2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Fast HOG based Person Detection devoted to a Mobile Robot with a Spherical Camera

A. A. Mekonnen[1], C. Briand[1], F. Lerasle[1], A. Herbulot[1]

***Abstract*— In this paper, we present a fast Histogram of Oriented Gradients (HOG) based person detector. The detector adopts a cascade of rejectors framework by selecting discriminant features via a new proposed feature selection framework based on Binary Integer Programming. The mathematical programming explicitly formulates an optimization problem to select discriminant features taking detection performance and computation time into account. The learning of the cascade classifier and its detection capability are validated using a proprietary dataset acquired using the *Ladybug2* spherical camera and the public INRIA person detection dataset. The final detector achieves a comparable detection performance as Dalal and Triggs [2] detector while achieving on average more than 2.5x - 8x speed up depending on the training dataset.**

I. Introduction

For decades it has been demonstrated that the autonomy of an autonomous mobile robot highly depends on its environment perception capabilities. For example, if one considers autonomous robot navigation, the success depends on the robot's ability to perceive its surrounding well and its ability to distinguish obstacles from free paths. With such consideration, an omnidirectional camera is the quintessential sensor. An omnidirectional camera usually provides a 360^o Field Of View (FOV) in the horizontal direction and sometimes even cover more than 120^o in the vertical plane, pretty much the essential surrounding. As a consequence, they are gaining much appreciation and use in robotic applications, including but not limited to: robot localization, mapping, ground robot navigation, etc., [15]. One such application is detection of people in the vicinity of a mobile robot be it for active interaction or social considerations during navigation in crowded environments. With a complete horizontal FOV, the robot is appraised of any activity in its complete surrounding which allows it to be better reactive and considerate [9], [22].

(a) (b)

Fig. 1: Ladybug2 camera and a stitched, and unwrapped image.

Contrary to their amenities, omnidirectional cameras are not trivial to use. The actual technique used to cover wide FOV governs the added difficulty. Currently, there are three prevalent types of omnidirectional cameras: dioptric, manage wide angle coverage via combination of shaped lenses; catadioptric, those that combine a classical camera with a shaped mirror; and polydioptric, the kind which use multiple cameras–with overlapping fields of view–oriented in various directions. Both dioptric and catadioptric cameras suffer from pronounced geometric distortions, significantly non-uniform resolutions, and high sensitivity to illumination changes. On the other hand, polydioptric cameras provide real omnidirectional view without pronounced geometric, resolution, and/or illumination artifacts. But, as a result of their make, they result in a high resolution image that demands high computational resources for processing. The *Ladybug2* is one such kind of camera manufactured by Point Grey Inc [11]. The *Ladybug2* (fig. 1a) is a spherical omnidirectional camera system that contains six cameras mounted in such a way to view more than 75% of the full sphere. Each camera has a maximum resolution of 1024x768 pixels resulting in a 3500x1750 pixels stitched high resolution panoramic image (fig. 1b). The camera system has an IEEE-1394b (FireWire 800) interface that allows streaming at 30 fps with the drivers provided by the manufacturer [11].

In this work, we are interested in developing a person detection system to detect people around a mobile robot using a *Ladybug2* camera. As stated previously, this camera does not suffer from the severe geometric/illumination artifacts as the other omnidirectional camera families. Clearly, the processing power stipulated by the high resolution images is a major bottleneck that makes classical person detection approaches infeasible. Any application that intends to use these cameras has to take this into consideration. In this paper, we propose and implement an automated person detection system that not only tries to optimize over detection performance, but also optimizes over computation time required by the detector. We build upon the original Histogram of Oriented Gradients (HOG) features proposed by Dalal and Triggs [2], features that have proven useful for almost a decade and are still used by some of the state-of-the-art person detectors [3] though at the expense of CPU resources. We formulate a feature selection problem optimized via Binary Integer Programming (BIP) [13] taking detection performance and computation time into consideration to implement a person detector that has comparable detection performance to the original detector proposed by Dalal and Triggs and yet on average is more than 8 times faster on the *Ladybug2* images.

This paper is organized as follows: section II discusses related works briefly, section III presents the overview of our framework followed by feature pools and details of our

[1] CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France Email: {aamekonn, cyril.briand, frederic.lerasle, ariane.herbulot}@laas.fr

improved classifier learning in sections IV and V respectively. Experiments and results are presented in section VI and finally, the paper concludes with concluding remarks in section VII.

II. Related Works

To date, various perspective camera based person detectors have been proposed (see comprehensive surveys in [3], [7]) When considering a camera on a moving vehicle, as in a mobile robot, the detector has to rely on information per frame and can not rely on stationary or slowly changing background assumptions/models. In this vein, the first major successful breakthrough was the work of Dalal and Triggs [2] which introduced and used HOG features with a linear SVM classifier. To date, HOG is the most discriminative feature and no other single feature has been able to supersede it [3]. It has also been successfully used for person detection in 3D (RGB + D) data [14].

The main downside of HOG based detectors is the associated computation time. These features are extracted first by computing the gradient, then by constructing a histogram weighted by the gradient in an atomic region called a cell. Histograms of neighboring cells are grouped into a single block, cross-normalized and concatenated to give a feature vector per block. The final extracted feature within a given detection window is the concatenation of the feature vectors of constituent blocks (in one instance this amounted to a 3780 dimensional vector in [2]). For an arbitrary given image frame, person detection proceeds by testing all possible locations (position and scale), a.k.a sliding window approach, with this high dimensional vector which indeed reduces the speed significantly. To improve this, Zhu *et al.* [21], reformulated the problem as a feature selection procedure over HOG block size using AdaBoost in an attentional cascade structure. The cascade structure, pioneered by Viola and Jones [17], spreads the detection process into various nodes that reject a majority of negative windows, allowing only positive windows to progress through the entire cascade. This speeds up detection drastically. Another alternative is to parallelize the detection process over multiple processors [12], but, this necessitates the use of specialized Graphical Processing Unit (GPU).

Some works have managed to go beyond Dalal and Triggs detector. But, they had to either combine HOG with multiple other features (*e.g.* with Local Binary Patterns [18], with edgelets and covariance descriptors [20]) or consider a part based approach (*e.g.* [6]). Combining HOG with other features has showed advantages over detection performance as well as speed. Part based approaches, on the other hand, try to infer the presence of different parts of a person's body and aggregate the confidence to detect a person. Comparatively, these kind of approaches lead to improved results primarily because they can handle multiple poses and partial occlusions. But consequentially, they incur increased computation time.

When dealing with panoramic images from omnidirectional cameras in autonomous robots with limited embedded CPU resources it is impossible to directly use HOG based detectors. One has to resort to cheap features at the cost of reduced detection performance or their fusion with other sensors, for *e.g.* with Laser Range Finders (LRFs) [22]. Another possibility could be to constrain the region of interest within the images using hypothesis generated from other fast modes, *e.g.* from LRF [10].

In this work we present a person detector with a cascade configuration similar to Viola and Jones [17]. Each node of the cascade considers the original HOG features tweaked to be suited for feature selection (discussed in section IV), BIP for actual feature selection, and AdaBoost for feature weighting and classification. Contrary to most feature selection techniques that rely on boosting techniques where important features are selected taking the error rate into consideration, we use BIP to select discriminant features that have the least combined computation time and yet fulfill the False Positive Rate (FPR) and True Positive Rate (TPR) requirements of the node. To the best of our knowledge this is new in the literature. This paper claims three main contributions:

1) We develop and present a mathematical formulation based on BIP for feature selection taking both computation time and detection performance into consideration.
2) We present implementation details of a detector based on the above formulation.
3) We present a thorough and comparative evaluation of the proposed detector with Dalal and Triggs HOG detector on a proprietary dataset collected with *Ladybug2* camera and on the INRIA public person dataset.

Even though this work is presented with emphasis on a spherical camera, it is equivalently applicable to images from classical cameras as demonstrated with validation on a public dataset.

III. Framework Overview

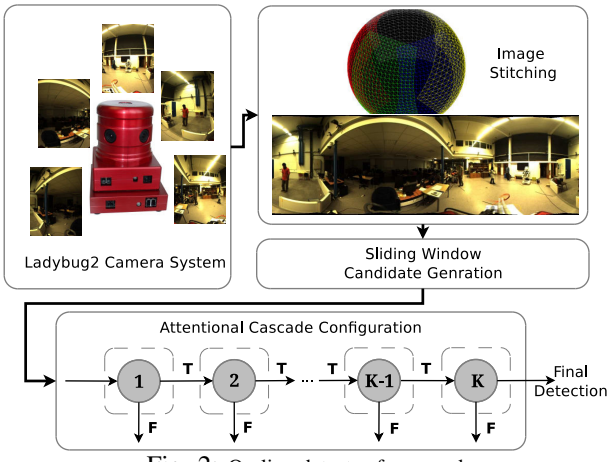

Fig. 2: On-line detector framework.

In this work, we adopt the attentional cascade detector configuration pioneered by Viola and Jones [16]. Each node rejects negative windows and passes along potential positive windows onto the next stage. Only those classified as true

detection by all nodes are considered as true targets. This structure has gained wide acceptance and has even been applied in recent part-based approaches [5].

Fig. 2 depicts the final detector applied on live image streams. The different images from the *Ladybug2* cameras are first projected onto a spherical calibrated mesh [11]. They are then blended along their overlapping fields of view to form a stitched sphere. This sphere is finally unwrapped to form a panoramic view. Candidate windows are then generated via a sliding window approach and fed to the attentional cascade classifier. At each stage of the cascade k, $k \in \{1, 2, ..., K\}$, a significant proportion of the negative samples are rejected and only those that make it till the end are considered as true detections.

A key issue in cascaded detector configuration is how to select which features to use in each node. Classically, many authors have resorted to AdaBoost, one variant of the boosted classifiers family, for feature selection and combination *e.g.* [16], [21]. But, this approach, which is quite suitable when dealing with homogeneous features with the same computational time, selects features solely based on their detection performance. When considering features with varying computation time, it is wise to take this factor into consideration. To address this, we propose a novel feature selection and classifier learning scheme illustrated in fig. 3 and detailed in section V.

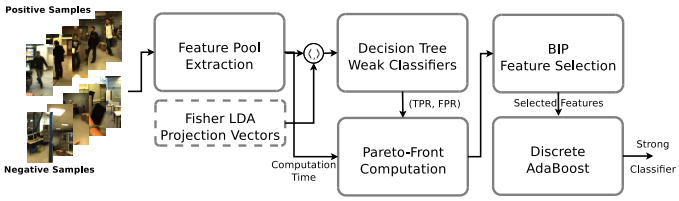

Fig. 3: Proposed feature selection and classifier learning scheme.

Given positive and negative training samples, Fisher's Linear Discriminant Analysis (Fisher LDA) is used to obtain a projection hyperplane that would maximize the inter-class scatter while minimizing the intra-class scatter. For each feature in the feature pool (discussed in section IV) a decision tree is learned which results in a specific TPR and FPR on a validation set. Next, taking these two criteria as well as computation time, Pareto-Front analysis [1] is used to reduce the number of features considered. This step is employed to decrease the number of features to a size manageable by the BIP module. Using this reduced feature set, an optimization problem is formulated via BIP to select relevant features with the smallest computation time, that fulfill the TPR and FPR requirements of the node.

BIP is a special case of integer programming where decision variables are required to be 0 or 1 (rather than arbitrary integers). It aims at minimizing a given linear objective function $f = c.x$ subject to the constraints that $A.x \geq b$, where x represents the vector of 0-1 variables (to be determined), c and b are known coefficient vectors, A is a matrix of coefficients (called constraint matrix). It is well-known that BIP is $\mathcal{NP}$-hard in the strong sense but, in practice, branch-and-cut techniques are able to solve huge binary integer program very fastly [13], [19]. Finally, discrete AdaBoost takes the features selected by the BIP module and builds a strong classifier by weighting and combining them.

IV. Feature Pool

Description: As it has been mentioned, no other single feature has been able to supersede HOG feature [3]. Hence, naturally, we have resorted to use it. HOG features are extracted first by computing the gradient, then by constructing a histogram weighted by the gradient in an atomic region called a cell. Histograms of neighboring cells are grouped into a single block, cross-normalized and concatenated to give a feature vector per block. The final extracted feature within a given detection window is the concatenation of the vectors from each feature block (for a detailed explanation please refer to [2]).

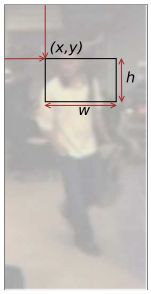

Fig. 4: Feature parameterized by (x, y, w, h).

In this work, we use the original HOG features proposed by Dalal and Triggs [2] along with their widely preferred/used computation, *i.e.* a cell size of 8x8 pixels, a feature block size of 2x2 cells and an 8 pixel horizontal and vertical stride. For a given 64x128 image window, this results in a 7x15 feature block layout (each feature block is a 36 dimensional vector). Now to get a pool of features, lets introduce an operator Ω that takes a starting location (x,y), width (w), and height (h), and concatenates all feature blocks within this region. Hence, for a specific input, the operator $\Omega(x, y, w, h)$ returns a concatenated feature which makes one component of our feature pool (fig. 4). Using all possible values of $x, y, w,$ and h in a given image region made of HOG feature blocks furnishes the considered feature pool, $\mathcal{F}$, represented as a set in eq. 1. With the 7x15 feature block considered in the work, this results in a total of 1792 features that make up our feature pool.

$$\mathcal{F} = \{\Omega(x, y, w, h) : 0 \leq x < 7; 0 \leq y < 15; \\ 1 \leq w \leq (7 - x); 1 \leq h \leq (15 - y)\} \quad (1)$$

In summary, in the works of Dalal and Triggs, all resulting feature blocks extracted from the 64x128 image window are concatenated, giving a single high dimensional vector–with exactly 7x15x36 dimensions–as a final feature. Whereas, in our case, we end up with a pool of features with dimensions ranging from 36 (smallest) to 7x15x36 (highest).

Computation Time: The features in our feature pool are of varying dimensions. Incidentally, the associated time taken to extract them varies. Since the smallest building unit is a single HOG feature block, determining the computation time of each feature obtained using the above defined Ω operator is straight forward. Each feature obtained using Ω contains an integral multiple of individual HOG blocks. If it takes τ milliseconds to compute the feature vector of a single block,

then it takes $n.\tau$ milliseconds for a feature made up of n blocks using the Ω operator. With this, the computations time for the different features in the pool varies from the smallest, τ, to the highest, $105.\tau$ milliseconds.

V. Classifier Learning with Computation Time Consideration

In the adopted cascade configuration, each node of the cascade is influenced by the implementation choice of weak learners, the weak classifiers that are trained on each distinct feature of the feature pool; feature selection algorithm, that chooses a subset of the features taking selected performance criteria into consideration; feature weighting and combining algorithm; and data mining techniques that try to robustify the classification performance of each node.

A. Weak Learners

These are each of the weak classifiers that are trained on each distinct feature of the feature pool, F. Each unique weak classifier is associated with and trained on a unique single feature. Recall that, we have chosen to use Fisher LDA to determine a projection hyperplane to project the multi-dimensional feature vectors to obtain a scalar value. Then, a decision tree is learned (equivalent to having multiple thresholds), per feature, to provide a binary classification output. Fisher LDA is preferred over complex classifiers like an SVM because of its comparatively short training duration. Given the large amount of features in the considered feature pool, employing SVM would lead to an overwhelming training period. In addition, Fisher LDA leads to a weak learner that is easy to integrate with boosting methods. Once a weak learner is trained on a given training set, it is characterized by three performance indicator parameters: its True Positive Rate (TPR), False Positive Rate (FPR) and computation time (τ_j; $j \in \{0, 1, ..., 1791\}$). Fisher LDA is implemented using the *alglib* C++ mathematical library[1].

B. Pareto Front Analysis

Recall that the total number of weak learners or features considered is 1792. As it will come evident in section V-C this amount of features is too much for a tractable optimization. Hence, the number of feature must be pre-reduced. To do this, Pareto Front Analysis is used to extract the dominant features–based on their TPR, FPR, and computation time. A simple algorithm outlined in [1] is used to select the dominant features that maximize TPR, and minimize both FPR and computation time. Fig. 5 shows an exemplary instance of extracted front amongst the whole depicted feature pool. The exact number of features extracted depends on their properties (TPR, FPR, and τ_j), but in our experiments the retained features never exceeded 200.

C. Binary Integer Programming

The BIP based feature selection makes the core of this work's contribution. Provided the BIP is handed a few number of weak learners or features F^*, such that $F^* \subseteq F$,

[1]ALGLIB Project – http://www.alglib.net/

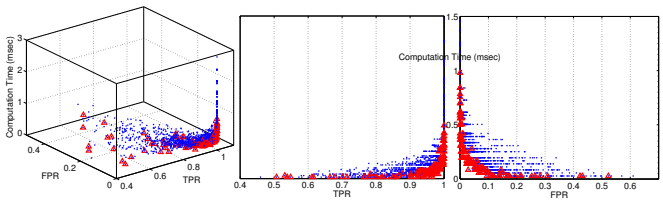

Fig. 5: Exemplary extracted Pareto Front. Each of the 1792 features are plotted as a blue dot using their TPR, FPR, and τ_j values. The extracted dominant features (that make the Pareto Front) are shown with red triangular markers. The plot is shown in 3D as well as projected 2D plots to aid visualization.

the optimization problem to select pertinent features that satisfy both the node TPR and FPR requirements with the minimum possible computation time is detailed subsequently.

The parameters of the proposed BIP are:

- $N \in \mathbb{Z}$: number of training images/samples;
- $M \in \mathbb{Z}$: number of weak learners considered, *i.e.*$|F^*|$;
- $y_i \in \{-1, 1\}$: $y_i = 1$ for positive samples, else $y_i = -1$ (negative samples);
- $h_{i,j} \in \{0, 1\}$: $h_{i,j} = 1$ if weak learner j detects sample i to be positive, else $h_{i,j} = 0$;
- $TPR \in [0, 1]$: minimum true positive rate required at the considered node of the cascade;
- $FPR \in [0, 1]$: maximum false positive rate at the node;
- $\tau_j \in \mathbb{R}$: computation time of weak learner j.

Decision Variables: The BIP decision variables are the following.

- $u_j \in \{0, 1\}$: $u_j = 1$ if weak learner j is selected, else $u_j = 0$;
- $t_i \in \{0, 1\}$: $t_i = 1$ if a positive sample i has been detected as positive (true positive) by at least one selected weak learner, else $t_i = 0$;
- $f_i \in \{0, 1\}$: $f_i = 1$ if a negative sample i has been detected as positive (false positive) by at least one selected classifier, else $f_i = 0$.

In total, there are $(2N + M)$ binary variables in the BIP, which is quite compact.

Objective Function and Constraints:

$$\begin{aligned}
\min \quad & \textstyle\sum_{j=1}^{M} \tau_j u_j && && (1)\\
\text{s.t} \quad & t_i \leq \textstyle\sum_{j=1}^{M} \frac{1+y_i}{2} h_{i,j} u_j && ,\forall i && (2)\\
& f_i \geq \frac{1-y_i}{2} h_{i,j} u_j && ,\forall (i,j) && (3)\\
& \textstyle\sum_{i=1}^{N} t_i \geq (\sum_{i=1}^{N} \frac{1+y_i}{2}) TPR && && (4)\\
& \textstyle\sum_{i=1}^{N} f_i \leq (N - \sum_{i=1}^{N} \frac{1+y_i}{2}) FPR && && (5)\\
& u_j, t_i, f_i \in \{0, 1\} && ,\forall (i,j) && (6)
\end{aligned}$$

The objective function (1) aims at minimizing the computation time. Constraints (2)-(5) express that a given rate of detection quality has to be reached (depending on the number of true and false positives). Constraints (2) link u_j and t_i variables so that $t_i = 0$ if image i has not been well-recognized.Constraints (3) link u_j and f_i variables so that $f_i = 1$ if a negative image i has been recognized as positive by at least one selected classifier. Constraint (4) expresses that the rate TPR of true positives, obtained

with the selected classifiers, has to be reached. Similarly, constraint (5) expresses that the rate FPR of false positives, obtained with the selected classifiers, must not be exceeded. The total number of constraints is $(N(M+1)+2)$, which could be huge for large N and M values. This optimization formulation is implemented using the Gurobi c++ library [8].

D. Discrete AdaBoost

Once the BIP furnishes a set of weak learners/features that fulfill the requirements set forth on the respective cascade node, the selected features are weighted and combined to obtain a strong classifier per node using AdaBoost. In this work, our implementation of the discrete AdaBoost of Viola and Jones [16] has been used because of its ease and good strong classifier construction behavior. Evidently, any other boosting framework that can accommodate a binary weak learner could be used.

E. Cascade Construction

The complete cascade structure of the final detector is built at the end of the training process. The training process involved is trivial. It relies on a labeled positive and negative sets first to learn the set of relevant features and then to use these features to train the AdaBoost classifier in each node of the cascade. To include vast number of negative training samples, the mining technique presented by Viola and Jones [17] is adopted. First, the node is constructed using a provided positive and negative samples. Once this is done, the trained nodes of the cascade (up to the current node) are subjected to a lot of negative samples (in hundreds of thousands). The mislabeled negative samples are kept for training consequent nodes of the cascade and the process continues until a tractable amount of negative samples have been tested.

VI. Experiments and Results

A. Evaluation metrics

To evaluate the detection performance, we have chosen to use the Pascal Visual Object Classification (VOC) evaluation metrics [4] as it is the well established and commonly used metrics in object detection/classification tasks. The evaluation involves a Precision-Recall curve and a single scalar quantity called Average Precision (AP), which is basically the area under the Precision-Recall curve. To determine these values the True Positives, False Positive, True Negative, and False Negatives of the test set are determined via a per-window approach [3]. The per-window approach relies on cropped labeled positive and negative train and test set. The training is performed using these cropped images and the test likewise (please refer [3] for details).

Computation time taken by the cascaded detector–relative to Dalal and Triggs detector–is another parameter taken into account. Since number of person containing candidate windows are relatively very small compared to the number of total candidate windows generated from person free zones, the total number of windows tested by cascade is highly influenced by the FPR. This means, if there are N_w candidate windows, it is safe to assume only N_w*FPR windows will pass onto the next stage. With this, if the total computation time taken by node k to evaluate a single candidate window is represented by ζ_k, the total computation time for a cascade with K nodes, ζ_K, is: $\zeta_K = \sum_{k=1}^{k=K} N_w(\text{FPR})^{(k-1)}.\zeta_k$. If we represent the time taken by Dalal and Triggs HOG to evaluate a single window to be ζ_{HOG}, the average speed up with respect to Dalal and Triggs detector would be given by eq. 2.

$$\text{Average Speed Up} = \frac{\zeta_{HOG}}{\sum_{k=1}^{k=K}(\text{FPR})^{(k-1)} * \zeta_k} \qquad (2)$$

But, recall that ζ_{HOG} and ζ_k are both integral multiples of τ, the time taken to evaluate a single HOG feature block. This simplifies the computation further and it becomes a ratio of number of constituent HOG feature blocks weighted by the cumulative FPR in the denominator.

B. Dataset

Experiments are carried out using two different sets of datasets. The first one is the public INRIA person detection dataset [2]. The training set for this dataset consists of 2416 cropped positive instances and 1218 images free of persons (out of which many negative train/test cropped windows could be generated). The test set contains 1132 positive instances and 453 person free images for testing purposes. This is the most widely used dataset for person detector validation and comparative performance analysis.

Fig. 6: Sample positive (the first four) and negative (the last four) images taken from the *Ladybug2* dataset

The second dataset is our proprietary dataset acquired using *Ladybug2* camera mounted on a mobile robot (referred as Ladybug Dataset henceforth[2]). It contains 1990 positive samples annotated by hand. It also contains 50 person free full resolution images acquired from our robotic and other rooms in the laboratory. Some 10000 negative windows are randomly sampled from these images. Sample cropped positive and negative instances are shown in fig. 6. The test set contains 1000 manually cropped positive samples and 30 person free images.

C. Results

Validation: In this framework the parameters that need to be specified are per node TPR and FPR and the depth of the decision tree to use. Another factor is the Fisher LDA weight computation. The Fisher LDA weights could be computed once using a subset of the training set and then the same weights will be used in all the cascade nodes. The other alternative is to do the weight computation specifically on

[2]Please visit `http://homepages.laas.fr/aamekonn/iros_2013/` for more illustrations

each node. To validate all this, the Ladybug Dataset training set is divided into a 60% training and 40% validation set and various train-validation cycles are performed to determine the effect of each parameter.

First, it is observed that computing Fisher LDA weights per each node makes the classifier overfit on the training set leading to a very deteriorated performance on the validation set. Hence, Fisher LDA is computed once, and the same weights are used throughout the cascade construction. Second, using a decision tree of depth of 2 showed better performance on the validation set (0.16% higher than the next best) as can be seen from the precision-recall curve in fig. 7. Third, varying the FPR showed little variation in AP but slightly better ($\approx 0.1\%$ higher) results are obtained when using an FPR of 0.4 and 0.6 during training. Evidently, higher FPR paves way to more number of cascade nodes but does not necessary result in more computation time.

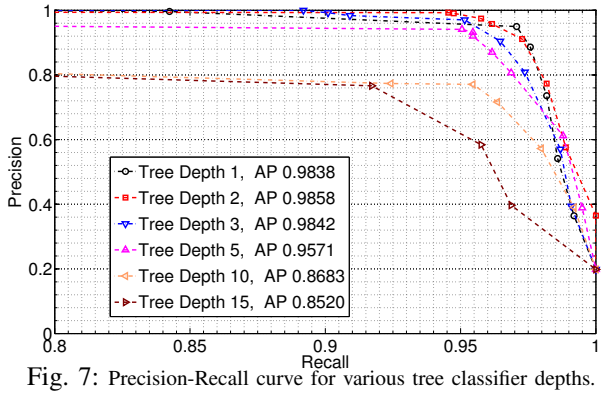

Fig. 7: Precision-Recall curve for various tree classifier depths.

Ladybug Dataset: Three different cascade classifiers are learned using FPR of 0.4, 0.5, and 0.6 but with fixed decision tree depth of 2. The results obtained are summarized in table I.

TABLE I: Comparative summary of learned cascade classifiers on Ladybug Dataset with varying FPR and Dalal and Triggs detector.

Method	K (No. of Cascade Nodes)	Average Speed Up over [2]	Average Precision
Cascade with FPR = 0.4	6	8.72x	0.9956
Cascade with FPR = 0.5	9	9.22x	0.9951
Cascade with FPR = 0.6	11	9.68x	0.9927
Dalal and Triggs [2]	–	1.0x	0.9987

As can be clearly seen from the table, with a less than 0.5% detection performance loss (AP loss), our cascade detector resulted in an 8.72x speed up on Dalal's detector and with a less than 1% loss resulted in a 9.68x speed up. Dalal and Triggs detector performance is obtained by training their open-sourced [3] detector with the Ladybug dataset training data. Fig. 8 show the precision-recall curve corresponding to the runs in table I. The features selected in the first four nodes of the cascade structure obtained using an FPR of 0.4 are shown in fig. 9 superimposed on an average gradient image of the positive data.

INRIA Person Dataset: Tests on this dataset are carried out to see the performance of our cascaded classifier on a public dataset and eventually compare its performance with Dalal and Triggs given the dataset has a lot of intra-class and inter-class variation. Again with this dataset, a decision tree depth of 2 is used. Three different cascade structures are learned using an FPR of 0.5, 0.6, and 0.7. Table II summarizes the results obtained.

[3]available here: `http://pascal.inrialpes.fr/soft/olt/`

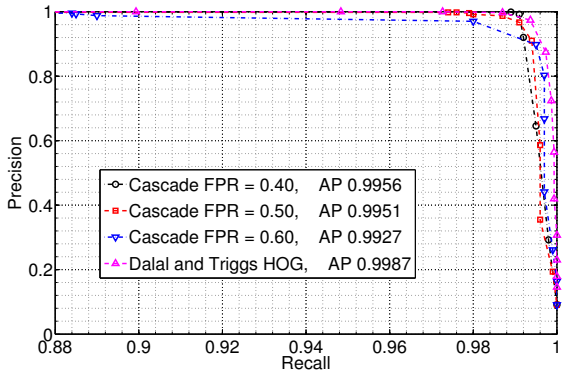

Fig. 8: Comparative Precision-Recall curve for selected cascade detector and Dalal and Triggs detector on the Ladybug Dataset.

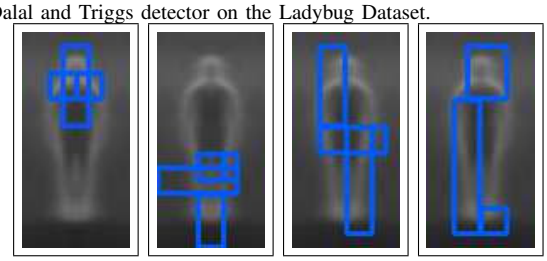

Fig. 9: Illustration of Selected HOG features of the first four cascade nodes using FPR of 0.4 and decision tree depth of 2. Clearly, the features become more computationally time consuming as one progresses down the nodes of the cascade.

TABLE II: Comparative summary of learned cascade classifiers on INRIA Dataset with varying FPR and Dalal and Triggs detector.

Method	K (No. of Cascade Nodes)	Average Speed Up over [2]	Average Precision
Cascade with FPR = 0.5	8	2.46	0.9066
Cascade with FPR = 0.6	11	2.98	0.9133
Cascade with FPR = 0.7	13	4.01	0.9198
Dalal and Triggs [2]	–	1.0x	0.9826

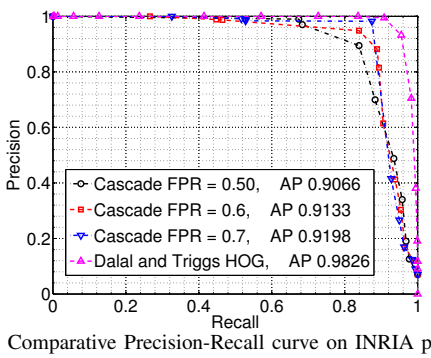

Fig. 10: Comparative Precision-Recall curve on INRIA person dataset.

Even on the challenging INRIA dataset, our cascaded detector resulted in a 4x speed up with a less than 7% AP

loss with a node FPR of 0.7. The corresponding Precision-Recall plot is shown in fig. 10.

D. Comments

The results obtained from both datasets show there is an average speed up by using out cascade framework in all cases. Of course, for difficult dataset, more features are required to attain the fixated detection performance. This in turn decreases the overall all speed gain as shown by the results from the INRIA dataset. This detector is ported on a B21R mobile robot called Rackham with an onboarded *Ladybug2* camera, the detector detects people running at a little less than 2 fps on a PIII 850 MHz PC[4]. Fig. 11 shows sample detection on a *Ladybug2* image. The results are shown as they are without any post-processing (grouping of overlapping detection). The learning/training phase is carried

Fig. 11: Sample person detection on the *Ladybug2* image.

out on a core i7 PC with an 8 GB of RAM. The two major time consuming parts are the Fisher LDA weight computation at the beginning and the BIP optimization (specially when huge data is considered). But, no cascade configuration that confirms to the current adopted scheme exceeded a 24h training period.

VII. Conclusions and Perspectives

In conclusion, a person detection framework that makes use of the proven discriminant HOG features in a cascade configuration has been presented. A new feature selection technique based on mathematical programming has been devised to select features with good detection performance and less computation time. The complete final learning system has been validated on a proprietary dataset acquired using *Ladybug2* camera, a sensor which is interesting but surprisingly marginally used in the robotics community– perhaps due to the time consumption with the associated high resolution images. The methodology is also quite suitable for conventional cameras (see our evaluation on public dataset). The final results show comparable detection performance to that of Dalal and Triggs detector while speeding up the detection by more than 8x on the *Ladybug2* images.

In the near future, we will use this detector in a tracking-by-detection framework to track all passers-by in the robot surrounding when navigating in crowds. The tracking information will then be used to realize a socially acceptable human aware navigation via control law based on visual servoing techniques.

[4]Please see `http://homepages.laas.fr/aamekonn/iros_2013/` for a video of the live run on the robot.

Acknowledgment

This work was supported by a grant from the French National Research Agency (ANR) under grant number ANR-12-CORD-0003.

References

[1] E. K. P. Chong and S. H. Zak. Multi-objective optimization. In *An Introduction to Optimization, Third Edition*, pages 541–563. John Wiley & Sons, Inc., 2008.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, Jun. 2005.

[3] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[5] P.F. Felzenszwalb, R.B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*, San Francisco, CA, USA, Jun. 2010.

[6] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[7] D. Geronimo, A.M. Lopez, A.D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1239–1258, 2010.

[8] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2012.

[9] M. Kobilarov, G. Sukhatme, J. Hyans, and P. Bataria. People tracking and following with a mobile robot using an omnidirectional camera and a laser. In *International Conference on Robotics and Automation (ICRA'06)*, Orlando, FL, USA, May 2006.

[10] A. A. Mekonnen, F. Lerasle, and I. Zuriarrain. Multi-modal person detection and tracking from a mobile robot in crowded environment. In *International Conference on Computer Vision Theory and Applications (VISAPP'11)*, Algarve, Portugal, Mar. 2011.

[11] Point Grey Inc. Ladybug2. `http://www.ptgrey.com/products/ladybug2/ladybug2_360_video_camera.asp`, 2012. [Online; accessed 29-January-2013].

[12] V. Prisacariu and I. Reid. fasthog - a real-time GPU implementation of hog. Technical Report 2310/09, Department of Engineering Science, Oxford University, 2009.

[13] T. J. Van Roy and L. A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14(7):199–213, 1986.

[14] L. Spinello and K. O. Arras. People detection in rgb-d data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, San Francisco, CA, USA, Sept. 2011.

[15] Editorial board. *Robotics and Autonomous Systems*, 58(6):IFC –, 2010. Omnidirectional Robot Vision.

[16] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai, HI, USA, Dec. 2001.

[17] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[18] X. Wang, T.X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *IEEE International Conference on Computer Vision (ICCV'09)*, Kyoto, Japan, Oct. 2009.

[19] L. A. Wolsey. Strong formulations for mixed integer programs: valid inequalities and extended formulations. *Mathematical Programming*, 97(7):423–447, 2003.

[20] B. Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, Anchorage, AK, USA, Jun. 2008.

[21] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA, Jun.

[22] Z. Zivkovic and B. Krose. Part based people detection using 2D range data and images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, San Diego, CA, USA, Nov. 2007.