Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

TOWARDS HUMAN INTERACTION ANALYSIS

Maedeh Aghaei Gavari

Advisor/s: Dr. Petia Ivanova Radeva

January 2013

"Productivity is never an accident. It is always the result of a commitment to excellence, intelligent planning, and focused effort."

Paul J. Meyer

American entrepreneur and author

ACKNOWLEDGEMENT

Carpe diem. Seize the day. Opportunities, if they appear, they are to be taken, not to miss them. When I started my Master studies I never imagined how rewarding and illustrative the execution of a thesis was. You have the opportunity of deeply learning about what you are really interested in, and you discover yourself working in a collaborative environment together with your advisor and research partners; Fully advisable.

I really want to thank and dedicate this thesis to all my family, and especially to my parents. They have given to me much more than I could imagine, and they even have helped me along this adventure at good and not so good moments, and due to these they are the ones who more deserve it. Special mention deserves Petia Ivanova Radeva, my Master thesis supervisor, who has been an exceptional director in all aspects, for dedication, interest, caring, patience, encouragement, and for the unconditional support she offers.

It is a pleasure to thank all those who made this thesis possible. My greatest appreciation and friendship goes to my supportive friend, Victor Borjas, who was involved in this project from the beginning and who always helped me with his great suggestions and supports along the project.

¡Gracias a todos!

ABSTRACT

Modeling and recognizing human behaviors in a visual surveillance task is receiving increasing attention from computer vision and machine learning researchers. Such a system should deal in particularly with detecting when interactions between people occur and classifying the type of interaction.

In this work we study a flexible model for detecting human interactions. This has been done by detecting the people in the scene and retrieving their corresponding pose and position sequentially in each frame of the video. To achieve this goal our work relies on robust object detection algorithm which is based on discriminatively trained part based models to detect the human bodies in videos. We apply a 'Gaussian Mixture Models based' method for background subtraction and human segmentation. The output from the segmentation method which is labeled human body is combined with the background subtraction methods to obtain a bounding box around each person in images to improve the task of human body pose detection.

To gain more precise pose detection models, we trained the algorithm on large, challenging but reliable dataset (PASCAL 2010) [49]. Our method is applied in home-made database comprising depth data from Kinect sensors. After successfully getting in every image sequence the corresponding label for each person as well as their pose and position, understanding of human motion comes naturally which is an important step towards human interaction analysis.

TABLE OF CONTENT

1. INTRODUCTION

1.1. TOWARDS HUMAN BEHAIVIOR ANALYSIS

Since the 90's human behavior analysis is one of the biggest challenges in artificial intelligence and one of the most important topics in computer vision science. Goal of this type of studies is to automatically analyze ongoing activities from an unknown video (e.g. a sequence of image frames). There are many applications related to this subject, as for example; surveillance systems (e.g. airport, bank, train station, etc.), virtual reality (e.g. interactive virtual worlds, virtual studios teleconferencing, etc.), motion analysis (e.g. choreography of dance, clinical studies of orthopedic patients, etc.) and Human-Robot Interaction (HRI).

The first step to achieve human behavior analysis in our work is recognition of human body in the different scenes. Methods for recognizing human body and their poses in individual frames have become increasingly popular because they are vital to achieve full automation in tracking. These models usually are sensitive on training and should be trained precisely on authentic datasets. We choose PASCAL2010 challenge as a large, convenient and trusty training set to train the pose estimation models. The second step to achieve our goal is to track human body articulation. This is a difficult problem due to the high dimensionality of the state space and the inherent ambiguity that arises from using 2D image features to estimate 3D pose parameters. Recovering the coordinates of various joints of the human body from an image or the relative location of human body is a critical step for several model-based human tracking and optical motion capture systems. Another difficulty of any tracking system could be also the problem of labeling of the moving objects. In the end, any given proposal to solve any problem must be tested over a reliable test set. To test our proposal, we provide our own home-made test set of video frames using Kinect.

1.2. ANALYSIS OF RELATED WORK

Pose estimation on video has been addressed in many previous works, either using multiple cameras [3] or a single camera. The outlines in [3] introduce rigid motion estimation for the multi-camera system itself using all information of all cameras simultaneously even in the case of non-overlapping views of the cameras. Most of the detection analysis approaches as well as our proposed methodology use single camera as their dataset provider. However, all these methods still suffer from ill-conditioned pose estimation problems which cause flat minima in translation and rotation error functions. Furthermore the relatively small viewing angle is also a problem which influences the accuracy of the estimation. In our methodology we find capturing depth data beside RGB data for each image as a way to overcome this problem.

In [5,14] a novel concept called 'pictorial structure' has been optimized to overcome the problem of computational difficulty of matching pictorial structure to image. Dynamic programming approach is used for efficiently assembling candidate parts into 'pictorial structures'. Dedicated detectors for each body part are learned considering both the appearance of body parts (head, limbs, hands) and the geometry of their assemblies using SVM and RVM classifiers. Pictorial structure models [15, 20] define a matching problem where parts have an individual match cost in a dense set of locations, and their geometric arrangement is captured by a set of 'springs' connecting pairs of parts. The patchwork of parts model from [2] is similar, but it explicitly considers how the appearance models of overlapping parts interact.

The model we use in this work to estimate the human's body pose is largely based on the pictorial structures framework from [15] and [20]. We use a dense set of possible positions and scales in an image, and define a score for placing a filter at each of these locations. The geometric configuration of the filters is captured by a set of deformation costs called 'springs' which connecting each part filter to the root filter, leading to a star-structured pictorial structure model. This model does not model interactions between overlapping parts.

Matching a deformable model to an image is a difficult optimization problem. Local search methods require initialization near the correct solution [2, 7, 43]. To guarantee a globally optimal match, more aggressive search is needed. One popular approach for part-based models is to restrict part locations to a small set of possible locations returned by an interest point detector [1, 18, 42]. Tree (and star) structured pictorial structure models [9, 15, 19, 45] allow for the use of dynamic programming and generalized distance transforms to efficiently search over all possible object configurations in an image, without restricting the possible locations for each part. We use these techniques for matching our models to images.

A similar and interesting work related to 'pictorial structures' also is done in [45]. This work addressed together three problems of face detection, pose estimation and landmark localization in real world cluttered images. They have created a model to encode elastic deformation and three dimensional structures based on mixture of trees with a shared pool of parts.

In [13,11,8] pose detection problem is addressed as inference over a generative model. Top-down reasoning as well as bottom-up reasoning mechanisms are integrated and can carry out the inference tasks in parallel. Human body configuration is modeled by a Markov network and pose estimation is done by inferring to the image cues such as appearance, shape, edge, and color. The next is to reassemble segments which are consistent with the constraints on kinematic properties of the image. In other approaches [12], segmentation and skeleton recognition methods applied to recover the segmentation mask associated with the human figure. Proposed solutions either assume very restrictive appearance models [5] or make use of cues, such as skin color [23] and face position [11], which are not reliable and can be found only in specific classes of images (e.g. sport players or athletes).

A large body of work in pose estimation focuses on the simpler problem of estimating the 3D pose from human body silhouettes [16, 17, 21, 22, 34]. It is possible to learn a map from silhouettes to poses, either direct, one-to-many or as a probabilistic mixture. However, silhouettes are inherently ambiguous as very different poses can generate

similar silhouettes, so to obtain good results either we resort to complex mixture models or restrict the set of poses, or use multiple views [34]. Besides silhouettes and appearance, motion is another important cue that can be used for pose estimation and tracking [4, 24]. Most works assume a parametric model of the optical flow, which can be either designed or learned from examples. But complex motion models are not the only way to make use of motion information. As shown in [35], simple image differences can provide an effective cue for pedestrian detection.

Our approach towards motion analysis of the people composed of two steps, the first step is to detect human bodies in the video frames and to use the geometric information of each human body location from this step to perform segmentation and background extraction. The next step is to retrain the algorithm to get exclusive models for each of four principle views each person can take (Frontal, Rear, Right, Left).

There is a significant body of work on deformable models of various types for object detection, including several kinds of deformable template models [7, 37, 43], and a variety of part-based models [2, 6, 9, 15, 18, 20, 28, 42]. Significant variations in shape and appearance, such as caused by extreme viewpoint changes, are not well captured by a 2D deformable model. Aspect graphs [31] are a classical formalism for capturing significant changes that are due to viewpoint variation. Mixture models provide a simpler alternative approach. For example, it is common to use multiple templates to encode frontal and side views of faces and cars [36]. Mixture models have been used to capture other aspects of appearance variation as well, such as when there are multiple natural subclasses in an object category [38]. To detect human body in our approach we used [46] an object detection system based on mixtures of multi-scale deformable part models.

We have applied our proposed methodology on our database of RGBD videos and the results we obtained were promising. In any experiment we performed on our videos, we achieved high accuracy in pose estimation of each person. Getting pose of each person in sequential frames besides having their corresponding position and some other individual information for each person in each frame, makes us able to move some steps toward human body motion and interaction analysis.

This thesis is organized as follows. The next chapter explains two main challenges of human body detection and segmentation that our proposed methodology fundamentally structured based on them. The third chapter discusses the procedure to train different models which are employed to detect specific human's body pose in an image. The forth chapter is organized to introduce the characteristic of the training dataset and the test dataset which we used in this work to evaluate our proposed method and the evaluation results. At the last part, in fifth chapter we present a summary of what we have done in this work and the future lines we aim to follow in the future.

2. PROBLEM ANALYSIS

In this chapter, we study in detail two main challenges of human body detection and segmentation which we are required to deal with them in this task to achieve our goal of human motion analysis in the indoor scenes. Before all else, we provide detailed description about part-based model approach which is a robust method to detect objects in complex environment and we take advantage of it to find human bodies in each frame of the videos. We explain procedure to train the part-based models and how these trained models are used to estimate the body poses. We also discuss the characteristic of the models and different tasks that have been done to speed up the detection using these models without any loss of information.

Segmentation strategy also is one of the main tasks in this work which is described in this chapter. To obtain segmented images we utilize Gaussian mixture models (GMM) to extract the background from each frame of the video. GMM also are used to assign a label to every person in the video which is an essential task in motion analysis and tracking projects.

2.1. DISCRIMINATIVELY TRAINED PART BASED MODEL

Object recognition is a difficult problem especially because the appearance of objects can vary greatly due to illumination or viewpoint. Non-rigid deformations, and intra-class variability in shape and other visual properties are the other reasons that make this task even more complicated. For example, people wear different clothes and take a variety of poses while cars come in a various shapes and colors.

The object detection system we describe is a robust approach in object detection which with high accuracy is able to detect highly variable objects using mixtures of multi-scale deformable part models. These models are trained using a discriminative procedure that only requires bounding boxes for the objects in a set of images. This approach builds on the pictorial structures framework [15, 20]. Pictorial structures represent objects by a collection of parts arranged in a deformable configuration. Each part captures local appearance properties of an object while the deformable configuration is characterized by spring-like connections between certain pairs of parts. While deformable models can capture significant variations in appearance, a single deformable model is often not expressive enough to represent a rich object category. The system described here uses mixture models to deal with these more significant variations.

We can think of the detector as a classifier which takes as input an image, a position within that image, and a scale. The classifier determines whether or not there is an instance of the target category at the given position and scale. Since the model is a simple filter we can compute a score as $\beta \cdot \Phi(x)$ where β is the filter, x is an image with a specified position and scale, and $\Phi(x)$ is a feature vector.

Star-structured part-based model is defined by a 'root' filter (analogous to the filter β) plus a set of parts filters and associated deformation models. The score of one of the star models at a particular position and scale within an image is the score of the root filter at the given location plus the sum over parts of the maximum, over placements of that part, of the part filter score on its location minus a deformation cost measuring the deviation of the part from its ideal location relative to the root. Equation (1) shows star models score formulation

where F_i is the filter i, d_i is deformation parameters, and (dx_i^2 , dy_i^2) demonstrates the displacements:

$$\text{Score } (p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H , p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2 , dy_i^2) \qquad (1)$$

Both root and part filter scores are defined by the dot product between a filter (a set of weights) and a sub-window of a feature pyramid computed from the input image. Figure 1 shows a star model for the person category. In the models of this methodology the part filters capture features at twice the spatial resolution relative to the features captured by the root filter. In this way visual appearance is modeled at multiple scales.

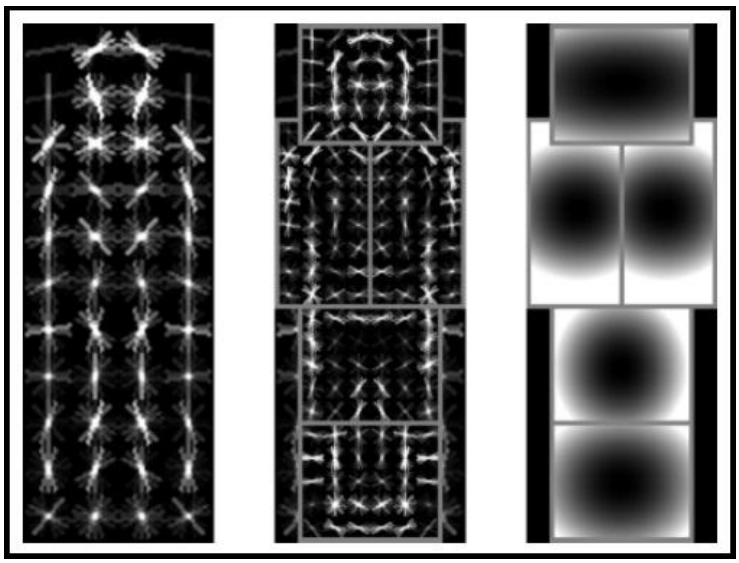


Figure 1. Detection obtained with a single component person model. The model is defined by a coarse root filter (the left most shape), several higher resolution part filters (the middle shape) and a spatial model for the location of each part relative to the root (the right most shape). The filters specify weights for histogram of oriented gradients features. Their visualization shows the positive weights at different orientations. The visualization of the spatial models reflects the 'cost' of placing the center of a part at different locations relative to the root.

To train models using partially labeled data a latent variable formulation of MI-SVM is used [3] that is called latent SVM (LSVM). In a latent SVM each example x is scored by a function of the following form:

$$f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z) \qquad (2)$$

Here β is a vector of model parameters, z are latent values, and $\Phi(x, z)$ is a feature vector. The set $Z(x)$ defines the possible latent values for an example x. A binary label for x can be obtained by thresholding its score. In analogy to classical SVMs, β is trained from labeled examples $D = (\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle)$, where $y_i \in \{-1, 1\}$, by minimizing the objective function:

$$L_D(\beta) = \frac{1}{2} ||\beta||^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \qquad (3)$$

where $\max(0, 1 - y_i f_\beta(x_i))$ is the standard hinge loss and the constant C controls the relative weight of the regularization term. Note that if there is a single possible latent value for each example ($|Z(x_i)| = 1$) then f_β is linear in β, and we obtain linear SVMs as a special case of latent SVMs.

In the case of one of the star models, β is the concatenation of the root filter, the part filters, and deformation cost weights, z is a specification of the object configuration, and $\Phi(x, z)$ is a concatenation of sub-windows from a feature pyramid and part deformation features. We note that (2) can handle very general forms of latent information. For example, z could specify a derivation under a rich visual grammar.

The second class of models represents an object category by a mixture of star models. The score of a mixture model at a particular position and scale is the maximum over components, of the score of that component model at the given location. In this case the latent information, z, specifies a component label and a configuration for that component. Figure 2 shows a mixture model for the bicycle category. Object models are defined by filters that score sub-windows of a feature pyramid. Feature sets are organized similar to the HOG features and by doing principal component analysis on HOG features the dimensionality of the feature vector can be significantly reduced with no noticeable loss of information.

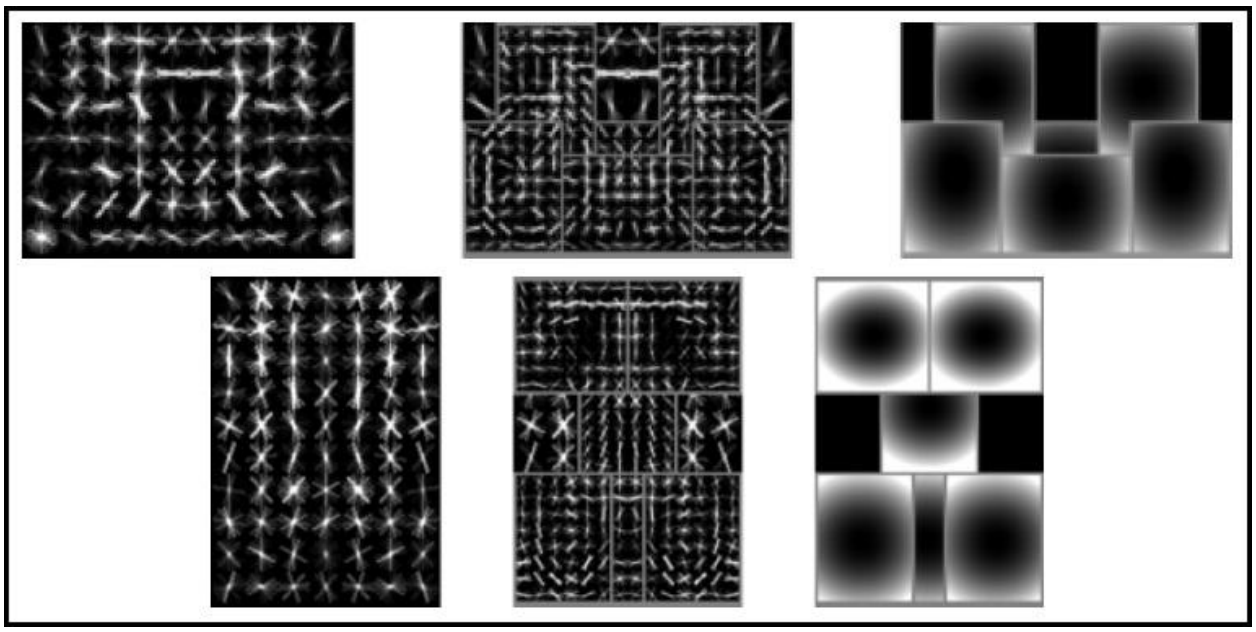


Figure 2. Detections obtained with a two component bicycle model. These examples illustrate the importance of deformations mixture models. In this model the first component captures sideways views of bicycles while the second component captures frontal and near frontal views. The sideways component can deform to match a 'wheelie'.

2.1.1. PEOPLE DETECTION MODELS

All of the models of detection involve linear filters that are applied to dense feature maps. A feature map is an array whose entries are d-dimensional feature vectors computed from a dense grid of locations in an image. Intuitively each feature vector describes a local image patch. In practice, a variation of the HOG features is used which is independent of the specific choice of features.

A filter is a rectangular template defined by an array of d-dimensional weight vectors. The response, or score, of a filter F at a position (x, y) in a feature map G is the 'dot product' of the filter and a sub-window of the feature map with top-left corner at (x, y):

$$\sum_{x', y'} F[x', y'] \cdot G[x + x', y + y']$$

The score is defined at different positions and scales in an image. This is done using a feature pyramid, which specifies a feature map for a finite number of scales in a fixed range. In practice, feature pyramids are computed by computing a standard image pyramid via repeated smoothing and sub-sampling, and then computing a feature map from each level of the image pyramid. Figure 3 illustrates the construction.

The scale sampling in a feature pyramid is determined by a parameter λ defining the number of levels in an octave. λ is the number of levels that is necessary to go down in the pyramid to get to a feature map computed at twice the resolution of another one. Fine sampling of scale space is important for obtaining high performance with deformable part based models. Initially in the basic resolution of the image the root filters find the silhouette of the human body without identifying their parts and in the next step the part filters are placed at twice the spatial resolution of the root filter.

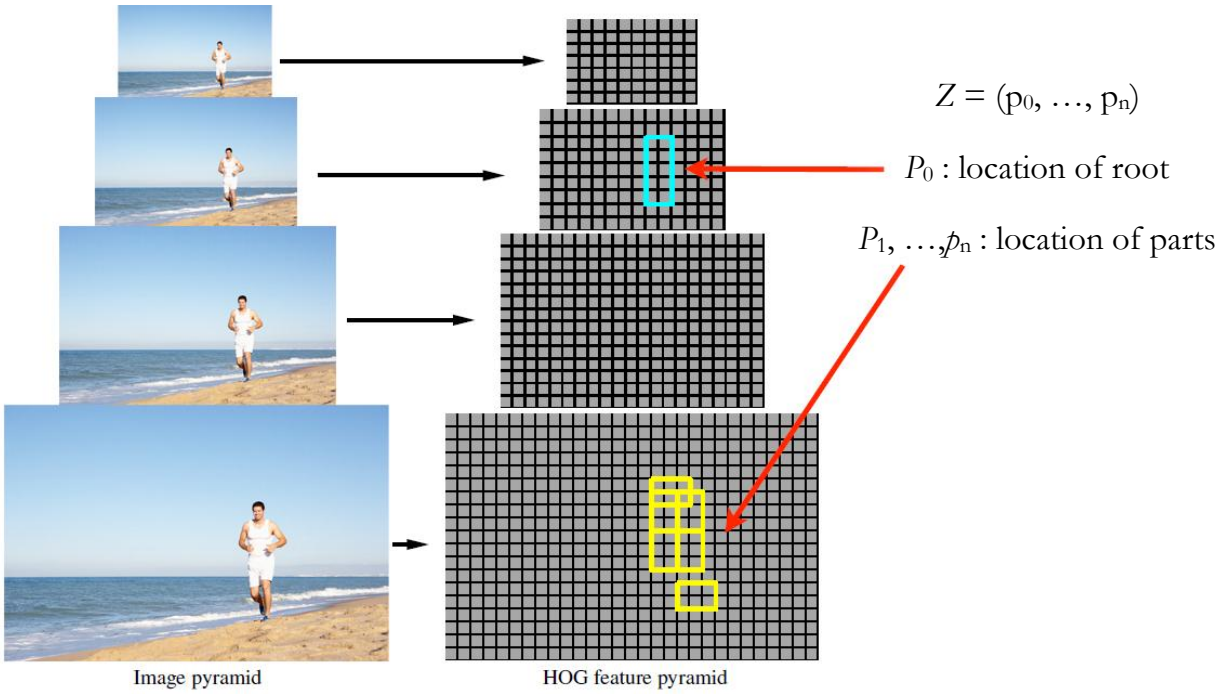


Figure 3. A feature pyramid and an instantiation of a person model within that pyramid. The part filters are placed at twice the spatial resolution of the placement of the root.

Let F be a $w \times h$ filter. Let H be a feature pyramid and $p = (x, y, l)$ specify a position (x, y) in the l-th level of the pyramid. Let $\Phi(H, p, w, h)$ denote the vector obtained by concatenating the feature vectors in the $w \times h$ sub-window of H with top-left corner at p in row-major order. The score of F at p is $F' \cdot \Phi(H, p, w, h)$, where F' is the vector obtained by concatenating the weight vectors in F in row-major order.

2.1.2. DEFORMABLE PART MODELS

The star models are defined by a coarse root filter that approximately covers an entire object and higher resolution part filters that cover smaller parts of the object. Figure 3 illustrates an instantiation of such a model in a feature pyramid. The root filter location defines a detection window (the pixels contributing to the part of the feature map covered by the filter). The part filters are placed λ levels down in the pyramid, so the features at that level are computed at twice the resolution of the features in the root filter level.

With this approach the part filters capture finer resolution features that are localized to greater accuracy when compared to the features captured by the root filter. If we consider building a model for a face, the root filter could capture coarse resolution edges such as the face boundary while the part filters could capture details such as eyes, nose and mouth. A model for an object with n parts is formally defined by a $(n + 2)$-tuple $(F_0, P_1, \ldots, P_n, b)$ where F_0 is a root filter, P_i is a model for the i-th part and b is a real-valued bias term. Each part model is defined by a 3-tuple (F_i, v_i, d_i) where F_i is a filter for the i-th part, v_i is a two-dimensional vector specifying an 'anchor' position for part i relative to the root position, and d_i is a four dimensional vector specifying coefficients of a quadratic function defining a deformation cost for each possible placement of the part relative to the anchor position.

An 'object hypothesis' specifies the location of each filter in the model in a feature pyramid, $z = (P_0, \ldots, P_n)$, where $P_i = (x_i, y_i, l_i)$ specifies the level and position of the i-th filter. We require that the level of each part is such that the feature map at that level was computed at twice the resolution of the root level, $l_i = l_0 - \lambda$ for $i > 0$. The score of a hypothesis is given by the scores of each filter at their respective locations (the data term) minus a deformation cost that depends on the relative position of each part with respect to the root (the spatial prior), plus the bias,

$$\text{Score}(P_0, \ldots, P_n) = \sum_{i=0}^{n} F_i' \cdot \Phi(H, P_i) - \sum_{i=1}^{n} d_i \cdot \Phi_d(dx_i, dy_i) + b$$

where

$$(dx_i , dy_i) = (x_i , y_i) — (2(x_{0,} y_0) + v_i)$$

gives the displacement of the *i*-th part relative to its anchor position as shown in figure 4 and

$$\Phi_d(dx , dy) = (dx , dy , dx^2 , dy^2)$$

are deformation features.

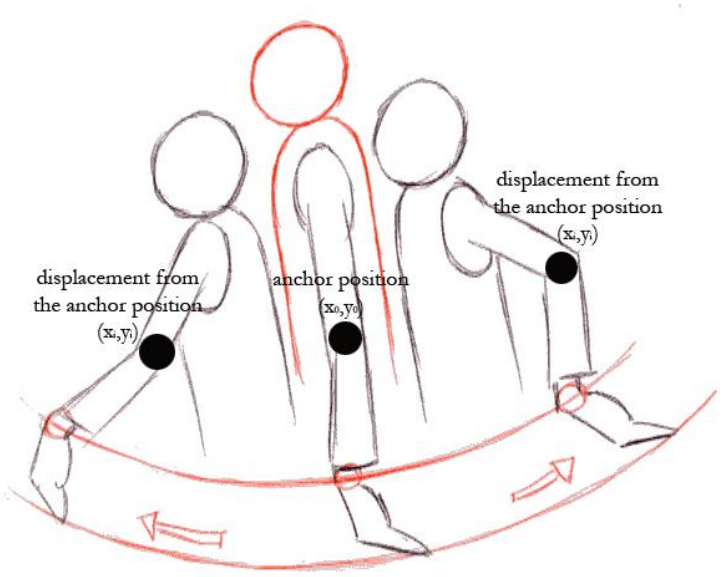


Figure 4: Displacement of the i-th part (i.e. arm) from its anchor position (x_0 y_0)

The deformation cost is an arbitrary separable quadratic function of the displacements and the bias term is introduced in the score to make the scores of multiple models comparable when we combine them into a mixture model.

2.1.3. FILTER MATCHING

To detect objects in an image an overall score for each root location is computed according to the best possible placement of the parts:

$$\text{Score}\ (P_0) = \max_{P_1,\ldots,P_n} \text{Score}\ (P_0\ ,\ \ldots\ ,\ P_n)$$

High-scoring root locations define detections while the locations of the parts that yield a high-scoring root location define a full object hypothesis. By defining an overall score for each root location, multiple instances of an object can be detected (we assume there is at most one instance per root location). This approach is related to sliding-window detectors because we can think of score(p_0) as a score for the detection window specified by the root filter. Dynamic programming and generalized distance transforms (min-convolutions) are the methods used to compute the best locations for the parts as a function of the root location. The resulting method is very efficient, taking $O(nk)$ time once filter responses are computed, where n is the number of parts in the model and k is the total number of locations in the feature pyramid. Following we briefly describe the method.

Let $R_{i,l}(x, y) = F_i'\ .\ \Phi(H, (x, y, l))$ be an array storing the response of the i-th model filter in the l-th level of the feature pyramid. The matching algorithm starts by computing these responses. Note that $R_{i,l}$ is a cross-correlation between F_i and level l of the feature pyramid. After computing filter responses the responses of the part filters are transformed to allow for spatial uncertainty:

$$D_{i,l}(x\ ,y) = \max_{dx\ ,\ dy} (R_{i,l}(x + dx\ ,y + dy) - d_i\ .\ \Phi_d\ (dx\ ,\ dy))$$

This transformation spreads high filter scores to nearby locations, taking into account the deformation costs. The value $D_{i,l}(x\ ,y)$ is the maximum contribution of the i-th part to the score of a root location that places the anchor of this part at position $(x\ ,y)$ in level l. The

overall root scores at each level can be expressed by the sum of the root filter response at that level, plus shifted versions of transformed and sub-sampled part responses:

$$\text{score}(x_0, y_0, l_0) = R_{0,l_0}(x_0, y_0) + \sum_{i=1}^{n} D_{i,l_0-\lambda}(2(x_0, y_0) + v_i) + b \qquad (4)$$

Recall that λ is the number of levels we need to go down in the feature pyramid to get to a feature map that was computed at exactly twice the resolution of another one. To understand equation (4) note that for a fixed root location we can independently pick the best location for each part because there are no interactions among parts in the score of a hypothesis. The transformed arrays $D_{i,l}$ give the contribution of the *i*-th part to the overall root score, as a function of the anchor position for the part. So we obtain the total score of a root position at level *l* by adding up the root filter response and the contributions from each part, which are pre-computed in $D_{i,l_0-\lambda}$. Figure 5 illustrates the matching process:

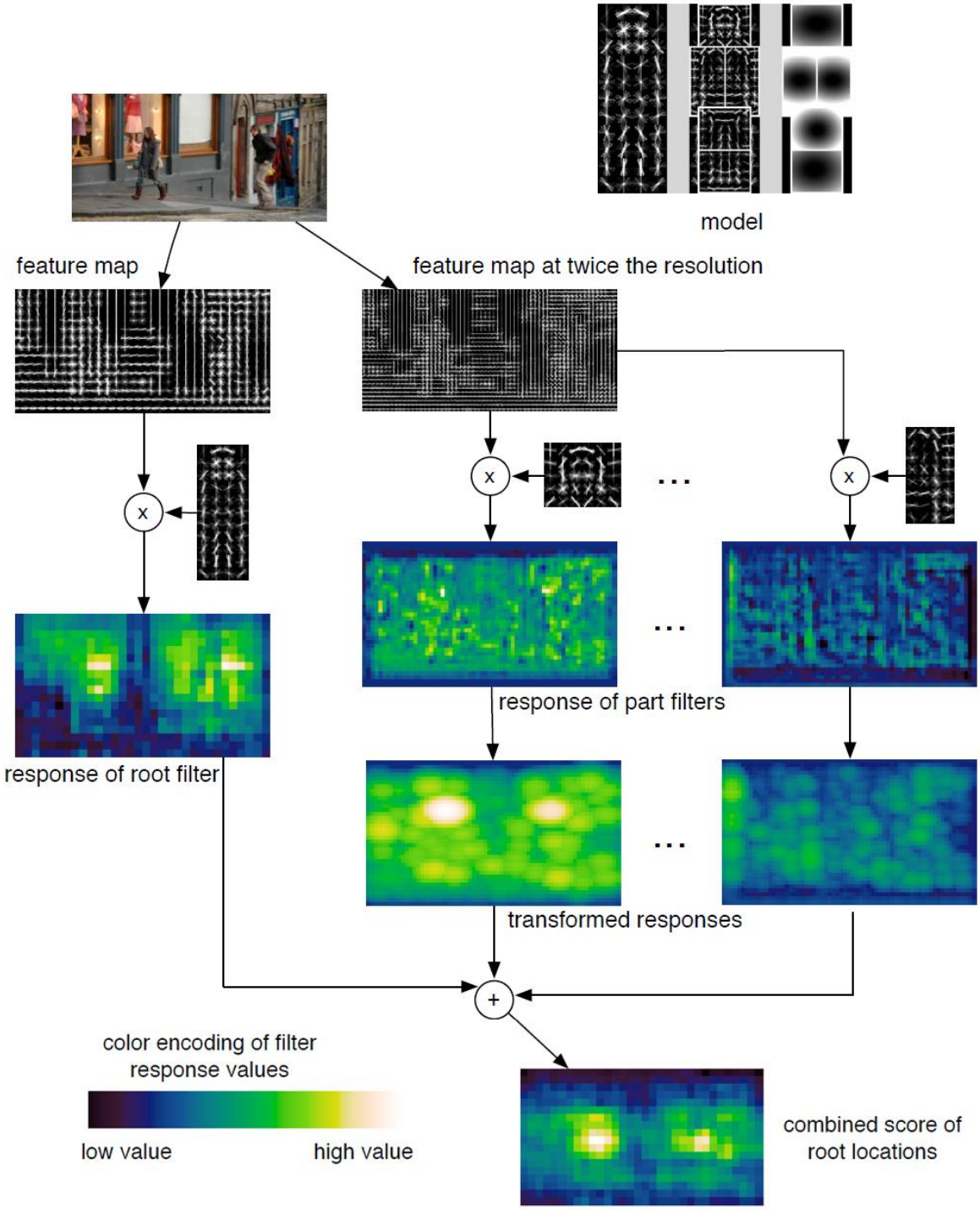


Figure 5. The matching process at one scale. Responses from the root and part filters are computed different resolutions in the feature pyramid. The transformed responses are combined to yield a final score for each root location. We show the responses and transformed responses for the 'head' and 'right shoulder' parts. Note how the 'head' filter is more discriminative. The combined scores clearly show two good hypothesis for the object at this scale.

2.1.4. MIXTURE MODELS

A mixture model with m components is defined by a m-tuple, $M = (M_1, \ldots, M_m)$, where M_c is the model for the C-th component. An object hypothesis for a mixture model specifies a mixture component, $1 < c < m$, and a location for each filter of M_C, $z = (C, P_0, \ldots, p_{n_c})$. Here n_c is the number of parts in M_c. The score of this hypothesis is the score of the hypothesis $z' = (P_0, \ldots, p_{n_c})$ for the C-th model component.

As in the case of a single component model, the score of a hypothesis for a mixture model can be expressed by a dot product between a vector of model parameters and a vector $\psi(H, z)$. For a mixture model the vector β is the concatenation of the model parameter vectors for each component. The vector $\psi(H, z)$ is sparse, with non-zero entries defined by $\psi(H, z')$ in a single interval matching the interval of β_C in β:

$$\beta = (\beta_1, \ldots, \beta_m)$$

$$\psi(H, z) = (0, \ldots, 0, \psi(H, z'), 0, \ldots, 0)$$

with this construction:

$$\beta \cdot \psi(H, z) = \beta \cdot \psi(H, z')$$

To detect objects using a mixture model the same matching algorithm is used which described above to find root locations that yield high scoring hypotheses independently for each component.

2.1.5. FEATURE DISCRIPTORS

The introduction of new local and semi-local features has played an important role in advancing the performance of object recognition methods. These features are typically invariant to illumination changes and small deformations. Many recent approaches use wavelet-like features [30, 41] or locally-normalized histograms of gradients [10, 29]. Other methods, such as [44], learn dictionaries of local structures from training images. In this work, histogram of gradient (HOG) feature from [10] is used as a starting point, and then a variation employs which reduces the feature size with no loss in performance. As in [26], Principal Component Analysis (PCA) is used to discover low dimensional features. Here we describe the 36-dimensional histogram of oriented gradients (HOG) features from [10] and introduce an alternative 13-dimensional feature set that captures essentially the same information.

❖ HOG Features

➢ Pixel-Level Feature Maps

Let $\theta(x, y)$ and $r(x, y)$ be the orientation and magnitude of the intensity gradient at a pixel (x, y) in an image. As in [10], the gradients are computed using finite difference filters, [-1 , 0 , +1] and its transpose. For color images the color channel with the largest gradient magnitude is used to define θ and r at each pixel. The gradient orientation at each pixel is discretized into one of p values using either a contrast sensitive (B_1), or insensitive (B_2), definition:

$$B_1(x, y) = \text{round}\left(\frac{p\theta(x,y)}{2\pi}\right) \bmod p$$

$$B_2(x, y) = \text{round}\left(\frac{p\theta(x,y)}{\pi}\right) \bmod p$$

Below we use B to denote either B_1 or B_2.

We define a pixel-level feature map that specifies a sparse histogram of gradient magnitudes at each pixel. Let $b \in \{0, \ldots, p - 1\}$ range over orientation bins. The feature vector at (x, y) is:

$$F(x, y)_b = \begin{cases} r(x,y) & if \;\; b = B(x,y) \\ b & otherwise \end{cases}$$

We can think of F as an oriented edge map with p orientation channels. For each pixel, we select a channel by discretizing the gradient orientation. The gradient magnitude can be seen as a measure of edge strength.

- Spatial Aggregation

Let F be a pixel-level feature map for a $w \times h$ image. Let $k > 0$ be a parameter specifying the side length of a square image region. Let's define a dense grid of rectangular 'cells' and aggregate pixel-level features to obtain a cell-based feature map C, with feature vectors $C(i, j)$ for $0 \leq i \leq [(w - 1)/k]$ and $0 \leq j \leq [(h - 1)/k]$. This aggregation provides some invariance to small deformations and reduces the size of a feature map.

The simplest approach for aggregating features is to map each pixel (x, y) into a cell $([x/k], [y/k])$ and define the feature vector at a cell to be the sum (or average) of the pixel-level features in that cell. Rather than mapping each pixel to a unique cell we follow [10] and use a 'soft binning' approach where each pixel contributes to the feature vectors in the four cells around it using bilinear interpolation.

- Normalization and Truncation

Gradients are invariant to changes in bias. Invariance to gain can be achieved via normalization. In [10] four different normalization factors for the feature vector $C(i, j)$ is used. We can write these factors as $N_{\delta,\gamma}(i, j)$ with $\delta, \gamma \in \{-1, 1\}$:

$$N_{\delta,\gamma}(i,j) = (||C(i,j)||^2 + ||C(i+\delta,j)||^2 + ||C(i,j+\gamma)||^2 + ||C(i+\delta,j+\gamma)||^2)^{1/2}$$

Each factor measures the 'gradient energy' in a square block of four cells containing (i,j). Let $T_\alpha(v)$ denote the component-wise truncation of a vector v by α (the i-th entry in $T_\alpha(v)$ is the minimum of the i-th entry of v and α). The HOG feature map is obtained by concatenating the result of normalizing the cell-based feature map C with respect to each normalization factor followed by truncation:

$$H(i,j) = \begin{pmatrix} T_\alpha\,(C(i,j)/N_{-1,-1}(i,j)) \\ T_\alpha\,(C(i,j)/N_{+1,-1}(i,j)) \\ T_\alpha\,(C(i,j)/N_{+1,+1}(i,j)) \\ T_\alpha\,(C(i,j)/N_{-1,+1}(i,j)) \end{pmatrix}$$

Commonly used HOG features are defined using $p = 9$ contrast insensitive gradient orientations (discritized with B_2), a cell size of $k = 8$ and truncation $\alpha = 0.2$. This leads to 36-dimensional feature vector.

❖ PCA and Dimensionality Reduction

By applying PCA on large number of 36-dimensional HOG features from different resolution, results show the eigenvalues indicate that the linear subspace spanned by the top 11 eigenvectors captures essentially all the information in a HOG feature. In fact the same detection performance obtains using the original 36-dimensional features or 11-dimensional features defined by projection to the top eigenvectors. Using lower dimensional features leads to model with fewer parameters and speeds up the detection and learning algorithms; however some of the gain is lost because a relatively costly projection step is needed when computing feature pyramids.

2.1.6. LATENT SVM

A latent SVM leads to a non-convex optimization problem. However, a latent SVM is semi-convex in the sense described below, and the training problem becomes convex once latent information is specified for the positive training examples. Recall that the maximum of a set of convex functions is convex. In a linear SVM we have that $f_\beta(x) = \max \beta \cdot \Phi(x, z)$ is linear in β. In this case the hinge loss is convex for each example because it is always the maximum of two convex functions. Note that $f_\beta(x)$ as defined in (2) is a maximum of functions each of which is linear in β. Hence $f_\beta(x)$ is convex in β and thus the hinge loss, $\max(0,1- y_i f_\beta(x_i))$, is convex in β when $y_i = -1$. That is, the loss function is convex in β for negative examples. This property of the loss function is called semi-convexity. In a general latent SVM the hinge loss is not convex for a positive example because it is the maximum of a convex function (zero) and a concave function $(1- y_i f_\beta(x_i))$. Now consider a latent SVM where there is a single possible latent value for each positive example. In this case $f_\beta(x_i)$ is linear for a positive example and the loss due to each positive is convex. Combined with the semi-convexity property, (3) becomes convex.

❖ LSVM Optimization

Let Z_p specify a latent value for each positive example in a training set D. We can define an auxiliary objective function $L_D(\beta, Z_p) = L_{D(Z_p)}(\beta)$, where $D(Z_p)$ is derived from D by restricting the latent values for the positive examples according to Z_p. That is, for a positive example we set $Z(x_i) = \{z_i\}$ where z_i is the latent value specified for x_i by Z_p. Note that:

$$L_D(\beta) = \min_{Z_p} L_D(\beta, Z_p)$$

In particular $L_D(\beta) <= L_D(\beta, Z_p)$. The auxiliary objective function bounds the LSVM objective. This justifies training a latent SVM by minimizing $L_D(\beta, Z_p)$. In practice we minimize $L_D(\beta, Z_p)$ using a 'coordinate descent' approach:

1) *Relabel positive examples*: Optimize $L_D(\beta, Z_p)$ over Z_p by selecting the highest scoring latent value for each positive example:

$$z_i = \text{argmax}_{z \in Z(x_i)} \beta \cdot \Phi(x_i, z)$$

2) *Optimize beta*: Optimize $L_D(\beta, Z_p)$ over β by solving the convex optimization problem defined by $L_{D(Z_p)}(\beta)$.

Both steps always improve or maintain the value of $L_D(\beta, Z_p)$. After convergence we have a relatively strong local optimum in the sense that step 1 searches over an exponentially-large space of latent values for positive examples while step 2 searches over all possible models, implicitly considering the exponentially-large space of latent values for all negative examples. We note, however, that careful initialization of β may be necessary because otherwise we may select unreasonable latent values for the positive examples in step 1, and this could lead to a bad model. The semi-convexity property is important because it leads to a convex optimization problem in step 2, even though the latent values for the negative examples are not fixed. A similar procedure that fixes latent values for all examples in each round would likely fail to yield good results. Suppose we let Z specify latent values for all examples in D. Since $L_D(\beta)$ effectively maximizes over negative latent values, $L_D(\beta)$ could be much larger than $L_D(\beta, Z)$, and we should not expect that minimizing $L_D(\beta, Z)$, would lead to a good model.

2.2. HUMAN BODY LABELING AND SEGMENTATION

Segmentation, background extraction and human body labeling in the videos are the second step towards the goal of this project after human body detection in the videos. Detecting and labeling intruding objects is an essential step in analyzing the scene. A usually applicable assumption is that the images of the scene without the intruding objects exhibit some regular behavior that can be well described by a statistical model. If we have a statistical model of the scene, an intruding object can be detected by spotting the parts of the image that do not fit the model, this process is usually known as 'background subtraction'. Background subtraction is a widely used method in Computer Vision for separating or segmenting out the foreground objects from the background of a video. The foreground objects are defined to be the parts of the image that changes and the background is made out of the pixels that stay relatively constant. Commonly used techniques for Background Subtraction Include 'Frame Differencing', 'Running Average as Background', 'Gaussian Mixture Models (GMM)' and 'Kernel Density Estimators'

GMM based method was first introduced by Stauffer and Grimson in 1999, and now it is the most widely used method for background subtraction due to its speed, simplicity and the ease of implementation. In this method, each pixel is modeled as a mixture of Gaussian distributions and any pixel intensity value that does not fit into one of the modeled Gaussian distributions is marked as a foreground pixel. In fact, it keeps K Gaussians for each pixel presenting a multi modal distribution of pixel gray-values. At each new frame the new gray-value y is checked against all Gaussians and the best matching Gaussian is selected, if y is within a threshold of standard deviations of the mean, a new Gaussian is created else. The parameters of the matched Gaussian (weight, mean, standard deviation) are updated using a learning rate parameter. The difficulty lies in the decision whether a matched Gaussian corresponds to the background (BG) or the foreground (FG) distribution. In the following, we describe Gaussian Mixture Models in detail and then we will explain how this skill is applied in our work.

2.2.1. GAUSSIAN MIXTURE MODELS

In this approach, each pixel is modeled with a mixture of Gaussian distributions so that the model is general enough to handle common background variations. Figure 6 shows different Gaussian distribution of a same pixel under different illumination conditions.

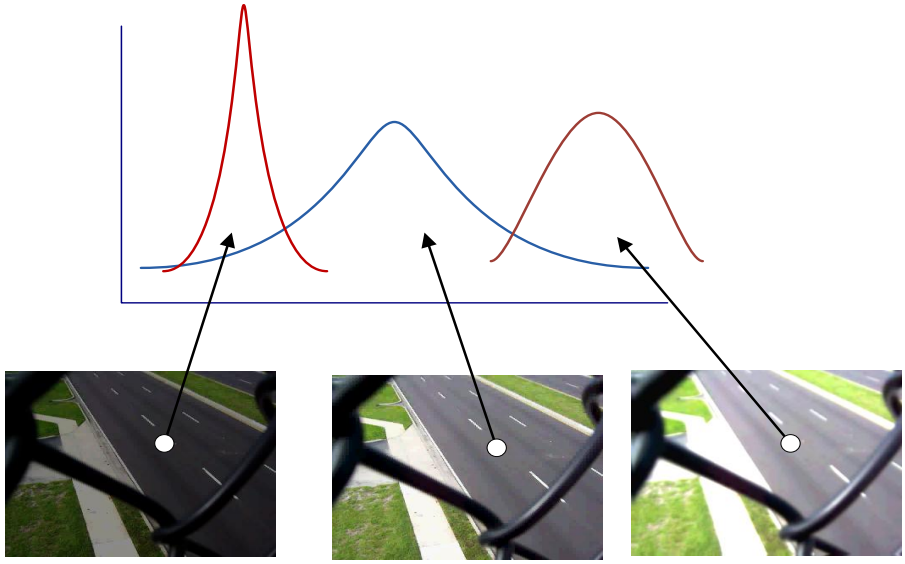


Figure 6. Gaussian distribution of a single pixel under different lighting conditions

The background subtraction involves two different tasks which are 'to learn the background model' and afterwards 'to classify each pixel as foreground or background'. Each of these tasks needs to be performed real-time, with having only the video frames as the input.

❖ Learning the background model

Following parameters of each Gaussian component need to be learned dynamically

- The parameters of Gaussians
  - Mean
  - Variance
  - Weight
- Number of Gaussians per pixel.

In our videos, a new person could come into the scene or a present person could leave it. In order to adapt to changes, we can update the training set by adding new samples and discarding the old ones. We choose a reasonable time period T and at time t we have:

$$X_T = \{x^{(t)}, \ldots, x^{(t-T)}\}$$

For each new sample update the training dataset X_T and Re-estimate $\hat{p}(\vec{x} \mid X, BG)$ (background model). However, among the samples from the recent history there could be some values that belong to the foreground objects and we should denote this estimate as $\hat{p}(\vec{x}^{(t)} \mid X_T, BG+FG)$ where $\vec{x}^{(t)}$ is the value of a pixel at time t in RGB or some other colorspace. We use GMM with M Gaussians:

$$\hat{p}(\vec{x} \mid X_T, BG+FG) = \sum_{m=1} \hat{\pi}_m N(\vec{x};\ \hat{\vec{\mu}}_m\ , \hat{\sigma}_m^2 I)$$

where $(BG+FG)$ is full scene model, $\hat{\vec{\mu}}_1$, …, $\hat{\vec{\mu}}_M$ are estimates of the means and $\hat{\vec{\sigma}}_1$, …, $\hat{\vec{\sigma}}_M$ are estimates of the variances that describe the Gaussian components.

The covariance matrices are assumed to be diagonal and the identity matrix I has proper dimensions. The mixing weights denoted by $\hat{\pi}_m$ are non-negative and add up to one. Given a new data sample $\vec{x}^{(t)}$ at time t the recursive update equations are:

$$\hat{\pi}_m \leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m)$$
$$\hat{\vec{\mu}}_m \leftarrow \hat{\vec{\mu}}_m + o_m^{(t)}(\alpha / \hat{\pi}_m)\ \vec{\delta}_m$$
$$\hat{\sigma}_m^2 \leftarrow \hat{\sigma}_m^2 + o_m^{(t)}(\alpha / \hat{\pi}_m)\ (\vec{\delta}_m^T\ \vec{\delta}_m - \hat{\sigma}_m^2)$$

where,

$$\vec{\delta}_m = \vec{x}^{(t)} - \hat{\vec{\mu}}_m$$

These equations are executed for each Gaussian component for each pixel at the arrival of each video frame. Instead of the time interval *T* that was mentioned above, here constant α describes an exponentially decaying envelope that is used to limit the influence of the old data. We keep the same notation having in mind that approximately $\alpha = 1/T$.

For a new sample, the ownership $o_m^{(t)}$ is set to 1 for the 'close' component with largest $\hat{\pi}_m$ and the others are set to zero. We define that a sample is 'close' to a component if the Mahalanobis distance from the component is for example less than three standard deviations. The squared distance from the *m*-th component is calculated as:

$$D_m^2(\vec{x}^{(t)}) = \vec{\delta}_m^T \vec{\delta}_m / \hat{\sigma}_m^2$$

If there are no 'close' components, a new component is generated with $\hat{\pi}_{M+1}=\alpha$, $\hat{\vec{\mu}}_{M+1}=\vec{x}^{(t)}$ and $\hat{\sigma}_{M+1}=\sigma_0$ where σ_0 is some appropriate initial variance. If the maximum number of components is reached, we discard the component with smallest $\hat{\pi}_m$.

The presented algorithm demonstrates an on-line clustering algorithm. Usually, the intruding foreground objects will be represented by some additional clusters with small weights $\hat{\pi}_m$. Therefore, we can approximate the background model by the first *B* largest clusters:

$$\hat{p}(\vec{x} \mid X_T, BG) \sim \sum_{m=1} \hat{\pi}_m N(\vec{x};\ \hat{\vec{\mu}}_m\ ,\ \hat{\sigma}_m^2 I)$$

if the Gaussians are sorted to have descending weights $\hat{\pi}_m$ we have:

$$B = \arg\min_b \left(\sum_{m=1}^{b} \hat{\pi}_m > (1 - c_f)\right)$$

where c_f is the measure of the maximum portion of data that can belong to *FG* without influencing the *BG* model. For example, if a new object comes into a scene and remains static for some time, it will probably generate an additional stabile cluster. Since the old

background is occluded, the weight π_{B+1} of the new cluster will be constantly increasing. If the object remains static long enough, its weight becomes larger than c_f and it can be considered to be part of the background.

❖ Classifying pixels as background or foreground

Pixel-based background subtraction involves a decision if the pixel belongs to background (BG) or foreground object (FG). Bayesian decision *R* is made by:

$$R = \frac{p(BG|\vec{x}^{(t)})}{p(FG|\vec{x}^{(t)})} = \frac{p(\vec{x}^{(t)}|BG)p(BG)}{p(\vec{x}^{(t)}|FG)p(FG)}$$

while $\vec{x}^{(t)}$ = value of the pixel at time *t* in RGB color space, $p(\vec{x}|BG)$ is background model and $\hat{p}(\vec{x}|X, BG)$ is estimated model based on the training set *X*. We assume that the samples in *X* are independent and the main problem is how to efficiently estimate the density function and to adapt it to possible changes.

Initially *P*(*FG*) is set equal to *P*(*BG*) , assuming the foreground objects are unknown and we assumed uniform distribution for the foreground object appearance $p(\vec{x}^{(t)}|BG) = C_{FG}$. Therefore, if $p(\vec{x}^{(t)}|BG) > C_{thr}$ the algorithm decides that pixel belongs to background where C_{thr} is a threshold value [48].

2.2.2. GMM APPLICATION

Segmentation with depth data has been considered by several authors. In order to apply our proposed methodology, we acquired home-made videos using Kinect to make our own dataset of RGBD images. The Kinect sensors capture in each frame the depth information of the scene along with their corresponding synchronized RGB image. Figure 7 shows a frame from our videos. Our system makes use of both depth and visible light image data to provide the segmentation.

Figure 7. A frame from our videos. This video is made inside an office including three people interacting with each other. The left side image shows the RGB image, while the right side image shows the corresponding depth image. Closer objects to the camera are shown darker in depth image.

Unlike visible light images, depth data usually contains misreading. Misreading occurs if the camera cannot see the laser beam because of occlusions or variation in lighting conditions. Figure 8 (a) shows raw depth data of a frame from our videos. Misread points are shown by black holes in the image. In this work, we employ a simple algorithm to fill in missing values for depth readings. Specifically, we apply the Successive Over Relaxation (SOR) method to iteratively extrapolate the 3D data of the missing pixels. This iterative extrapolation method alternates between two steps. The first step computes values of the missing pixels given their neighborhood while the second step computes values of the missing pixels given their previous values. The method terminates when the values of the

missing pixels do not change. In this work, initial values for missing pixels are set to zero. Figure 8 (b) shows the raw depth frame after filling up the reading errors of depth sensors. Once this step is done, we can go through the next step which is to apply GMM on the depth data to extract from the background each human body as the foreground. For every pixel, we analyze the different depth values it takes along the entire video. We analyze the histogram of these values and look for the farthest and more constant values of depth along the time *T*. These long lasting values correspond to the background. We label within each pixel the frames in which it is background or foreground. The result of this step is shown in figure 8 (c). In this figure, the pixels which are selected as foreground are shown by white pixels while the background pixels are shown by black pixels.

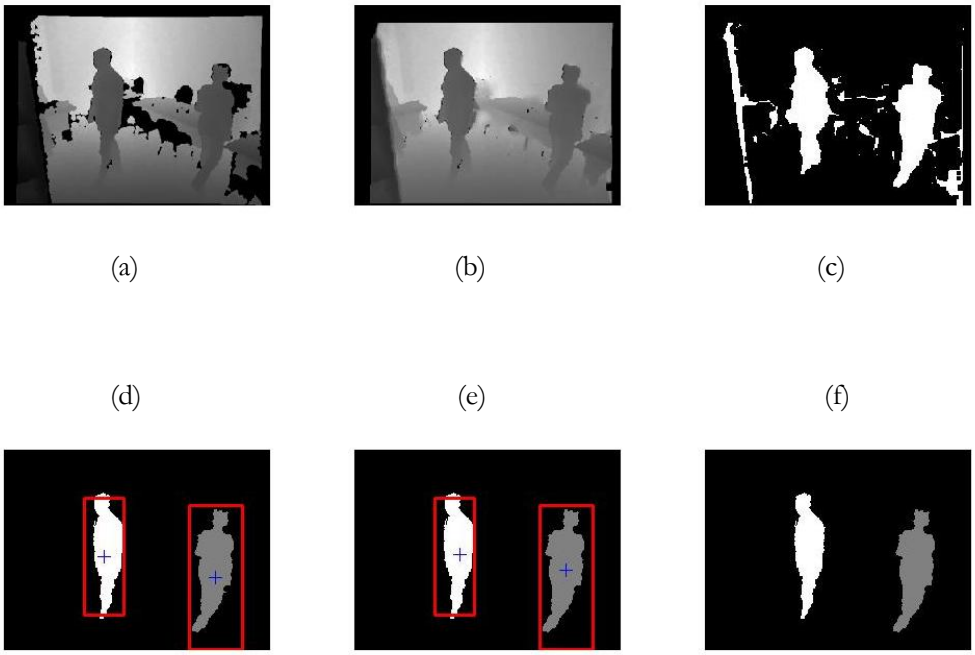


Figure 8. Shows sequential steps of image segmentation process. (a), raw depth image. (b), depth image after filling up the depth reading mistakes using SOR technique. (c), segmentation output, black pixels corresponding to the background vs. white pixels corresponding to the foreground. (d), labeling output from applying body detection algorithm on RGB images and extracting each human body GMM. (e), the segmented and labeled image from step (d) considering segmentation and labeling output from five previous frames. (f), the final segmented output.

To apply GMM on depth data will suffice our needs to perform segmentation, but in order to verify the segmentation results we also take advantages of the available RGB images. For each RGB frame we apply part based model (2.1.) to estimate if there is any human body in the scene, then in the detection location of each detected body we generate depth Gaussian Mixture Model and look for this model in the corresponding depth frame and keep a record of the match. This step is where we also label the foreground objects which are human bodies in our case. Different objects with different labels are shown by different color in the frame. In figure 8 (d) different objects are shown in different grey colors.

In every new frame, we look again for the detected people in its corresponding RGB image and also we look for it in the previous image models. To look into previous frames, has two advantages, first is that by looking into previous frames we can take advantages of the information the image model provides in previous frames and make correspondences between new frame and the previous frames. It helps to easily recover missing information in current frame if there happen any misreading, using the values the pixels gain in previous frames. The second advantage is the opportunity it provides to track and label each person. Figure 8 (e) is corresponding to the result of this step and figure 8 (f) is the final result of the segmentation procedure.

In this work, every person gain a label once they intrude the scene and by tracking the scene updates from frame to frame, we can decide when someone intrude the scene to assign them new label and when the existing person leaves the scene to re-assign their label. This procedure makes us able to obtain a segmented person with their corresponding label along the video which leads us toward our goal of human motion analysis.

3. TRAINING

Considering the object detection problem, simple models generally can perform better in practice compared to rich models. One reason is that rich models often suffer from difficulties in training. For object detection, rigid templates and bag-of-features models can be easily trained using discriminative methods such as Support Vector Machines (SVM). Discriminative training methods select model parameters so as to minimize the mistakes of a detection algorithm on a set of training images. Such approaches directly optimize the decision boundary between positive and negative examples.

Richer models are more difficult to train, in particular because they often make use of latent information. Consider the problem of training a part-based model from images labeled only with bounding boxes around the objects of interest. Since the part locations are not labeled, they must be treated as latent (hidden) variables during training. More complete labeling might support better training, but it can also result in inferior training if the labeling used suboptimal parts. Automatic part labeling has the potential to achieve better performance by automatically finding effective parts. More elaborate labeling is also time consuming and expensive.

Part-based deformable models are parameterized by the appearance of each part and a geometric model capturing spatial relationships among parts. For generative models, one can learn model parameters using maximum likelihood estimation. In a fully-supervised setting training images are labeled with part locations and models can often be learned using simple methods [9, 15]. In a weakly-supervised setting training, images may not specify locations of parts. In this case, one can simultaneously estimate part locations and learn model parameters with EM (Expectation-Maximization) algorithm [2, 18, 42].

To obtain high performance using discriminative training it is often important to use large training sets. In the case of object detection, the training problem is highly unbalanced

because there is vastly more background than objects. This motivates a process of searching through the background data to find a relatively small number of potential false positives, or hard negative examples. Data-mining methods can be made to converge to the optimal model defined in terms of the entire training set. Now we consider the problem of training models from images labeled with bounding boxes around objects of interest. This is the type of data available in the PASCAL datasets. Each dataset contains thousands of images and each image has annotations specifying a bounding box and a class label for each target object present in the image. Note that this is a weakly labeled setting since the bounding boxes do not specify component labels or part locations.

In this work, we not only consider the problem of person detection, but also we aim to find pose of each person in all of the detection. To do this we re-train the algorithm and this time we also take into account the 'pose' information available in PASCAL database in training phase. For any of the four basic pose a person could gain (Frontal, Rear, Right and Left), we train the algorithm and obtain specific model for each of them. We describe a procedure for initializing the structure of a mixture model and learning all parameters. Parameter learning is done by constructing a latent SVM training problem described in section 2.1.

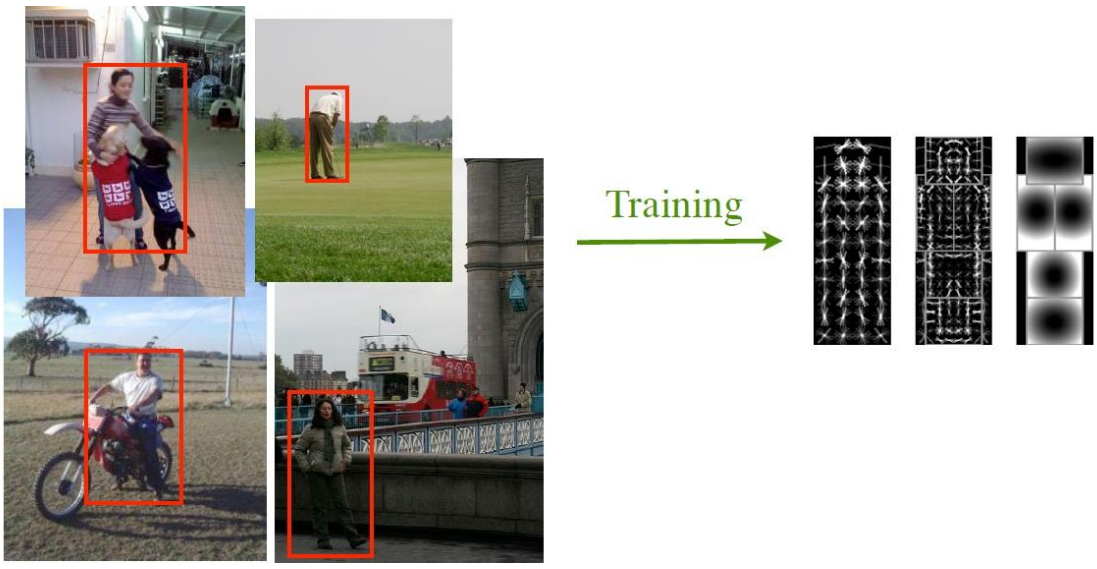


Figure 9. Weakly labeled data available in PASCAL data set. The object 'person' is annotated by a bounding box around them. Bounding boxes do not specify component labels or part locations.

3.1. LEARNING PARAMETERS

If C be an object class, let us assume the training examples for C are given by positive bounding boxes P and a set of background images N. P is a set of pairs (I, B) where I is an image and B is a bounding box for an object of class C in I. Let M be a (mixture) model with fixed structure. Recall that the parameters for a model are defined by a vector β. To learn β, a latent SVM training problem is defined with an implicitly defined training set D, with positive examples from P, and negative examples from N.

Each example $\langle x, y \rangle \in D$ has an associated image and feature pyramid $H(x)$. Latent values $z \in Z(x)$ specify an instantiation of M in the feature pyramid $H(x)$. Now define $\phi(x, z) = \psi(H(x), z)$. Then $\beta \cdot \phi(x, z)$ is exactly the score of the hypothesis z for M on $H(x)$. A positive bounding box $(I, B) \in P$ specifies that the object detector should 'fire' in a location defined by B. This means the overall score (7) of a root location defined by B should be high. For each $(I, B) \in P$ is defined a positive example x for the LSVM training problem. $Z(x)$ is defined so the detection window of a root filter specified by a hypothesis $z \in Z(x)$ overlaps with B by at least 50%. There are usually many root locations, including at different scales that define detection windows with 50% overlap. Treating the root location as a latent variable is helpful to compensate for noisy bounding box labels in P. A similar idea was used in [40].

Now consider a background image $I \in N$. We do not want the object detector to 'fire' in any location of the feature pyramid for I. This means the overall score (7) of every root location should be low. Let G be a dense set of locations in the feature pyramid. A different negative example x is defined for each location $(i, j, l) \in G$. $Z(x)$ is defined so the level of the root filter specified by $z \in Z(x)$ is l, and the center of its detection window is (i, j). Note that there are a very large number of negative examples obtained from each image. This is consistent with the requirement that a scanning window classifier should have low false positive rate.

The procedure Train is outlined below:

Data:

Positive examples $P = \{(I_1,B_1),\ldots, (I_n,B_n)\}$

Negative images $N = \{J_1, \ldots, J_m\}$

Initial model β

Result: New model β

```
1  F_n := 0;
2  for relabel := 1 to num-relabel do
3      F_p := 0;
4      for i := 1 to n do
5          Add detect-best (β,I_i,B_i) to F_p
6      end
7      for datamine := 1 to num-datamine do
8          for j := 1 to m do
9              if |Fn| ≥ memory-limit then break
10             Add detect-all (β,Jj , -(1 + δ)) to F_n
11         end
12         β := gradient-descent(F_p ∪ F_n)
13         Remove (i , v) with β . v < - (1 + δ) from F_n
14     end
15 end
```

The outermost loop implements a fixed number of iterations of coordinate descent on $L_D(\beta, Z_P)$. Lines 3-6 implement the *Relabel positives* step. The resulting feature vectors, one per positive example, are stored in F_P. Lines 7-14 implement the *Optimize beta* step. Since the number of negative examples implicitly defined by N is very large the LSVM data-mining algorithm is used. Data-mining iterated a fixed number of times rather than until convergence for practical reasons. At each iteration, hard negative examples are collected in

Fn, a new model using gradient descent trains, and then *Fn* shrinks by removing easy feature vectors. During data-mining, the cache grows by iterating over the images in *N* sequentially, until reaches a memory limit.

The function `detect-best` (β , *I* , *B*) finds the highest scoring object hypothesis with a root filter that significantly overlaps *B* in *I*. The function `detect-all` (β , *I* , *t*) computes the best object hypothesis for each root location and selects the ones that score above *t*. Both of these functions can be implemented using the matching procedure in Section 2.1.3. The function `gradient-descent` (*F*) trains β using feature vectors in the cache. The models are constrained to be symmetric along the vertical axis. Filters that are positioned along the center vertical axis of the model are constrained to be self-symmetric. Part filters that are off-center have a symmetric part on the other side of the model. This effectively reduces the number of parameters to be learned in half.

3.2. INITIALIZATION OF TRAINING PARAMETERS

The LSVM coordinate descent algorithm is susceptible to local minima and thus sensitive to initialization. This is a common limitation of other methods that use latent information as well. Initialization and training mixture models happens in three phases as follows.

❖ Phase 1. Initializing Root Filters:

For training a mixture model with m components the bounding boxes are sorted in P by their aspect ratio and split into m groups of equal size $P_1, \ldots, P_m$. Aspect ratio is used as a simple indicator of extreme intra-class variation. m different root filters $F_1, \ldots, F_m$ are trained, one for each group of positive bounding boxes.

To define the dimensions of F_i, the mean aspect ratio of the boxes in P_i and the largest area not larger than 80% of the boxes are selected. This ensures that for most pairs $(I, B) \in P_i$, F_i can place in the feature pyramid of I so it significantly overlaps with B. F_i is also trained using a standard SVM, with no latent information, as in [10]. For $(I, B) \in P_i$ the image region is wrapped under B so its feature map has the same dimensions as F_i. This leads to a positive example. Random sub-windows of appropriate dimension from images in N are selected to define negative examples. Figure 10 (a) and (b) show the result of this phase when training a two component car model.

❖ Phase 2. Merging Components:

The initial root filters are combined into a mixture model with no parts and the parameters of the combined model retrain using Train on the full (un-split and without warping) data sets P and N. In this case, the component label and root location are the only latent variables for each example. The coordinate descent training algorithm can be thought of as a discriminative clustering method that alternates between assigning cluster (mixture) labels for each positive example and estimating cluster 'means' (root filters).

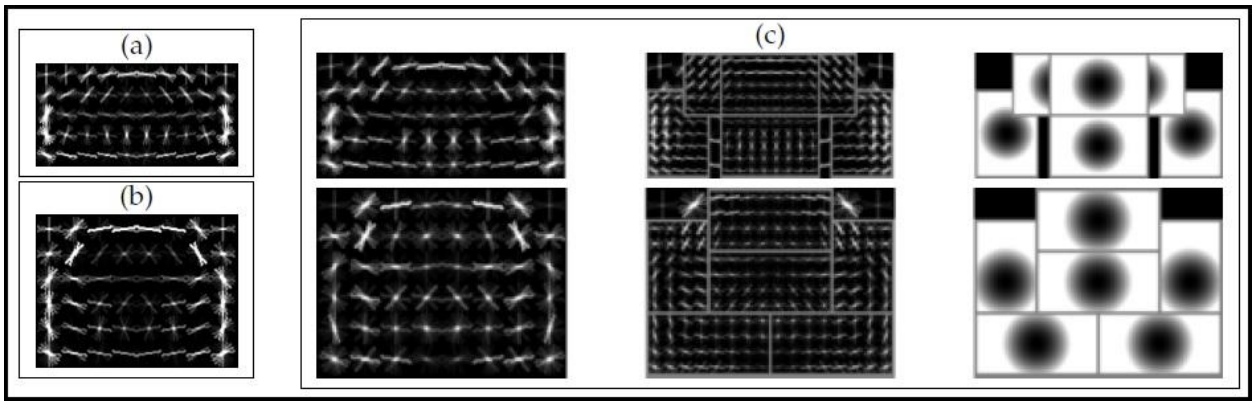


Figure 10. (a) and (b) are the initial root filters for a car model (the result of Phase 1 of the initialization process). (c) is the initial part-based model for a car (the result of Phase 3 of the initialization process).

❖ Phase 3. Initializing Part Filters:

The parts of each component are initialized using a simple heuristic. The number of parts is fixed at six per component, and using a small pool of rectangular part shapes. Parts are greedily placed to cover high-energy regions of the root filter (The 'energy' of a region is defined by the norm of the positive weights in a sub-window). A part is either anchored along the central vertical axis of the root filter, or it is off-center and has a symmetric part on the other side of the root filter. Once a part is placed, the energy of the covered portion of the root filter is set to zero, and we look for the next highest-energy region, until six parts are chosen.

The part filters are initialized by interpolating the root filter to twice the spatial resolution. The deformation parameters for each part are initialized to $d_i = (0, 0, .1, .1)$. This pushes part locations to be fairly close to their anchor position. Figure 10 (c) shows the results of this phase when training a two component car model. The resulting model serves as the initial model for the last round of parameter learning.

3.3. TRAINING APPLICATION

One of our goals in this work is to obtain strong and realistic results in human body pose estimation. To achieve this goal, our approach is to concentrate on estimating the upper bodies pose estimation, which is more relevant for indoor scene since in this type of the scenes, the lower body is often occluded and the upper body conveys most of a person's gestures.

To augment detection models to be capable of estimate upper-body poses, we are obligated to adapt the problem of model training with this new purpose. First step to achieve this goal is to provide an appropriate dataset to train the deformable part-based algorithm based on them. As far as we are interested in estimating the bodies pose just by considering each person's upper body, we are required to provide a new dataset with new 'bounding box' information for every image in PASCAL2010 dataset. The new bounding boxes exclusively cover people's upper body in training dataset as positive samples. The lower body of every person object is considering as background or negative examples.

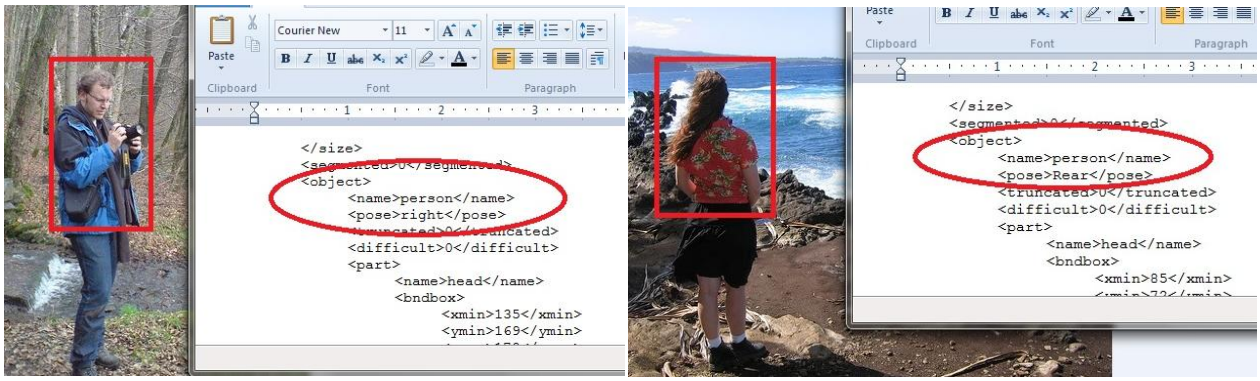


Figure 11. Shows two images from PASCAL2010 data set with their corresponding *.xml information file. Every person in this data set, get specific information. All of the information of an image is accumulated in a xml file which accompanies the image. This information contains specific characteristic of the image such as the image coordinate, object class, object coordinate in the image, object pose and etc.

In PASCAL2010 dataset for every person is defined one specific pose among five poses: Frontal, Rear, Left, Right and Unspecified. Regarding to the camera, every person gets one of these labels in their 'pose' tag in their corresponding .xml file. We use these data to create a new class for each body pose and train a separate model distinctively for each body pose. By considering the 'pose' in training procedure, whole the training volume is consumed to train different variation in appearance distinctly for one specific pose at the time and it leads to train and produce a robust detection model for each pose separately. Figure 12 shows the acquired model for the Right pose vice the Left pose of the object Person-Upper-Body.

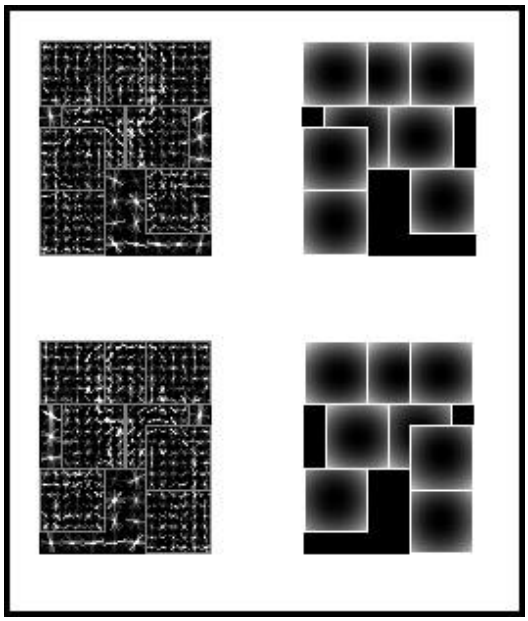


Figure 12. Corresponding models to detect people in the Right position vs. the Left position

4. VALIDATION

Validation of results is an important phase in this work to ensure us whether the methodology that we employed is satisfying our goal. It also provides us results in a timely way, to assess and evaluate whether we need to modify some specific stage in the methodology (i.e. change in dataset, methodology, etc.). In this way, validation allows for comparison of alternative parameters and methodologies. There are several validation procedures that could be used throughout the development of a methodology to be used for selection of parameters. The task we used to validate our study is to obtain pose detection accuracy by calculating precision and recall of our methodology's output.

Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. In even simpler terms, high recall means that an algorithm returned most of the relevant results. High precision means that an algorithm returned more relevant results than irrelevant.

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

	Actual Class (Observation)	
Predicted class (expectation)	**TP** (True Positive) Correct Result	**FP** (False Positive) Unexpected Result
	FN (False Negative) Missing Result	**TN** (True Negative) Correct Absence of Result

Table 1. Precision – Recall evaluation table

In a classification task, the precision for a class is the number of 'true positives' (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of 'true positives' and 'false positives', which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of 'true positives' divided by the total number of elements that actually belong to the positive class (i.e. the sum of 'true positives' and 'false negatives', which are items which were not labeled as belonging to the positive class but should have been). Often, there is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other.

We validate our method by measuring the number of True Positive, True Negative, False Positive and False Negative observations once we apply a pose detector on the video frames. In the end, we measure the F-1 score of the classifier which is a measure of the test's accuracy and considers both the precision and the recall of the test to compute the final score. An optimal classifier system would be the one that could predict all the human's body poses in given images correctly. We could interpret it as a weighted average of the precision and recall, where the best F-1 score has its value at 1 and worst score at the value 0.

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

It is considered a single value obtained combining both the precision and recall measures and indicates an overall utility of the system. In the following, we provide some information about the characteristic of the captured videos by us and then we explain the procedure of evaluating the methodology that we used.

4.1. DATA

We evaluate our methodology on our home-made videos acquired using Kinect camera. The goal of evaluating our system is to predict the overall precision, recall and F-1 score of the system when we aim to predict the pose of each person individually throughout the video.

We define the result as 'true positive' when the filter is able to detect the pose of the person correctly, as 'false positive' when the filter detects the pose of the person but the detection is not correct (i.e. detect pose of a person as being in its right side while they are in their left side), as 'true negative' when the filter cannot predict the pose of the person because the person is mostly (experimentally more than ~50%) occluded or has taken strange pose which is almost impossible to detect, and as 'false negative' when the filter cannot detect the pose of the person, in a situation it is supposed to be able to find the pose correctly.

Each of the acquired videos contains hundreds of frames with varying number of people who enter the scene and after a while physical interaction with the other people they leave the scene. It is necessary to keep the scene empty of people when we start capturing the movie, to augment the segmentation algorithm proficiency. This aids the segmentation algorithm to recognize the background before intruding any objects into the scene. All the videos are recorded in an office, where the Kinect camera is located at height of about 2.5 meters above the floor and the camera is rotated towards the floor. Table 2 summarizes number of frames and people in each video:

	# Frames	# People
Video #1	625	2
Video #2	126	1
Video #3	1239	2
Video #4	557	3

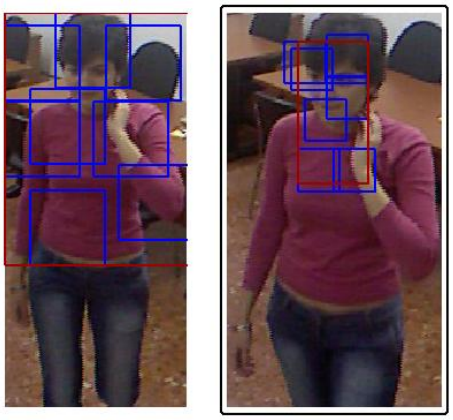
Table 2. Characteristic of the acquired videos

4.2. EVALUATION

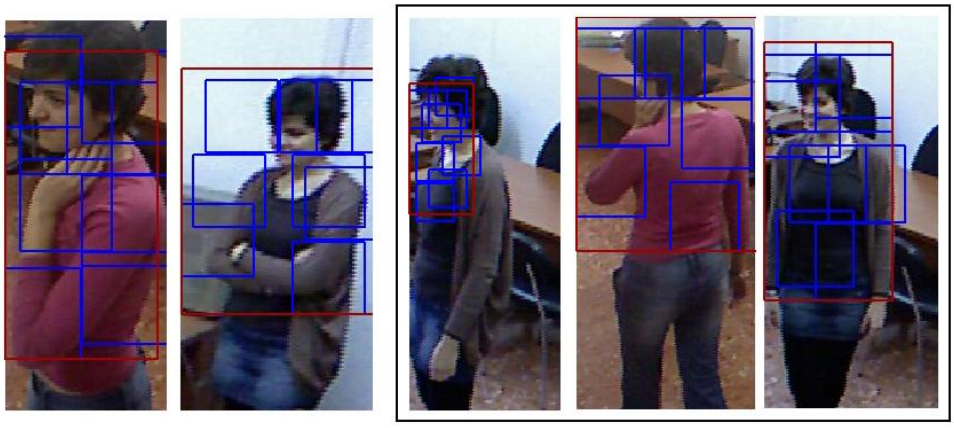
We evaluated our system using the videos we have acquired for this purpose and we emphasize that the test set that we consider to use is a difficult test bed for object detection. This difficulty is mostly because of the quality of the images from the Kinect in an indoor office with regular lighting system and specific furniture inside an office which cause an over-crowded background and makes task of background subtraction slightly difficult.

At test time, the goal is to predict the bounding boxes of all people's body of a specific class (pose) in an image (if any). In practice, a system will output a set of bounding boxes with corresponding scores, and we can threshold these scores at different points to obtain a precision-recall curve across all images in the test set. For a particular threshold the precision is the fraction of the reported bounding boxes that are correct detections, while the recall is the fraction of the objects found. A predicted bounding box is considered correct if it overlaps more than 50% with the ground-truth bounding box, otherwise the detection is considered a false positive detection. Multiple detections are penalized. If a system predicts several bounding boxes that overlap with a single ground-truth bounding box, only one prediction is considered correct, the other are considered false positives.

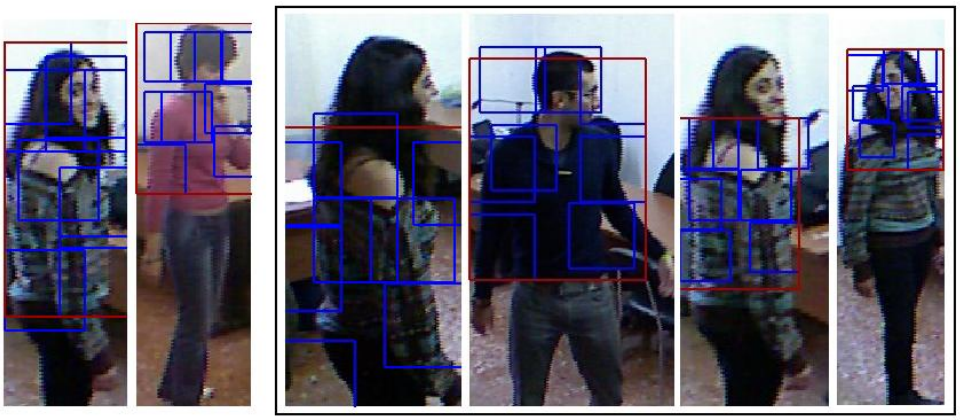
We trained a two component model for each class (pose). Figure 13 shows some detection we obtained using those models. We show both high-scoring correct detection and high-scoring false positives. For all the experiments shown here we use the objects not marked as 'difficult' from the PASCAL 2010 'trainval' dataset to train the models. The system is fairly efficient. Using a desktop computer it takes about 6 hours to train the model on PACAL2010. The system makes use of the multiple-core architecture for computing filter responses in parallel, although the rest of the computation runs in a single thread:



(a)



(b)



(c)

Figure 13. True Positive detection results against False Positives. False positives are shown inside a black border. (a) Detection results for Frontal pose, (b) Left pose and (c) Right pose.

Table 3 and 4 summarize results of different models on each pose category:

	TP	FP	TN	FN	Precision	Recall	Accuracy
Video #1	244	15	227	124	**94%**	**66%**	**78%**
Video #2	43	8	67	1	**84%**	**97%**	**90%**
Video #3	438	274	930	102	**62%**	**80%**	**70%**
Video #4	300	12	402	150	**96%**	**66%**	**78%**

Table3. Left-Right detection results on acquired videos

	TP	FP	TN	FN	Precision	Recall	Accuracy
Video #1	231	23	118	116	**90%**	**66%**	**81%**
Video #2	64	10	39	5	**86%**	**92%**	**89%**
Video #3	964	208	484	90	**69%**	**83%**	**75%**
Video #4	374	10	398	82	**97%**	**76%**	**87%**

Table4. Frontal-Rear detection results on acquired videos

Figure 14 also plot the precision-recall graph:

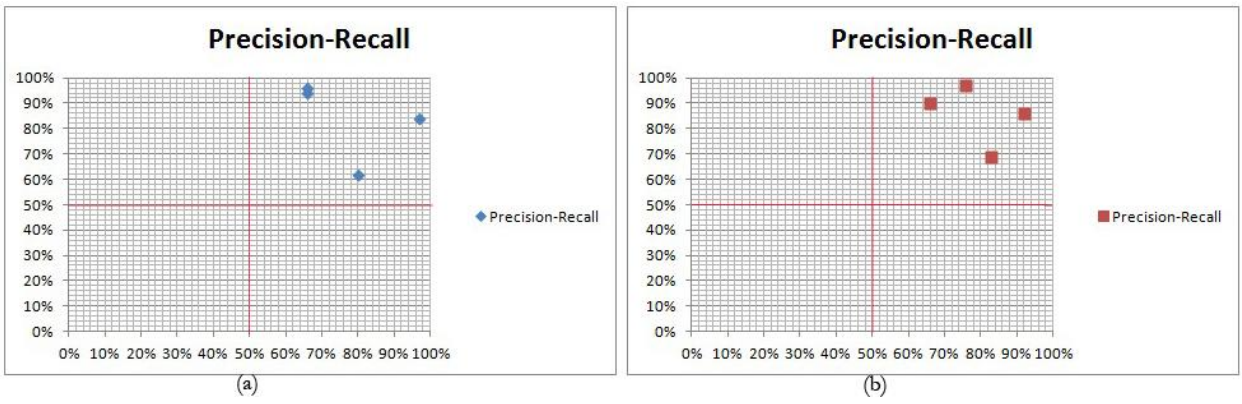


Figure 14. Precision- recall scatter plot. (a) shows the scatter plot for left-right pose classifier and (b) shows the scatter plot for frontal-rear pose classifier

For each list of videos, there is a point in the plot that corresponds to the precision and recall result values of specific classifier on that video. These plots are extremely useful to compare different approaches for a particular data set. If we think of 'precision' as percentage of positive prediction which are correct, and 'recall' as percentage of positive cases are caught, we aim to catch high percentage positive cases which are correct. Based on this, a good precision-recall output must results points which are located in the upper-right corner of the graph because we want to have high precision and high recall. Looking at the figure we can conclude that both of our classifiers perform efficiently in our difficult test set, since all the points in both classifiers are accumulated in the upper-right side of the graph.

Beside precision-recall graph, F-measure can be also a meaningful performance measure since it also combines the both recall and precision into a global measure. As we mentioned before, the best classifiers are those which provide F-score close to one. As we can see in table 3 and table 4, the accuracy results are so close to 1.0, without high variability among the test results over different videos. It demonstrates our methodology is reliable and the detection error has a low rate, although as experiment shows, we still can augment its output accuracy by capturing video frames with higher resolution and also by maximizing the frame rate in producing a video. To maximize the frame rate, leads to avoid losing of information between sequential frames of the video. To use vacant spaces as background rather than crowded backgrounds also improves the detection results. Generally the models we introduced to predict the pose, are contrast sensitive. Contrast sensitivity is the ability to perceive the difference in brightness between a foreground color and a background color. This ability of being distinguished is increasing when there is a high contrast between the background and the foreground object. It is also highly related to the size, distance and illumination of the object to be detected. Maximum contrast occurs with white on black or vice versa. Higher contrast levels result in a greater likelihood of detection by the model.

5. CONCLUSION AND FUTURE LINES

In this work we proposed a methodology for human body pose estimation and their motion analysis in the video scenarios. To do this we trained a classifier on modified version of PASCAL2010 dataset. Modification applied to confine the positive examples that algorithm uses to train the model, to person's upper-body. We also create separate class of object for each of four specific poses every person can get (Frontal, Rear, Left, Right). After a successful training we gain four different classifiers which are able to estimate bodies pose in an image. We concentrate on upper-body since we believe that in indoor scenes, lower body are mostly occluded and also in general, upper body plays the major role in pose detection based on its variation under different positions.

The classifier itself is originally an object detection system based on mixtures of multi-scale deformable part models. This system relies heavily on new methods for discriminative training of classifiers that make use of latent information. It also relies heavily on efficient methods for matching deformable models to images.

We also made use of GMM based segmentation methods to extract the background from the scene and make the task of pose estimation faster and more accurate. Segmentation also made us able to discriminate every person from the others in a video and therefore we could assign specific label to each person entire the video. A decent labeling is an essential task to track objects during the video.

The resulting system is both efficient and accurate, leading to state-of-the-art results on our home-made datasets. Our models are already capable of discriminating among different poses, but we would like to move towards richer models. This can be done by using wider dataset and more accurate labeling of them. On the other side, in fact in this work we are not able to distinguish frontal pose of a person from rear pose of them and both of these poses are considered as one pose. In the future we would like to include head

or face detectors into our framework to discriminate between frontal and rear pose of each person based on the result of these detectors. To predict the people's movement trajectory and plan a motion path for each of them is another approach that we consider to follow in the future. This model is often used in robot navigation to avoid obstacles or approach a target, but here, it could be used to predict near-future person movement for the next seconds which is essential in task of human behavior analysis and prediction.

REFRENCES

[1] Y. Amit and A. Kong, "Graphical templates for model registration," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 3, pp. 225–236, 1996.

[2] Y. Amit and A. Trouve, "POP: Patchwork of parts models for object recognition," International Journal of Computer Vision, vol. 75, no. 2, pp. 267–282, 2007.

[3] Deutscher, J., Davison, A., Reid, I.: Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. CVPR 2001, vol. 2, pp. 669-676.

[4] R. Fablet and M. J. Black. Automatic detection and tracking of human motion with a view-based representation. *ECCV*, 2002.

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. CVPR, vol 2:pp. 2066–2074, 2000.

[6] M. Burl, M. Weber, and P. Perona, "A probabilistic approach to object recognition using local photometry and global geometry," in European Conference on Computer Vision, 1998.

[7] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models,"
IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 6, pp. 681–685, 2001.

[8] G. Hua, M.-H. Yang, and Y. Wu. Learning to estimate human pose with data driven belief propagation. *CVPR*, vol 2:pp. 747–754, 2005.

[9] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, "Spatial priors for part-based recognition using statistical models," in IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[11] M. W. Lee and I. Cohen. Proposal driven MCMC for estimating human body pose in static images. *CVPR*, 2004.

[12] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. *CVPR*, vol 2:pp. 326–333, 2004.

[13] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision*, vol 43, no. 1:pp. 45–68, 2001.

[14] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. ECCV, vol 4:pp. 700–714, 2002.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman,, "Pictorial structures for object recognition," International Journal of Computer Vision, vol. 61, no. 1, 2005.

[16] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. *CVPR*, vol 2:pp. 882–888, 2004.

[17] A. Elgammal and C. S. Lee. Inferring 3d body pose from silhouettes using activity manifold learning. *CVPR*, 2004.

[18] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in IEEE Conference on Computer Vision and Pattern Recognition, 2003.

[19] R. Fergus, P. Perona, and A. Zisserman, "A sparse object category model for efficient learning and exhaustive recognition," in IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[20] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," IEEE Transactions on Computer, vol. 22, no. 1, 1973.

[21] R. Rosales and S. Sclaroff. Learning body pose via specialized maps. *In NIPS*, 2002.

[22] C. Smichisescu. Learning to reconstruct 3d human motion from bayesian mixture of experts. *CVPR*, 2005.

[23] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[24] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. *Computer Vision and Image Understanding*, 1999.

[26] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in IEEE Conference on Computer Vision and Pattern Recognition, 2004.

[28] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection
with interleaved categorization and segmentation," International Journal of Computer Vision, vol. 77, no. 1, pp. 259–289, 2008.

[29] D. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, November 2004.

[30] C. Papageorgiou, M. Oren, and T. Poggio, "A general frameworkfor object detection," in IEEE International Conference on Computer Vision, 1998.

[31] W. Plantinga and C. Dyer, "An algorithm for constructing the aspect graph," in Foundations of Computer Science, 1985., 27th Annual Symposium on, 1986, pp. 123–131.

[34] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3d structure with a statistical image-based shape model. *ICCV*, 2003.

[35] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *ICCV*, pages pp. 734–741, 2003.

[36] H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," in IEEE Conference on Computer Vision and Pattern Recognition, 2000.

[37] J. Coughlan, A. Yuille, C. English, and D. Snow, "Efficient deformable template detection and localization without user initialization," Computer Vision and Image Understanding, vol. 78, no. 3, pp. 303–319, June 2000.

[38] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," Massachussets Institute of Technology, Tech. Rep. A.I. Memo No. 1521, 1994.

[40] P. Viola, J. Platt, and C. Zhang, "Multiple instance boosting for object detection," in Advances in Neural Information Processing Systems, 2005.

[41] P. Viola and M. Jones, "Robust real-time face detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, May 2004.

[42] M. Weber, M. Welling, and P. Perona, "Towards automatic discovery of object categories," in IEEE Conference on Computer Vision and Pattern Recognition, 2000.

[43] A. Yuille, P. Hallinan, and D. Cohen, "Feature extraction from faces using deformable templates," International Journal of Computer Vision, vol. 8, no. 2, pp. 99–111, 1992.

[44] E. Bernstein and Y. Amit, "Part-based statistical models for object classification and detection," in IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[45] X. Zhu and D. Ramanan, "Face Detection, Pose Estimation, and Landmark Localization in the Wild," Computer Vision and Pattern Recognition (CVPR) Providence, Rhode Island, June 2012.

[46] P. F.Felzenszwalb, and R. B. Girshick, and D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models" in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.32, no. 9, pp. 1627-1645, 2010

[47] A. Bissacco, M. H. Yang, and S. Soatto, "Fast Human Pose Estimation using Appearance and Motion via Multi-Dimensional Boosting Regression" in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[48] Z.Zivkovic, "Improved adaptive Gausian mixture model for background subtraction", International Conference Pattern Recognition, UK, August, 2004.

[49] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/

[50] http://www.wikipedia.org/