(i = 1 \ldots M)$ belong to Class 1 or 2 and the associated labels be $y_i = 1$ for Class 1 and -1 for Class 2. Decision function which needs to be determined is

$$f(\mathbf{x_i}) = \mathbf{x_i} \cdot \mathbf{w} + b$$

Therefore,

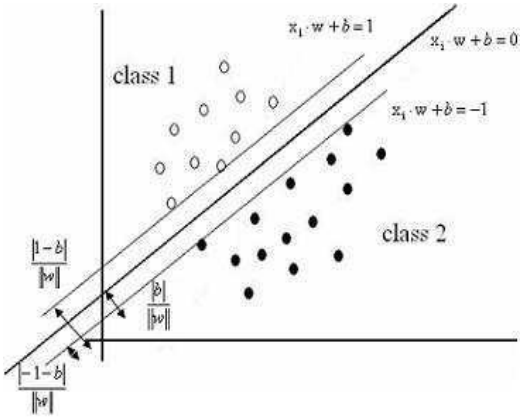
$$\mathbf{x_i} \cdot \mathbf{w} + b > 0 \text{ for } y_i = +1$$
$$\mathbf{x_i} \cdot \mathbf{w} + b < 0 \text{ for } y_i = -1$$

As training data are linearly separable, no training data satisfy $\mathbf{x_i} \cdot \mathbf{w} + b = 0$, to control separability we can consider following inequalities:

$$\mathbf{x_i} \cdot \mathbf{w} + b \geq +1 \text{ for } y_i = +1$$
$$\mathbf{x_i} \cdot \mathbf{w} + b \leq -1 \text{ for } y_i = -1$$

Combining these into one inequality:

$$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i$$



Support Vectors

The vectors which satisfy the equality in the previous equation are called *support vectors*.
They are the points which lie closest to the decision surface and are therefore the most difficult to classify. They have a direct bearing on the optimum location of the decision surface. (*In fact nobody else has ...!!*)

Train and Test

Training: During the training, one computes the SVM from the available data set. One computes the α_i as well as the support vectors.

Train and Test

Training: During the training, one computes the SVM from the available data set. One computes the α_i as well as the support vectors.

Testing: On testing we simply determine on which side of the decision boundary a given test pattern $\mathbf{x}$ lies and assign the corresponding class label. i.e, we take the class of $\mathbf{x}$ to be $sgn(\mathbf{w} \cdot \mathbf{x} + b)$

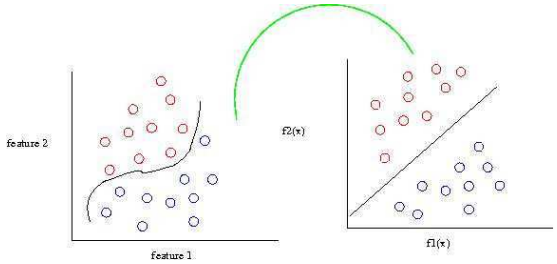
$$sign(\sum_i \alpha_i y_i x_i^T x + b)$$

Kernels

Interestingly, the vectors appear as only dot product in the formulation. This allow us to solve the problem in a very high dimension (where the data set will well behave) without explicitly bothering about the mapping which converts into higher dimension. We need only a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$

$$K(\mathbf{s}_i, \mathbf{x}_i) = \Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{x}_i)$$

Classification done in feature space



Equivalence with other classifiers

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

is equivalent to a polynomial classifier of order p

$$K(\mathbf{x}, \mathbf{y}) = \exp^{\frac{-||\mathbf{x}-\mathbf{y}||}{2\sigma^2}}$$

is equivalent to a Gaussian radial basis function classifier.

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta)$$

is equivalent to a particular case of two-layered sigmoidal neural network.

SVM Classification

Problem: Given a set of training examples (vectors $\mathbf{x}_i$ and their corresponding labels $y_i \in \{-1, +1\}$), find an optimal decision boundary.

Solution SVM Training results in a classification solution such that:

- Classify the sameple $\mathbf{z}$ as Class 1 (or label is $+1$) if

$$\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{z} + b$$

  is positive.
- Else assign it to the negative (other) class.
- α_is and b are computed during training.

Support Vectors and Classification

- $\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{z} + b$ is evaluated for each test sample.
- α_i is typically zero for most of the samples. It is non-zero for samples which are "difficult" to classify. Therefore the summation needs to be done only for a smaller set of samples.
- The samples where α_i is non-zero are called "Support Vectors".
- SVM training can also give you a solution in a different feature space. i.e., $\sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{z}) + b$
- The above can be evaluated with the help of a kernel function $\kappa()$ as $\sum_i \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{z}) + b$
- In fact, SVM allows training as well as testing using kernel functions without exactly knowing $\phi()$.

Linear SVM Classification

- Input vectors (samples) appear only as dotproducts in training and testing of SVMs. This makes SVM directly applicable along with kernels.
- When there is no additional kernel, the classification solution is linear and we call it as a Linear Kernel SVM or Linear SVM.
- Dalal and Triggs use Linear SVM for the classification.
- $\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{z} + b$ can then be rewritten as

$$\mathbf{w}^T \mathbf{z} + b$$

- In the case of Dalal and Triggs, it requires 3780 multiplication, one addition and one comparison per window.

Comments on Multiple Kernel Learning

- Kernels are popular in providing nonlinear decision boundary.
- However, it is not very obvious, what Kernel works well for a given problem.
- One can define a new kernel as:

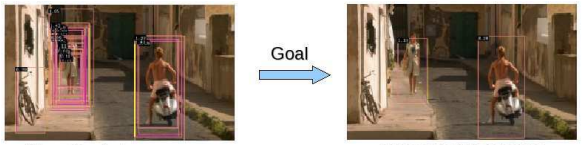
$$\kappa(,) = \sum_i d_i \kappa_i(,)$$

- Then the question boils down to learning the "optimal" d_i for a given task, given a set of individual kernels.
- MKL could be done by learning a specific kernel or directly computing a Kernel matrix.

$$\mathbf{K} = \sum_i d_i \mathbf{K}_i$$

Non Maximum Suppression

- After running classifier over test image there are multiple detections of same object and odd false positives.
- To merge such multiple detections and suppress odd false positives non-maximum suppression is done.
- The more overlapping detections there are in the neighborhood of an image region, the higher the probability for it to be a true positive.
- The goal is to fuse the nearby overlapping detections, but overlaps occurring at very different scales should not be fused



Non Maximum Suppression

- There are three steps in it:
  - Negative values are zeroed via clipping function.
  - Each detection is mapped to 3-D position in scale-space.
  - The final step applies mean shift mode detection algorithm to each detection. This provides local smoothing of detections.

Detection by Dalal and Triggs: Dataset

- 1239 image window containing human together with their left-right reflections (2478 in total) are used as training examples.
- Shown below are some examples of positive training image windows.

Inria person dataset- http://pascal.inrialpes.fr/data/human/

Detection by Dalal and Triggs: Dataset

- A fixed set of 12180 patches sampled randomly from 1218 person-free training images provided the negative set of examples.
- Shown below are some examples of person-free or negative training images.

Detection by Dalal and Triggs: Training

- HOG descriptors extracted from all the training images and a linear Support Vector Machine is trained using these descriptors.
- Then the trained classifier is ran over person-free images.
- The false positives obtained are added to the negative set and a linear SVM is trained on positive set + (initial negative set + false positive).
- Thus obtained is the final classifier for detection.

Detection by Dalal and Triggs: Testing

- Detector window is moved over imge at different scales.
  - Let H_i, W_i be image height and width at scale i.
  - δx, δy be horizontal and vertical shifts; H_n, W_n be window height and width.
  - Then number of windows evaluated at scale i is given by,

$$nr\_windows_i = (\lfloor \frac{H_i - H_n}{\delta y} \rfloor + 1) \times (\lfloor \frac{W_i - W_n}{\delta x} \rfloor + 1)$$

- For an image of size 480×640 and scale factor, $S_r = 1.05$, $\delta x = 8$, $\delta y = 8$, $H_n = 128$, $W_n = 64$
  - Number of scale levels $= 35$
  - In total for 34078 windows HoG descroptor is to be computed and are to be evaluated by linear SVM.
- It takes 9 seconds to compute HoG and classify them on single core AMD 64 bit Processor, 1.8GHz, 2GM RAM.

Detection by Dalal and Triggs: Code Related Details

- Navneet's source code is publicly avilable.
- In folder *learcode/lib* of source code:
  - File *windetect.cpp* does the HoG computation over windows, classifies using linear SVM and applies Non-maximum suppression.
  - HoG computation is implemented in *rhogdense.cpp*.
  - On compiling binaries are generated in *learcode/app*.
- Precompiled binaries are also provided.
- With help of scripts *runonimage.sh*, *runall.sh*, it is possible to both train and test classifiers.

HOG source code and binaries at- http://pascal.inrialpes.fr/soft/olt/

Human Detection: Dalal and Triggs

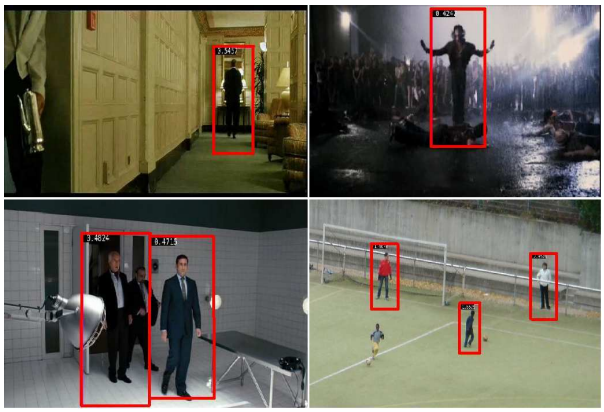
More Qualitative Results:

Human Detection: Dalal and Triggs

More Qualitative Results:

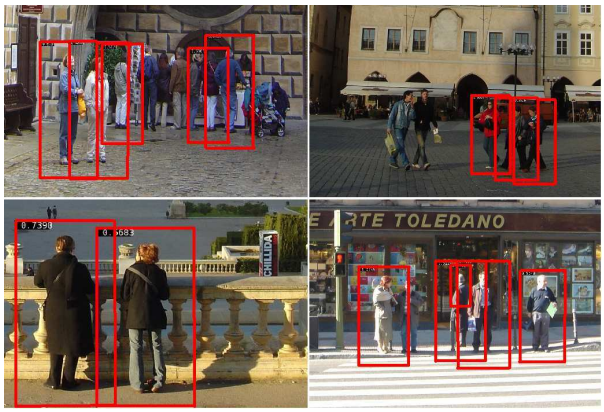
Human Detection: Dalal and Triggs

More Qualitative Results:



Human Detection: Dalal and Triggs

More Qualitative Results:



Visualization of R-HOG and what SVM learns

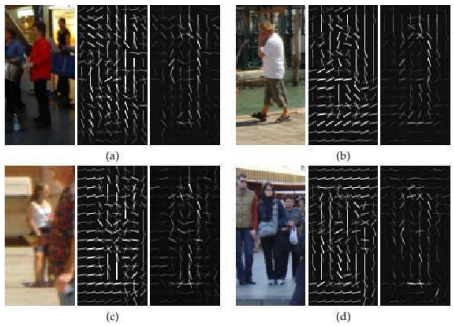


Figure: Each triplet from left to right display: (1) the input image, (2) the corresponding R-HOG feature vector (only the dominant orientation of each cell is shown), (3) the dominant orientations selected by the SVM (obtained by multiplying the feature vector by the corresponding weights from the linear SVM.)

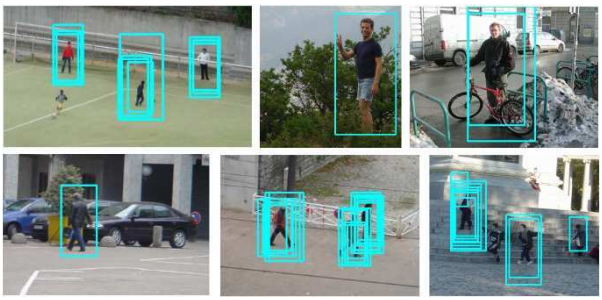
Discussion

- Better performance achieved for Human Detection with:
  - No gradient smoothing, $[-1, 0, 1]$ derivative mask.
  - Number of orientation bins $(\beta) = 9$ with *range* $= \pi$, $\beta = 18$ with *range* $= 2\pi$
  - Any normalization except L1-norm
  - More overlapping between the blocks.
  - 8×8 or 6×6 pixels per cell with 2×2 and 3×3 cells per block respectively.
- Increasing overlapping between the blocks and number of bins (β) always boosts performance but descriptor size increases.
- These optimal parameter values may change for some other objects class.
- Using blocks and cells of variable size improves result but computational cost greatly increases.

Extensions of HoG: Integral Histogram of Oriented Gradients

- In CVPR'06, S. Avidan et. al. proposed a variant of HOG, *Integral HOG*, which can be integrated into the *cascade of rejectors* framework.

Qualitative Results:



Extensions of HoG: HoG Feature Pyramid

- In CVPR'08, Felzenszwalb et al proposed HoG Feature Pyramid by computing HOG features of each level of a standard image pyramid.

Qualitative Results:

A Discriminatively Trained, Multiscale, Deformable Part Model, CVPR'08

Integral Histogram of Oriented Gradients

- In CVPR'06, S. Avidan et. al. proposed a variant of HOG, *Integral HOG*, which can be integrated into the *cascade of rejectors* framework.
- The major differences are:
  - Use of Integral image.
  - Variable size blocks.
- These variations give great speed up over original HOG.
- Combined with feature selection using *Adaboost* and Viola, Jones like Cascade Training, it gives a very fast detector while maintaining the accuracy level similar to HOG + Linear SVM.

Computation of HOG using Integral images

- Orientation of all the pixels is discretized into 9 (β) bins.
- An integral image for each bin of the HOG is computed and used to compute HOG of any rectangular region efficiently.
- This requires only 4×9 image access operations.
- Integral images gives speed but also restricts:
  - Use of Gaussian mask for spatial weighing.
  - Trilinear interpolation.
  - Use of L2-norm as L1-norm is faster to compute using Integral image.
- Block descriptor of Integral HOG is inferior to that of HOG.
- This is balanced by use of blocks of variable sizes in the detection window.

Variable-size Blocks

- Advantages of using a set of variable-size blocks are twofold:
  - For a specific object category, the useful patterns tend to spread over different scales. The original fixed number of fixed size blocks (105 16×16 blocks) only encode very limited information.
  - Some of the blocks in this large set of blocks might correspond to a semantic part in object, say human leg. A small number of fixed-size blocks is less likely to establish such mappings.
- The difficulty of selecting blocks from a large set is dealt by using *Adaboost*.

Cascade Training

- A rejection cascade similar to one proposed by Viola, Jones is used with following differences:

Feature each feature is a 36D block descriptor ($\eta = 2,\ \beta = 9$) Instead of one component of descriptor.

Weak Classifier separating hyperplane obtained by training linear SVM instead of threshold comparison.

- For each level of the cascade a strong classifier consisting of several weak classifiers is constructed.
- In each level of the cascade weak classifiers are added until:
  - *false positive rate* $\leq f_{max}$ (maximum acceptable false positive rate), and
  - *detection rate* $\geq d_{min}$ (minimum acceptable detection rate)
- This integration of the cascade-of-rejectors with HoG features matches the detection performance of previous methods with a up to 70X speedup.

References

- Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection", CVPR 2005.
- Navneet Dalal, "Finding people in images and videos", PhD Thesis.
- C. Papageorgiou and T. Poggio, "A trainable system for object detection. IJCV", 2000.
- A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components", PAMI, 2001.
- P. Felzenszwalb and D. Huttenlocher, "Efficient matching of pictorial structures", CVPR 2000.
- R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse pictures of people", ECCV 2002.

References

- P. Viola, M. J. Jones, and D. Snow. "Detecting pedestrians using patterns of motion and appearance", ICCV 2003.
- K. Mikolajczyk, C. Schmid, and A. Zisserman, "Human detection based on a probabilistic assembly of robust part detectors", ECCV 2004.
- Qiang Zhu, Shai Avidan, Mei-Chen Yeh and Kwang-Ting Cheng, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients", CVPR'06.
- P. Felzenszwalb, D. McAllester and D. Ramanan, "A Discriminatively Trained, Multiscale, Deformable Part Model", CVPR 2008.