Real-Time Human Detection Using Contour Cues

Jianxin Wu Christopher Geyer James M. Rehg

***Abstract*— A real-time and accurate human detector, C^4, is proposed in this paper. C^4 achieves 20 fps speed and state-of-the-art detection accuracy, using only one processing thread without resorting to special hardwares like GPU. Real-time accurate human detection is made possible by two contributions. First, we show that contour is exactly what we should capture and signs of comparisons among neighboring pixels are the key information to capture contours. Second, we show that the CENTRIST visual descriptor is particularly suitable for human detection, because it encodes the sign information and can implicitly represent the global contour. When CENTRIST and linear classifier are used, we propose a computational method that does not need to explicitly generate feature vectors. It involves no image pre-processing or feature vector normalization, and only requires $O(1)$ steps to test an image patch. C^4 is also friendly to further hardware acceleration. In a robot with embedded 1.2GHz CPU, we also achieved accurate and 20 fps high speed human detection.**

I. Introduction

Human detection in video is important in a wide range of applications that intersect with many aspects of our lives: surveillance systems and airport security, automatic driving and driver assistance systems in high-end cars, human-robot interaction and immersive, interactive entertainments, smart homes and assistance for senior citizens that live alone, and people-finding for military applications. The wide range of applications and underlying intellectual challenges of human detection have attracted many researchers.

The goal of this paper is to detect humans in real-time, with a high detection rate, and few false positives. In particular, for human detection on-board a robot, the computational efficiency of the detector is of paramount importance. Not only must human detection run at video rates, but it also can use only a small number of CPU cores (or a small percentage of CPU cycles) so that other important tasks such as path planning and navigation will not be hindered.

Recent progress in human detection has advanced the frontiers of this problem in many aspects, e.g., features, classifiers, testing speed, and occlusion handling [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. However, at least two important questions still remain open:

- **Real time detection**. The speed issue is very important, because real-time detection is the prerequisite in most of the real-world applications [12] and in a robot in particular.
- **Identify the most important information source**. Features like HOG [1] and LBP [8] have been successful in practice. But we do not know clearly yet what is the critical information encoded in these features, or why they achieve high pedestrian detection performance in practice.

In this paper we argue that these two problems are closely-related, and we demonstrate that an appropriate feature choice can lead to an efficient detection architecture. In fact, feature computation is the major speed bottleneck in existing methods. Current methods can only run at about 10 fps (frames per second) [9], even when utilizing the 100+ parallel processing threads of a GPU. Most of this time is spent in computing the features (including image pre-processing, feature construction, and feature vector normalization).

This paper makes two contributions. First, we show that the contour defining the outline of the figure is the essential information in human detection, through a series of carefully-designed experiments in Sec. III-A. *We find that the signs of comparisons among neighboring pixels are critical to represent a contour, while the magnitudes of such comparisons are not as important.*

Second, we propose to detect humans using the contour cues, and show that the recently-developed CENTRIST [13] feature is suitable for this purpose (Sec. III-B). In particular, it encodes the signs of local comparisons, and has the key capability to capture global (or large scale) structures and contours. We also compare CENTRIST and other features in Sec. III-C.

CENTRIST is very appealing in terms of speed. In Sec. IV we describe *a method for feature evaluation that does not involve image pre-processing or feature vector normalization.* In fact, we show that it is not even necessary to explicitly compute the CENTRIST feature vector, because it is seamlessly embedded into the classifier evaluation, achieving video-rate detection speed. We use a cascade classifier, and call the proposed method C^4, since we are detecting humans emphasizing the human contour using a cascade classifier and the CENTRIST visual descriptor.

C^4 produces an accurate detector running in real-time (using only one single CPU core (or thread), not involving GPU or other special hardware). We present detection results in Sec. V. We present two forms of experimental evaluation. First, we present results on a standard benchmark human detection dataset. Second, we present the results of on-line, real-time testing of C^4 on an iRobot PackBot, operating autonomously and untethered. Specifically, we demonstrate pedestrian following based on real-time on-board pedestrian detection and ground plane estimation. We will make our

J. Wu is with the School of Computer Engineering, Nanyang Technological University, Singapore jxwu@ntu.edu.sg

C. Geyer is with the iRobot Corporation, Bedford, MA 01730, USA cgeyer@irobot.com

J. Rehg is with the Center for Behavior Imaging and the School of Interactive Computing at the Georgia Institute of Technology, Atlanta, GA 30332, USA rehg@cc.gatech.edu

detection system available to other researchers to facilitate progress on this topic.

II. Related work

Accurate detection is still a major interest in human detection, especially in terms of high detection rate with low FPPI (false positive per image) [2]. Achievements have been made in two main directions: features and classifiers.

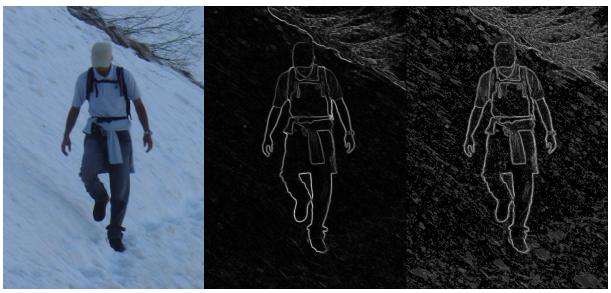
Various features have been applied to detect pedestrians, e.g., Haar features [7] and edgelet [10]. However, HOG is probably the most popular feature in human detection [1], [3], [4], [6], [8]. The distribution of edge strength in various directions seem to efficiently capture humans in images. Recently, variants of Local Binary Pattern (LBP) also show high potentials [5], [8]. A recent trend in human detection is to combine multiple information sources, e.g., color, local texture, edge, motion, etc. [14], [6], [8], [15]. Introducing more information channels usually increases detection accuracies, at the cost of increased detection time.

In terms of classifiers, linear SVM is widely used, probably for its fast testing speed. With the fast method to evaluate Histogram Intersection Kernel (HIK) [16], [17], HIK SVM was used to achieve higher accuracies with slight increase in testing time [4].

Recent research also substantially speeds up human detection. Cascade (e.g., [7], [11]) and integral image (e.g., [14], [8]) were widely used to accelerate human detection. However, the detection speed is still far slower than frame rate. Thus GPU was frequently used to distribute the computation loads into hundreds of parallel threads. For example, the system in [9] achieved about 10 fps, and similarly 4 fps in [8], both using GPU. In Sec. IV we will present a method that runs at 20 fps using only a single processing thread (and it is also very friendly to further GPU speedup). Table I compares the speed and accuracy of several fast vision-based detection methods, including C^4, the method proposed in this paper.[1]

There have been numerous previous works in the robotics community which developed pedestrian detection systems for mobile robot platforms [18], [19], [20], [21]. The majority of these works employ some form of ranging sensor (representative examples are [18] and [21]). 3D sensors has the advantage of leveraging 3D cues for detection and tracking (e.g., humans will protrude above the ground plane and can often be segmented reliably in depth), and several impressive systems have been demonstrated. However, this approach has some significant disadvantages: active ranging systems can have limited resolution and range, limited temporal sampling rates, difficulties with strong outdoor lighting, add to system expense, and emit an energy signature. Therefore it seems useful to explore the viability of passive EO sensing technologies such as video cameras.

[1][11] reported "up to 70x" speedup to HOG and 5-30 fps on different input images. Its speed in Table I is computed based on these numbers.



(a) Original image (b) Sobel image (c) Only signs

Fig. 1: Detecting humans from their contours (1b) and signs of local comparison (1c).

III. Using CENTRIST to detect human contour

A. Signs of local comparisons are critical for encoding contours and human detection

We believe that contour is the most useful information for pedestrian detection, and the signs of comparisons among neighboring pixels are key to encode the contour. Both hypotheses are supported by experiments presented in this section.

Hypothesis 1: For pedestrian detection the most important thing is to encode the contour, and this is the information that HOG is mostly focusing on. Local texture can be harmful, e.g., the paintings on a person's T-shirt may confuse a human detector. In Fig. 1b, we compute the Sobel gradient of each pixel in Fig. 1a and replace a pixel with the gradient value (normalized to $[0\ 255]$). The Sobel image smooths high frequency local texture information, and the remaining contour in Fig. 1b clearly indicates the location of a human.

Fig. 6 in [1] also indicated that image blocks related to the human contour are important in the HOG detector. However, we do not know clearly what information captured by HOG makes it successful in human detection.

We will experimentally show that contour is the important information captured by HOG. We used the original HOG detector in [1], but use the Sobel version of test images. The original HOG SVM detector was trained with features where contour and other information (e.g., fine-scale textures on the clothes) are interwoven with each other (cf. Fig. 1a). It is unusual that without modification it will detect humans on Sobel testing images where contour is the main information (cf. Fig. 1b). Surprisingly, the detection accuracy is 67% at 1 FPPI, higher than 7 out of 12 methods evaluated in [14].

Thus we believe that contour is the most important information captured by HOG and for pedestrian detection. One important difference between C^4 and existing methods is that we explicitly detect humans from the Sobel image.

Hypothesis 2: Signs of comparisons among neighboring pixels are key to encode the contour. We usually use image gradients to detect contours, which are computed by comparing neighboring pixels. We show that the signs of such comparisons are key to encode contours while the magnitudes of comparisons are not as important.

TABLE I: Speed comparison of several fast vision-based human detection methods. VGA resolution is 640x480, and qVGA is 320x240. Accuracy is at 1 FPPI (false positive per image).

Method	GPU	qVGA speed	VGA speed	speedup to HOG	Accuracy
HOG [1]	No		0.075 fps [2]	1x	74.4% (Sec. V-B)
ChnFtrs [14]	No		0.5 fps [14]		86%
HOG-LBP [8]	Yes		about 4 fps		about 87%
HOG cascade [11]	No	5-30 fps		12-70x	
GPU HOG [9]	Yes	34 fps [9]	10 fps [9]	34x [9]	similar to HOG
C^4	No	109 fps	20 fps	80x	83.5%

In order to verify this hypothesis, for a given image I, we want to create a new image I' that retains signs of local comparisons but ignores their magnitudes. In other words, we want to find an image I' such that

$$\operatorname{sgn}\left(I(p_1) - I(p_2)\right) = \operatorname{sgn}\left(I'(p_1) - I'(p_2)\right), \qquad (1)$$

for any neighboring pair of pixels p_1 and p_2. An example is shown in Eqn. 2.

$$I : \left(\begin{array}{c|c|c} 32 & 2 & 8 \\ \hline 38 & 96 & 64 \end{array}\right) \qquad I' : \left(\begin{array}{c|c|c} 1 & 0 & 1 \\ \hline 2 & 3 & 2 \end{array}\right) \qquad (2)$$

Note that the pixel 96 is converted to a value 3, because of the path of comparisons $2 < 32 < 38 < 96$. In other words, although the magnitude of comparisons in I are ignored in I', the spatial relationships among multiple comparisons in I will provide a "pseudo-magnitude" in I'. Another important observation is that gradients computed from I and I' will have quite different magnitudes. Fig. 1c shows such a sign comparison image I' (in which pixel values are scaled to $[0\ 255]$) when I is Fig. 1b. We can easily detect the human contour in Fig. 1c.

We further verified hypothesis 2 in human detection. Applying the original HOG detector to sign comparison testing images (like Fig. 1c), we achieved 61% detection accuracy at 1 FPPI (better than 7 methods evaluated in [14]). Although we observe lower detection rates when the Sobel images or the sign comparison images are used as test images, it is important to note that the classifier was trained using the original images. The fact that we obtain higher accuracies than many existing methods without modifying the HOG classifier is noteworthy. Thus we argue that the most useful information for human detection is the global contour information, and the signs of comparisons among neighboring pixels is the key to encode a contour.

B. The CENTRIST visual descriptor

We then propose to use the CENTRIST visual descriptor [13] to recognize humans, because it succinctly encodes the crucial sign information, and does not require pre- or post-processing. CENTRIST means CENsus TRansform hISTogram. We will show in this section why CENTRIST is suitable for this task, and compare CENTRIST with other popular descriptors in Sec. III-C.

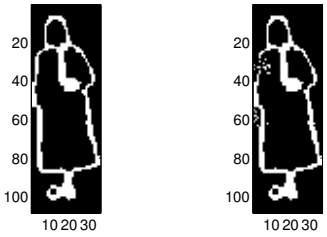
Census Transform (CT) is originally designed for establishing correspondence between local patches [22]. Census transform compares the intensity value of a pixel with its eight neighboring pixels, as illustrated in Eqn. 3. If the center pixel is bigger than (or equal to) one of its neighbors, a bit 1 is set in the corresponding location. Otherwise a bit 0 is set.

$$\begin{array}{c|c|c} 32 & 64 & 96 \\ \hline 32 & \mathbf{64} & 96 \\ \hline 32 & 32 & 96 \end{array} \Rightarrow \begin{array}{c} 1\ 1\ 0 \\ \hline 1\ \ 0 \\ \hline 1\ 1\ 0 \end{array} \Rightarrow (11010110)_2 \Rightarrow \mathrm{CT} = 214 \qquad (3)$$

The eight bits generated from intensity comparisons can be put together in any order (we collect bits from left to right, and top to bottom), which is consequently converted to a base-10 number in $[0\ 255]$. This is the CT value for the center pixel. The CENTRIST descriptor is a histogram of these CT values [13].

As shown in Eqn. 3, CT values succinctly encode the signs of comparisons between neighboring pixels. The only thing that seems to be missing from CENTRIST, however, is the power to capture global (or larger scale) structures and contours beyond the small 3×3 range.

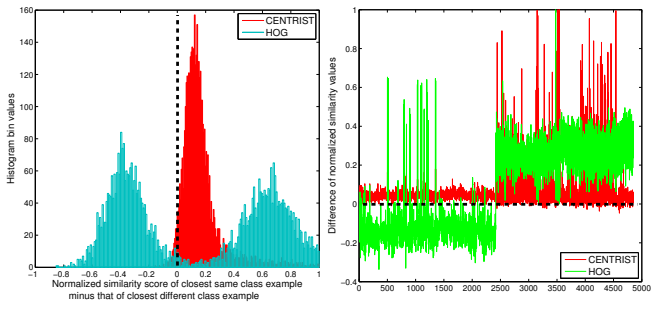
More importantly, if we are given an image I with CENTRIST $\boldsymbol{h}$, then among the small number of images I' that has a matching CENTRIST descriptor, we expect that I' will be similar to I, especially in terms of global structure or contour, which we illustrate in Fig. 2. Fig. 2a shows a 108×36 human contour. We divide this image into 12×4 blocks, thus each block has 81 pixels. For each block I, we want to find an image I' that has the same histogram and CENTRIST descriptor as I.[2] As shown in Fig. 2b, the reconstructed image is similar to the original image. The global characteristic of the human contour is well preserved in spite of errors in the left part of the image.



(a) human contour (b) reconstruction

Fig. 2: Reconstruct human contour from CENTRIST.

[2]We choose to work with small blocks with 81 pixels and binary images to make simulated annealing converge in a reasonable amount of time. Please refer to [13] for details of the reconstruction algorithm.



(a) Histogram of s_{NN} (b) Plot of s_{NN}

Fig. 3: Histogram and plot of similarity score differences.

The fact that CENTRIST not only encodes important information (signs of local comparisons) but also implicitly encodes the global human contour makes us believe that it is a suitable representation for detecting human contours.

C. Comparing with HOG and LBP

Now we will compare CENTRIST with HOG and LBP, two visual descriptors that are popular in human detection.

For classification tasks, the feature vectors of examples in the same class should be similar to each other, while examples in different classes should have dissimilar descriptors. For any example $\boldsymbol{x}$, we will compute the similarity score between $\boldsymbol{x}$ and all other examples. Let $\boldsymbol{x}_{\text{in}}$ be the most similar example to $\boldsymbol{x}$ within the same class. Similarly, let $\boldsymbol{x}_{\text{out}}$ be the most similar example that is in a different class. Obviously we want $s_{\text{NN}} = s(\boldsymbol{x}, \boldsymbol{x}_{\text{in}}) - s(\boldsymbol{x}, \boldsymbol{x}_{\text{out}})$ to be positive and large, where $s(\boldsymbol{x}, \boldsymbol{y})$ is the similarity score between $\boldsymbol{x}$ and $\boldsymbol{y}$. A positive s_{NN} means that $\boldsymbol{x}$ is correctly classified by a nearest neighbor (1-NN) rule. Thus s_{NN} is an intuitive and easy-to-compute measure to determine whether a descriptor suits certain tasks.

Fig. 3 compares CENTRIST (on Sobel images) and HOG (on original input images) using the INRIA dataset [1]. In Fig. 3a we use all the 2416 human examples, and randomly generate 2 non-human examples from each negative training image which leads to 2436 non-human examples. Fig. 3a shows the distribution (histogram) of s_{NN} for CENTRIST and HOG. Similarity scores are normalized to the range $[0\ 1]$, and a negative s_{NN} (i.e., in the left side of the black dashed line) is an error of 1-NN classifier. It is obvious that the CENTRIST curve resides almost entirely in the correct side (2.9% 1-NN error), while about half of the HOG curve is wrong (46% 1-NN error). Fig. 3b further shows that HOG errors are mostly in the first half of the dataset which are human examples.

It is argued in [13] that visual descriptors such as HOG or SIFT [23] pays more attention to detailed local textural information instead of structural properties (e.g., contour) of an image. We further speculate that this is due to the fact that the magnitudes of local comparisons used in HOG pay more attention to local textures. It is also obvious that we can not reconstruct an image from its HOG or SIFT descriptor. In Fig. 3 the HOG vectors are l_2 normalized, we set $s(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{y}$. For CENTRIST, the histogram intersection kernel [24] is used to compute similarity scores.

CENTRIST has close relationship with LBP, another popular feature for human detection. If we switch all bits '1' to '0' and vice versa in Eqn. 3, the revised formula is an intermediate step to compute the LBP value for the same 3×3 region [25]. However, the more important difference is how the LBP values are utilized. Pedestrian detection methods use "uniform LBP" [5], [8], in which certain LBP values that are called "non-uniform" are lumped together. We are, however, not able to reconstruct the global contour because the non-uniform values are missing. In addition, [5] and [8] involves interpolation of pixel intensities. These procedures make their descriptors to only encode a blurred version of the most important information, i.e., signs of neighboring pixel comparisons. We computed the distribution of s_{NN} for the uniform LBP descriptor. It has an error rate of 6.4% for the 1-NN classifier, more than twice of the error rate for CENTRIST (2.9%). However, LBP has better s_{NN} distribution than HOG (46% 1-NN error). Our conjecture is that the incomplete and blurred local sign information in LBP is still less sensitive than HOG in the presence of noise and distractions from local textures.

IV. Fast Linear Method and Detection Framework

Given the virtues of CENTRIST, we will use it to detect humans. We use 108-by-36 as the detection window size, and divide the image patch into 9×4 blocks (so each block has 108 pixels). Following [1], we treat any adjacent 2×2 blocks as a super-block and extract a CENTRIST descriptor from each super-block. There are $8 \times 3 = 24$ super-blocks, thus the feature vector for a candidate image patch has $256 \times 24 = 6144$ dimensions. A one-pixel-wide border of each super-block is not included when computing the CENTRIST descriptor because the Census Transform requires a 3×3 region.

A. Fast scanning using a linear classifier

Suppose we already trained a linear classifier $\boldsymbol{w} \in \mathbb{R}^{6144}$, we can divide $\boldsymbol{w}$ to smaller units corresponding to the super-blocks. In other words, $\boldsymbol{w}$ is considered as a concatenation of $\boldsymbol{w}_{i,j} \in \mathbb{R}^{256}, 1 \le i \le 8, 1 \le j \le 3$. Given an image patch with feature vector $\boldsymbol{f}$ (similarly separated into $\boldsymbol{f}_{i,j}$), it is classified as containing a human if

$$\boldsymbol{w}^T\boldsymbol{f} = \sum_{i=1}^{8}\sum_{j=1}^{3} \boldsymbol{w}_{i,j}^T\boldsymbol{f}_{i,j} \ge \theta. \tag{4}$$

Inspired by [26], we propose a method that computes Eqn. 4 using a fixed number machine instructions for each image patch, i.e., an $O(1)$ method. We also improve the method in [26] by using only one integral image.

Let us denote the dimension of a detection window as (h, w). A block has size $(h_s, w_s) = (h/9, w/4)$, and a super-block is $(2h_s, 2w_s)$. Given an image I, its corresponding Sobel image S, and CT image (of S) C. For a detection

window with top left corner (t, l), it is not difficult to show that the term $\boldsymbol{w}^T \boldsymbol{f}$ in Eqn. 4 is equal to:

$$\sum_{i=1}^{8}\sum_{j=1}^{3}\sum_{x=2}^{2h_s-1}\sum_{y=2}^{2w_s-1} w_{i,j}^{C(t+(i-1)h_s+x,l+(j-1)w_s+y)}, \quad (5)$$

where $w_{i,j}^k$ is the k-th component of $\boldsymbol{w}_{i,j}$, and $C(x, y)$ is a pixel in the CT image C. We starts from $x = 2$ and ends at $2h_s - 1$ to exclude the border.

We then create auxiliary images $A_{i,j}$ for $1 \leq i \leq 8$, $1 \leq j \leq 3$ with the same size as the input image I. The (x, y) pixel of $A_{i,j}$ is set to

$$A_{i,j}^{x,y} = w_{i,j}^{C(x,y)}, \quad (6)$$

then Eqn. 5 becomes

$$\sum_{i=1}^{8}\sum_{j=1}^{3}\left(\sum_{x=2}^{2h_s-1}\sum_{y=2}^{2w_s-1} A_{i,j}^{t+(i-1)h_s+x,l+(j-1)w_s+y}\right). \quad (7)$$

Using the integral image trick, the term in the parenthesis of Eqn. 7 can be computed using 3 arithmetic operations. Thus Eqn. 7 (and equivalently Eqn. 4) can be computed in $O(1)$ steps.

The advantage of using CENTRIST is that it does not require normalization. In contrast, normalization is essential in HOG [1]. We can compute $\boldsymbol{w}^T \boldsymbol{f}$ in a sum of pixel-wise contribution manner without explicitly generating $\boldsymbol{f}$.

Eqn. 7 is similar to the method for ESS with spatial pyramid matching in [26]. However, there is no need to generate multiple integral images. Instead, we define a single auxiliary image A, with

$$A(x, y) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} w_{i,j}^{C((i-1)h_s+x,(j-1)w_s+y)}, \quad (8)$$

where $n_x = 8$, $n_y = 3$. Then

$$\boldsymbol{w}^T \boldsymbol{f} = \sum_{x=2}^{2h_s-1}\sum_{y=2}^{2w_s-1} A(t + x, l + y). \quad (9)$$

Only one integral image is needed to compute Eqn. 9, which saves not only large memory space but also computation time. In practice, Eqn. 9 runs about 3 to 4 times faster than Eqn. 7. Please note that the technique of Eqn. 9 is general, and can be used to accelerate other computations, e.g., ESS with spatial pyramid matching.

The proposed method does not involve image preprocessing (e.g., smoothing) or feature vector normalization. In fact, the feature extraction component is seamlessly embedded into classifier evaluation. These properties together contribute to a real-time human detection system.

B. Detection framework

In the training phase, we have a set of 108×36 positive training image patches $\mathcal{P}$ and a set of larger negative images $\mathcal{N}$ that do not contain any pedestrian. We first randomly choose a small set of patches from the images in $\mathcal{N}$ to form a negative training set $\mathcal{N}_1$. Using $\mathcal{P} \bigcup \mathcal{N}_1$ we train a linear SVM classifier H_1.

A bootstrap process is used to generate a new negative training set $\mathcal{N}_2$. H_1 is applied to all patches in the images in $\mathcal{N}$. In this bootstrapping process, we also re-scale the negative image to examine more patches. We then train H_2 using $\mathcal{P}$ and $\mathcal{N}_2$. This process is repeated until all patches in $\mathcal{N}$ are classified as negative by at least one of $H_1, H_2, \ldots$ We then train a linear SVM classifier using $\mathcal{P}$ and the combined negative set $\bigcup_i \mathcal{N}_i$, which we call H_{lin}.

Linear classifiers ensure fast testing speed (and fast bootstrapping process). However, it has been shown that HIK achieves higher classification accuracies on histogram features than linear SVM classifiers [17], [4]. We will train a second HIK SVM classifier to achieve higher detection accuracy. We use H_{lin} on $\mathcal{N}$ to bootstrap a new negative training set $\mathcal{N}_{\text{final}}$, and train an SVM classifier using the libHIK HIK SVM solver of [27], which we call H_{hik}. In the testing / detection phase, a cascade with two nodes H_{lin} and H_{hik} is used.

We call the proposed method C^4, as we are detecting humans based on their contour information using a cascade classifier and CENTRIST.

C. Pedestrian detection on-board a robot

We integrated the C^4 pedestrian detection algorithm onto an iRobot PackBot in order to achieve on-board pedestrian detection and to enable pedestrian following. The implementation first captured images from a TYZX G2 stereo camera system and then processed the imagery using an Intel 1.2 GHz Core 2 Duo embedded in an add-on computational payload. We used the raw camera imagery to perform the detection and used the stereo range data to estimate the distance to the pedestrian. We used a particle filter to track the pedestrian between frames and to eliminate outliers. Finally, a following component was implemented to steer the robot chassis and command the neck pan axis.

We compared the basic approach described above with an optimized method that utilized the stereo data. We use the range image to provide hypotheses for where pedestrians may be standing. From the stereo data we use RANSAC to estimate a ground plane, and we sampled the depths along the ground plane's horizon. With the depth and coordinates of the plane we can calculate a box that would contain a pedestrian standing on the plane at the given position on the horizon and given distance. This gives us far fewer windows to test with the detector, which reduces both computation and false positives. Figures 4a, 4b, and 4c show the raw detections from the C^4 algorithm, the hypotheses generated from the stereo data, and the results of the C^4 classifier evaluated only on these hypotheses.

Note that the C^4 detector was tailored to work with the robot to a 3-layer cascade instead of 2-layer for faster speed (but less accurate). By default the detection procedure is as follows. The detector looks for pedestrians of different projected sizes within the image. Recall that distance to the pedestrian determines the size of the pedestrian in the image: pedestrians that are farther away appear smaller in the image; while closer pedestrians appear bigger. The detection

(a) C^4 (b) C^4+stereo (c) C^4+stereo, final

Fig. 4: On-board detection example. The three images are raw detection results using C^4, C^4+stereo, and the post-processed result of C^4+stereo, respectively. The green line is the ground plane estimated using stereo.

system searches the image at multiple scales, and for each scale runs a classifier on every single possible location of a pedestrian of that size within the image. Lacking any other information about the scene, this is the only reliable approach to detection. However, when other information is available, we should be able to use this to our advantage to decrease the number of windows on which to run the classifier, and to decrease the false positive rate. In particular, we can use information about the ground plane – which we can acquire from the stereo camera. For example, Fig 4 shows the result of the pedestrian detection classifier being run on all possible windows (C^4, Fig. 4a), and the many redundancies that are generated despite the fact that in most of the locations there cannot be a pedestrian [28]. Pedestrians are bound to the ground and we can use this fact as a prior to limit the search range (C^4+stereo, Fig. 4b). The redundancies are a problem because each of the redundant windows has to be filtered out, increasing algorithm and computational complexity. The results after post-processing are shown in Fig. 4c (cf. Sec. V-C for post-processing details).

V. Results

We experimented on the INRIA pedestrian dataset [1]. We will show the speed, accuracy, and related discussions of C^4 in Sec. V-A to V-C. Results of human detection on-board the robot are described in Sec. V-D.

There are 2416 positive training image patches and 1218 negative images for bootstrapping in the INRIA dataset. We crop the examples to 108×36 pixels, which is a tight boundary and removed the extra padding pixels. At testing time, a brute-force strategy is used to search image patches at all possible positions and scales. We successively down-sample the test image by a factor of 0.8, and scan in a grid with step size 2.

We use the groundtruth and matching criterion in [2]. A detection rectangle R_d and a groundtruth rectangle R_g is considered as a correct match if

$$\frac{\text{Area}(R_d \cap R_g)}{\text{Area}(R_d \cup R_g)} > 0.5. \tag{10}$$

We also follow [2] which requires that one groundtruth rectangle can only match to at most one detected window.

TABLE II: Distribution of computing time (in percentage).

Processing module	Percent of used time
Sobel gradients	16.55%
Computing CT values	9.36%
Integral Image	44.65%
Resizing image	5.68%
Brute-force scan	23.75%
Post-processing	0.02%

A. Detection speed

C^4 achieves much faster speed than existing human detectors. On a 640×480 video, its speed is 20.0 fps, using only 1 processing core of a 2.8GHz CPU. As far as we know, the fastest existing system (with a reasonably low false alarm rate and high detection rate) ran at about 10 fps [9], which utilized the parallel processing cores of a GPU. Detailed comparisons are available in Table I (page 3).

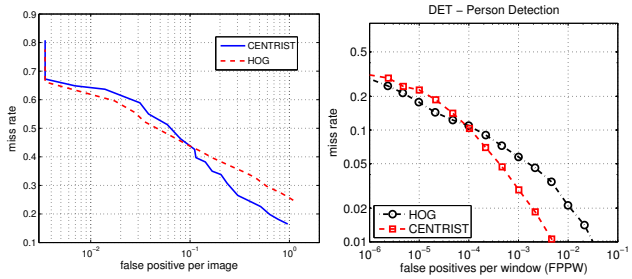
Real-time processing is a must-have property in most human detection applications. Our system is already applicable in some domains, e.g., robot systems. However, there is still huge space for speed improvements, which will make C^4 suitable even for the most demanding applications, e.g., automatic driver assistance. Table II is the break-down of time spent in different components of C^4. Most of these components are very friendly to acceleration using special hardware (e.g., GPU).

The fact that we do not need to explicitly construct feature vectors for H_{lin} is not the only factor that makes our system extremely fast. H_{lin} is also a powerful classifier. It filters away about 99.43% of the candidate patches, only $< 0.6\%$ patches require attentions of the expensive H_{hik} on the INRIA dataset. C^4 used 27.1 seconds on the INRIA dataset's test images, while the executables of the HOG detector [1] used 2167.5 seconds (i.e., an 80 fold speedup).

C^4 runs faster in smaller images. In a 480×360 YouTube video with many pedestrians, its speed is 36.3 fps. Its speed is 109 fps on 320x240 frames.

B. Detection Accuracy on the INRIA dataset

Accuracy of the system using both the false positive per window (FFPW) and image (FPPI) metrics are shown in Fig. 5. In Fig. 5 we compare with HOG [1] (using



(a) FPPI Results (b) FPPW Results

Fig. 5: Comparison C^4 with HOG on the INRIA dataset.

Fig. 6: Example of post-processing.

executables accompanying [1]). We will compare with other methods directly using the accuracy numbers published in respective papers.

C^4 detects 83.5% humans at 0.96 false detection per image. It is comparable to the state-of-the-art results on the INRIA dataset, e.g., ChnFtrs [14] and HOG-LBP [8]), both having around 86% detection rate at 1 FPPI. Multiple information channels were used in these methods. We could also use multiple channels to further improve C^4. C^4 has higher accuracies than HOG [1] (74.4% at 1 FPPI, Fig. 5a) and many other methods compared in [14], [2].

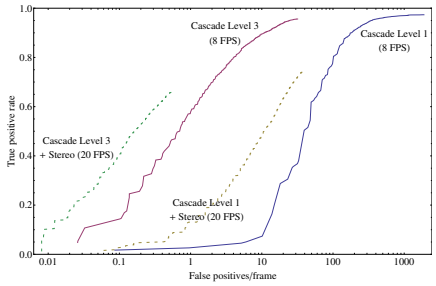
Fig. 5b shows the FPPW performance of H_{hik} (H_{lin} is not used when computing the FPPW curve.) The FPPI and FPPW numbers are not linearly correlated but have similar trends. C^4 outperforms HOG when false positive rate is $\geq 10^{-4}$ (or 0.1 in the FPPI curve), and is not as good as HOG in the range of lower false positive rates. But they converge in the left end of both curves.

C. Importance of post-processing

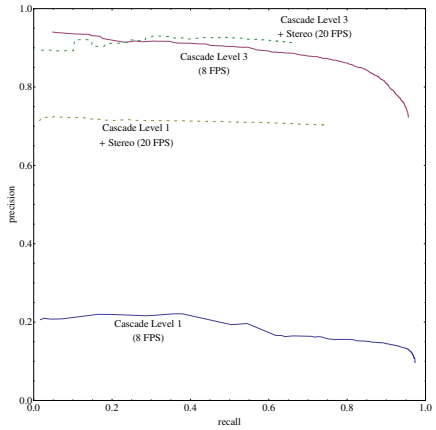
C^4 and HOG intersected at 10^{-1} and 10^{-4} respectively at FPPI and FPPW curves. The non-maximal suppression (NMS) step contributes to this relatively big difference. In C^4 a location is treated as a false positive if there are less than 3 detected windows at that location. This requirement will not hurt true positives because our step size is 2 and there are usually many detection windows around true humans.

A small step size also means that NMS will greatly reduce the number of false detections. Examples are shown in Fig. 6. There are 17 and 5 false detections in these two images, respectively.[3] After NMS, the first image only has 1 false positive, and the second image does not contain any remaining false detection.

[3]The middle part of the second image contains 2 false detections which are very close to each other.



(a) ROC



(b) PR

Fig. 7: ROC and Precision-Recall curves for the combined and standalone C^4 detector, as well as at different cascade levels.

The HOG curve in Fig. 5a is slightly different from [14]. In Fig. 5a HOG detects 34% of the humans with only 1 false positive in all test images. However, HOG only detects about 10% of the humans with 1 false detection when evaluated in [14]. On the contrary, [14] reported a higher HOG detection rate at 1 FPPI (77%) than that in our experiments (74%). Although it is not totally clear what makes these differences, we believe that beyond non-maximal suppression, tightness of the detection window is also an important factor.

We used a very tight 108×36 bounding box during the training time. We relaxed detected patches to 120×42 during the post-processing step. It seems that the overly relaxed detection window in [2] or [1] is adverse when we seek an extremely low false positive rate.

D. Detection results on-board a robot

In order to better understand the performance of the combined approach (C^4+stereo) on the robot, we tested on the images collected at iRobot's Bedford facility. Figures 7a and 7b show a detailed analysis, with ROC curves (Fig. 7a) and precision-recall curves (Fig. 7b) for both approaches, as well as showing the performance for the different cascade levels of the 3-layer C^4 cascade tailored for the robot.

The ROC curves demonstrate a reduction in false positives, however, the detection rate peaks lower than the standalone version. This is likely because the standalone C^4 was not explicitly trained on the output of the ground plane pedestrian hypotheses generator. The precision recall curves similarly show an increase in precision with the stereo approach at higher recall rates-though again the recall rate reaches a lower maximal value with the stereo approach. We show ROC and precision-recall curves at different cascade levels to show the comparative improvements due to cascade level.

Overall, the combined approach results in fewer false positives, and is faster at approximately 20 frames per second (50 milliseconds) on the embedded Intel 1.2 GHz Core 2 Duo. Alone, the C^4 runs at approximately 8 frames per second (120 milliseconds) on the same hardware. The combined approach results in a 60% reduction in computation, and depending on detection rate, almost a factor of 5 reduction in false positives.

VI. Conclusions and future work

In this paper we proposed a real-time and accurate human detector, C^4, which detects humans using the contour cues, a cascade classifier, and the CENTRIST visual descriptor.

First we show, through carefully designed experiments, that contour is the most important information source for human detection, and the signs of comparisons among neighboring pixels are the key to encode contours.

We then show that CENTRIST [13] is particularly suitable for human detection, because it succinctly encodes the sign information, and is able to capture large scale structures or contours.

A major contribution of this paper is extremely fast human detection. C^4 detects humans with 20 fps speed on 640x480 images, using only 1 processing thread, and achieves accuracies comparable to the state-of-the-art.

Time consuming pre-processing and feature vector normalization are not needed in CENTRIST. Furthermore, using a linear classifier and CENTRIST, we do not need to explicitly generate the CENTRIST feature vectors and it takes only $O(1)$ operations to evaluate an image patch.

Currently C^4 has slightly lower detection accuracy than methods such as those in [8], [2]. However, similar to the techniques in [8], [2], we believe that the accuracy of C^4 can be improved by using multiple information sources, e.g. incorporating color and other feature types. In particular, multiple feature channels will help C^4 under very strict false positive requirements. Furthermore, the speed of C^4 can be further improved by using special hardware like GPU.

Finally, on an iRobot PackBot with embedded Intel Core 2 Duo 1.2GHz CPU, we combined C^4 with a stereo vision system, and achieved accurate and video rate (1.2GHz) human detection without hindering other robot functionalities.

Acknowledgement

We thank the anonymous reviewers for their useful comments and suggestions. We are grateful for the support of the Office of Naval Research under prime contract N00014-09-C-0101, and the NSF under RI award 0916687. J. Wu is supported by the Singapore MoE AcRF Tier 1 grant RG 34/09.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, 2005, pp. 886–893.
[2] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *CVPR*, 2009.
[3] P. F. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *CVPR*, 2008.
[4] S. Maji and A. C. Berg, "Max-margin additive classifiers for detection," in *ICCV*, 2009.
[5] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou, "Discriminative local binary patterns for human detection in personal album," in *CVPR*, 2008.
[6] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis, "Human detection using partial least squares analysis," in *ICCV*, 2009.
[7] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *ICCV*, 2003, pp. 734–741.
[8] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *ICCV*, 2009.
[9] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele, "Sliding-windows for rapid object class localization: A parallel technique," in *DAGM-Symposium*, 2008.
[10] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *IJCV*, vol. 75, no. 2, pp. 247–266, 2007.
[11] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *CVPR*, vol. 2, no. 1491-1498, 2006.
[12] D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf, "Survey on pedestrian detection for advanced driver assistance systems," *IEEE TPAMI*, vol. 32, no. 7, pp. 1239–1258, 2010.
[13] J. Wu and J. M. Rehg, "CENTRIST: A visual descriptor for scene categorization," *IEEE TPAMI*, vol. to appear.
[14] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *BMVC*, 2009.
[15] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *CVPR*, vol. I, 2005, pp. 878–885.
[16] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *CVPR*, 2008.
[17] J. Wu and J. M. Rehg, "Beyond the Euclidean distance: Creating effective visual codebooks using the histogram intersection kernel," in *ICCV*, 2009.
[18] K. O. Arras, Ó. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," in *ICRA*, 2007.
[19] T. Nakada, S. Kagami, and H. Mizoguchi, "Pedestrian detection using 3D optical flow sequences for a mobile robot," in *Sensors*, 2008, pp. 776–779.
[20] D. Schulz, W. Burgard, A. Fox, and D. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *Intl. J. of Robotics Research*, vol. 22, no. 2, pp. 99–116, 2003.
[21] L. Spinello, K. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3D range data," in *AAAI*, 2010.
[22] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *ECCV*, vol. 2, 1994, pp. 151–158.
[23] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[24] M. J. Swain and D. H. Ballard, "Color indexing," *IJCV*, vol. 7, no. 1, pp. 11–32, 1991.
[25] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE TPAMI*, vol. 24, no. 7, pp. 971–987, 2002.
[26] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *IEEE TPAMI*, vol. 31, no. 12, pp. 2129–2142, 2009.
[27] J. Wu, "A fast dual method for HIK SVM learning," in *ECCV*, ser. LNCS 6312, 2010, pp. 552–565.
[28] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *IJCV*, vol. 80, no. 1, pp. 3–15, 2008.