Person Following using Histograms of Oriented Gradients

Jonathan Brookshire

iRobot Corporation
8 Crosby Drive, Bedford, MA 01730
jbrookshire@irobot.com

Abstract. In order for robots to effectively interact with people in close proximity, the systems must first be able to detect, track, and follow people. This paper describes results from the development of a mobile robot which will follow a single, unmarked pedestrian using vision. This work demonstrates an improvement over existing pedestrian following applications because (1) it uses sufficiently strong classifiers such that it does not need to adapt to any particular pedestrian, (2) uses only vision and does not rely on any laser range devices, (3) provides a single point benchmark for the level of performance required from a detector to achieve pedestrian following, and (4) its performance is characterized over several kilometers in both rainy and dry weather conditions. The system leverages Histograms of Oriented Gradients (HOG) features for pedestrian detection at over 8 Hz using video from a monochrome camera. The pedestrian's heading is combined with distance from stereo depth data to yield a 3D estimate. A particle filter with some clutter rejection provides a continuous track, and a waypoint follow behavior servos the iRobot PackBot robot chassis to a desired location behind the pedestrian. The final system is able to detect, track, and follow a pedestrian over several kilometers in outdoor environments, demonstrating a level of performance not previously shown on a small unmanned ground vehicle.

Keywords: person following, pedestrian tracking, histogram of oriented gradients, detection error tradeoff, weather

1 Introduction

One of our long term goals is to design systems that enable humans and robots to work cooperatively, side-by-side in real world environments. Here, we are interested in situations where the robot and human are adjacent in the environment and the operation of the robot is not the human's primary task. Enabling natural and efficient human interaction with the robot in this kind of situation is an ambitious goal, requiring advances in the robot's ability to interpret human commands and react to its environment with context and high level reasoning. We concentrate on the first part of the problem: keeping the robot physically adjacent to the human(s). This task involves detecting a pedestrian, tracking his/her path, and navigating the robot to a desired location with respect to the pedestrian. A video of our results is available at [23]. A popular example of a pedestrian following application is the robotic "mule" that hauls gear and supplies for a group of dismounted soldiers. The same technology could also be used as a building block for everything from elder care to smart golf carts.

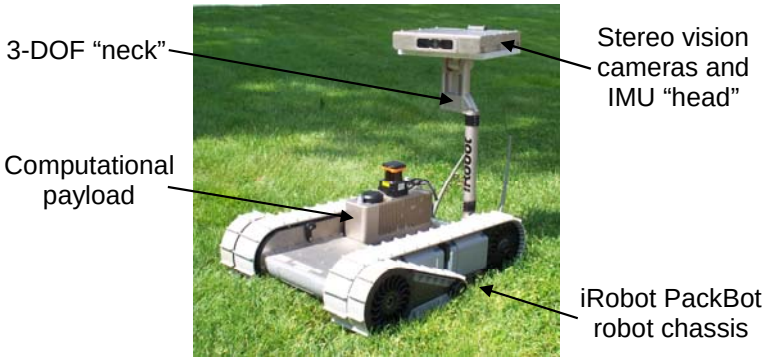

Fig 1 The robot is an iRobot PackBot chassis (tracked vehicle) augmented with stereo vision and additional computing power

Our work differs from previous efforts in several ways. First, our system does not require tuning and does not attempt to learn any particular target. Although on-line learning can have advantages (e.g., by selecting the target's resilient features), the pedestrian's constantly changing pose, lighting conditions, background, etc. make this a challenging learning problem. Our system is able to follow a pedestrian target without any training period or prior knowledge of the specific target. Second, many approaches rely only on depth data for detection. The advantages of this approach are reduced data rates and simplified, real-time detection schemes. Vision, however, offers a rich set of features which can allow

pedestrians to be identified where depth data is insufficient. Additionally, building an approach that relies only on vision allows for the later possibility of fusing, for example, LIDAR data with visual detections to improve performance. This is of particular interest when one sensor may fail (e.g., dusty or rainy conditions). Third, building a complete person following system has allowed us to establish a single reference point for required detector performance (e.g., false positive and miss rate). Although this result will not uniquely apply to all systems, we have found it critical in our practical application, allowing us to design person detection schemes with a performance goal. Fourth, although many systems have demonstrated person following, few have quantified the performance over a variety of operators and over several kilometers of distance in realistic, all-weather scenarios.

As shown in Fig 2, the pedestrian detection is performed using the video stream from a single camera of a stereo pair (the stereo depth data is used only to estimate the pedestrian's distance). The demonstration was developed on an off-the-shelf iRobot PackBot. The system was then upgraded with our standard, modular computational payload (see Fig 1). It provides an Intel 1.2 GHz Core 2 Duo-based computer, GPS (not used), LIDAR (used only for ground truth, see Following Performance), and IMU. In addition, a Tyzx G2 stereo camera pair ("head") is mounted at the top of a 3-DOF neck. During following, only the pan (left-right camera movement) axis of the neck is moved in an attempt to keep the target in the center of the field of view. By decoupling the orientation of the head and the chassis, we are able to maintain track while placing fewer requirements on the motion of the chassis. The robot has a top speed of about 2.2 m/s (5 MPH) and the software was written in the iRobot Aware 2.0 Intelligence Software.

We perform several experiments to validate our system's performance in both dry and rainy weather conditions. We evaluate the Detection Error Tradeoff (DET) curve of the pedestrian detector on widely used datasets for comparison to other algorithms. We also characterize the complete performance of our system (detector, tracker, and follower) against a ground truth derived from human-annotated LIDAR, to verify that the end-to-end person following system is operating correctly.

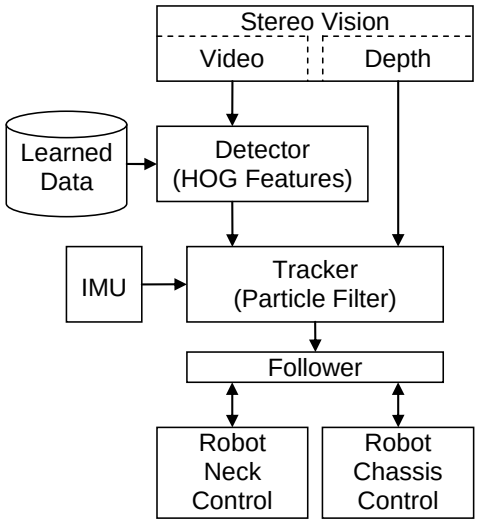

Fig 2 System architecture

1.1 Related work

Because person following must operate in real-time in order to adapt to changes in the target's trajectory, many existing person following solutions have found various ways to simplify the perception problem. Dense depth or scanned range data have been used to identify and follow people effectively. For example, in our previous work [2], we used a Support Vector Machine (SVM) trained on depth data from an infrared ranging SwissRanger. Satake and Minura [1] used the depth information from stereo cameras and templates to identify people. The system developed by Kleinehagenbrock [4] fused information from a LIDAR with estimates based on skin-color to follow people. Other systems [7-9] relied primarily on LIDAR data to perform following. Our system differs from these in that we use depth information only to estimate the pedestrian's distance and not for detection. This is an important distinction because the depth data may not be as feature rich or available in all weather conditions (see Performance in Rain). Note that we focus only on pedestrian (upright, walking people) detection and not general person (various postures, etc.) detection.

Other methods have been developed that rely primarily on vision for detection, but attempt to learn features of a particular person on-line. In [3], Kwon developed a system which uses two cameras and learns a color histogram describing the pedestrian to follow. The robot developed by Yoshimi [6] used stereo vision and adjusted the features being tracked based on several conditions.

Similar approaches, e.g., [10, 11], use color space or contour detection to find people. Our system differs from these systems in that we use a different set of features (see below) and do not adjust our tracker to any particular pedestrian in the scene. This allows our system to be used with any operator without prior, user-specific training or tuning.

In addition to these systems, which provide complete person following, our work is also related to many detection and tracking research efforts. Our detector is based on the Histogram of Oriented Gradient (HOG) features developed by Dalal [13, 14] and uses the same machine learning applied by Zhu [12] (see Detection). HOG features have been shown to be computationally efficient and robust to lighting and color changes.

The sections that follow describe each aspect of the system in more detail. The Detection section details the HOG detector and its implementation. The Tracking section discusses the particle filter implementation used to filter the incoming detections. Finally, the Following section describes the navigation algorithm used on the robot.

2 Detection

Our algorithm uses the Histogram of Oriented Gradient (HOG) features originally developed by Dalal [13, 14]. Zhu [12] applied a series of machine learning techniques and adaptations to [13], which allowed the system to run in real-time. Our detection algorithm is similar to that of [12], but we use slightly different learning parameters (see The Off-Line Learning Process for details) and make a different trade-off between speed and performance. Due to page limitations, we provide only a brief discussion of the detection strategy here; we refer the reader to [12-14] for complete details.

The algorithm works by learning a set of linear Support Vector Machines (SVMs) trained on positive (pedestrian) and negative (non-pedestrian) training images. This off-line learning process generates a set of SVMs, weights, and image regions that can then be used on-line to classify an unknown image as either positive or negative.

2.1 The Features

As with many computer vision applications, part of the challenge is to find a descriptive set of features. If the feature space is rich enough – that is, it provides sufficient information to identify targets – these features can be combined with machine learning algorithms to classify targets and non-targets. The work done in [14] demonstrated success using HOG features for pedestrian detection. A single HOG is a way to encode local gradients. In this process, the gradient is first calculated for each pixel. Next, the training image (see Fig 3) is divided into a number of sub-windows, often referred to as "blocks". The blocks span size from 8 to 64 pixels, have various length-to-width ratios, and densely cover the image (i.e., overlap). Each block is divided into quadrants and the HOG of each quadrant is calculated. A HOG is a histogram with nine evenly spaced bins for orientation into which the gradients vote (nine bins was suggested in [13] as being the most effective for classifying pedestrians). Thus, each quadrant produces nine features for a total of 36 features per block.

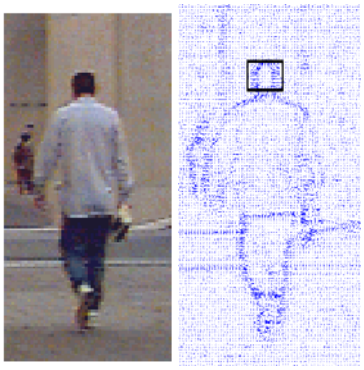

Fig 3 A typical pedestrian training image (left) and its gradient (right)

2.2 The Off-Line Learning Process

The algorithm learns what defines a pedestrian by examining a series of positive and negative training images. Because this process is time consuming and does not need to be repeated, it is performed off-line. During this learning process, a single block is randomly selected (e.g., see Fig 3, right). The 36D-feature HOG for this block is then calculated for some subset of the positive and negative training images. A linear SVM is then trained on the resulting HOGs to develop a maximally separating hyperplane. We use N-fold cross validation to judge the performance of the classifiers: blocks that distinguish humans and non-humans

well will result in a quality SVM classifier. The SVM's performance, then, represents the performance of that particular block.

In general, a single block will not be sufficient to classify positive and negative images successfully. However, these "weak" block classifiers can be combined to form stronger classifiers. The AdaBoost algorithm [21] provides a statistically founded means to choose and weight a set of weak classifiers. The algorithm repeatedly selects weak classifiers and weights and sums the score from each classifier into an overall score.

Performance is further improved by recognizing that, as the 64x128 pixel detection window is densely scanned across an image, many of the detection windows can be easily classified as not containing a pedestrian. A rejection cascade is employed to take advantage of this kind of situation. Our cascade uses several sequential AdaBoost-learned classifiers to discard obviously negative windows quickly [12]. Thus, the cascade efficiently spends its time classifying difficult windows. Although we follow the algorithm by Zhu [12], we adjusted the learning parameters (e.g., target overall false positive rate) to make a different performance/speed trade-off. We adjusted the number of rejection cascade levels empirically until a suitable frame rate was obtained. Whereas Zhu had 30 rejection levels, we had only seven and achieved a frame rate of about 8 Hz.

The learning process was distributed onto 10 processors (using the MPICH multiprocessor software architecture) to decrease training time. Training on 2000 images from the INRIA training dataset [13] took about two days.

2.3 The Detection Process

In order to detect pedestrians at various distances and positions, the 64x128 pixel detection window is scanned across the image in position and scale. The monochrome video is 500x312 pixels and, at 16 zoom factors, requires a total of 6,792 detection evaluations per image. With this many evaluations, scaling the image, scanning the detection window, and calculating the HOG features proves too slow. To compensate, the algorithm applies the Integral Histogram (IH) technique as described in [12,18].

In addition to using the IH technique, we also improve performance by scaling the IH rather than scaling the image. In this process, (1) the IH is calculated for the original image, (2) the IH is scaled, and (3) the HOG features are calculated.

Because the IH is calculated only once in (1), the scaling in (2) is only an indexing operation, and the IH provides for speedy calculation of the HOG in (3), the process is appropriate for real-time operation. Thus, by scaling the IH (as shown in Fig 5) instead of scaling the image directly (Fig 4), the processing time is reduced by 64%. It is worth noting, however, that the two strategies are not mathematically equivalent. Scaling the image (e.g., with bilinear interpolation) and then calculating the IH is not the same as calculating the IH and scaling it. That said, both algorithms seem to work well in practice and the latter is significantly faster. This IH scaling technique was also used during the off-line learning process.

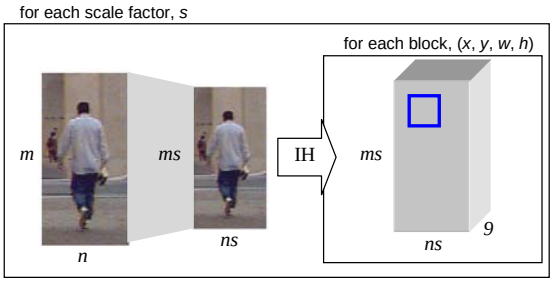

Fig 4 Repeatedly scaling the image and calculating the IH proved too slow

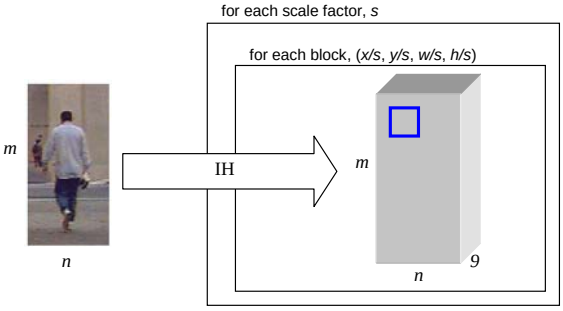

Fig 5 Instead, we create a single IH and simply adjust the index based on the scale factor

3 Tracking

The task of the tracking algorithm is to filter the incoming detections into a track which can be used to follow the pedestrian continuously. This tracker is an important element because the raw pedestrian detections cannot be tracked unfiltered. The detector will occasionally fail to detect the pedestrian, leaving a unfiltered system without a goal location for some period of time. Additionally, the detector will occasionally detect the pedestrian in the wrong position. An unfiltered system might veer suddenly due to a single spurious detection. Our

particle filter implementation mitigates these affects and enforces limits on the target's likely motion.

The heading of the pedestrian can be estimated using the camera's parameters and the pixel locations of the detections. The distance from the robot's head is then estimated directly from the stereo camera's depth data.

For our experiments, we use a single target tracker which serves to filter clutter and smooth the response when the pedestrian is missed. In the case of a moving platform chasing a moving target, the problem is complicated because the tracker must account for both the motion of the platform and the target.

The filtering is performed using a particle filter where each particle is processed using a simple Kalman filter. The pedestrian's state is represented as $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$ and we empirically find that this constant velocity model was sufficient to follow a pedestrian at walking speeds. Each particle includes the pedestrian's state and covariance matrix. Each detection in each frame triggers an update cycle where the input detections are assumed to have a fixed covariance; the prediction stage propagates the state based on velocity and noise parameters.

Motion of the platform. The state of the platform (e.g., position and velocity) could be incorporated as part of the system state and modeled by the particle filter. To avoid the added computational complexity, however, we chose to simplify the problem and assume that the motion of the platform is known. The stereo vision head also has an IMU sensor which provides angular rate information. As the head moves (either from the motion of the neck, chassis, or slippage), readings from the IMU allow the system to update the pedestrian's state relative to the chassis. In practice, we found that assuming the accelerations and chassis motion had no noise was reasonable. Unlike, for example, mapping applications where accelerometer noise could accumulate, our application uses accelerations to servo the head relative to its current position – i.e., absolute position is not important for our application.

Clutter. Occasionally, the detector will also generate spot noise, or clutter. These clutter detections are relatively uncorrelated and may appear for only a single frame. However, they can be at drastically different positions from the target and may negatively affect the track. As described in [20], we allow detections to be associated with a "clutter target" with some fixed likelihood (which can be thought of as a clutter density). For each particle individually, each

detection is associated either with the pedestrian target or the clutter target based on the variance of the pedestrian target and the clutter density. In other words, if a detection is very far from a particle, and therefore unlikely to be associated with it, the detection will be considered clutter. This process works well, but degenerates in the case where the target has a very large variance; the fixed clutter density threshold causes the majority of detections to be considered clutter and the tracker must be manually reset. However, this only occurs when the tracker has been run for an extended period of time (several minutes) without any targets. The situation might be handled with a dynamic clutter density or a method to reset the tracker when variances become irrelevantly large.

4 Following

The tracker provides the vector $\vec{p}$ which describes the position of the pedestrian relative to the robot chassis. Our following algorithm, as shown in Fig 6, is a "greedy" tracker in that it attempts to take the shortest path (from C) to get several meters behind the pedestrian, facing the pedestrian (to C').

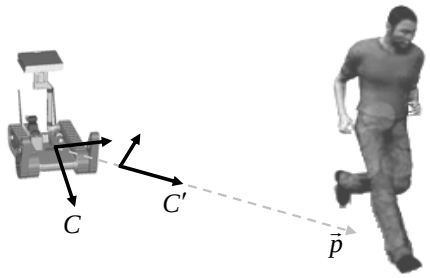

Fig 6 The follower attempts to stay 5 m behind the pedestrian, facing him

The C' frame is provided as a waypoint to an Aware 2.0 waypoint module. The Aware 2.0 Intelligence Software uses a model of the platform to generate a number of possible paths (which correspond to a rotate/translate drive command) the robot might take. Each of these potential paths is scored by the waypoint module and any other modules (e.g., obstacle avoidance). The command of the highest scoring path is then executed.

The greedy following used here works well outdoors, but does clip corners. Another possible following strategy is to servo along the pedestrian's path. This strategy has the advantage of traveling the hopefully obstacle free path of the pedestrian, but can result in unnecessary robot motion and did not seem necessary for the initial outdoor environments. For robust indoor and outdoor following, it

will be necessary to perform either some path planning on a locally generated map or follow the path of the pedestrian.

5 Performance and Experiments

Fig 7 Video [23] captures of the pedestrian following in operation through a variety of terrains

As shown in the series of frame captures in Fig 7, the person following is reasonably robust to changes in the target's pose. This strength results from the choice of features (e.g., HOGs are robust to changes in lighting), visual processing (e.g., processing at different scales and positions), and training data (e.g., the training data includes a variety of poses). Forward, backward, and side aspects of the pedestrian are detected reliably. The robot uses about 70% of its 1.2 GHz Intel Core 2 Duo and runs its servo loop at an average of 8.4 Hz (the remaining processing power will be used for gesture recognition and obstacle avoidance). With this configuration, we are able to travel paths similar to those shown in Fig 8. This particular path was ~2 km (1.25 miles) long and was logged using the robot's GPS. The path traveled over unimproved surfaces (as shown in Fig 7) and paved parking lots and sidewalks (as shown in the path in Fig 8).

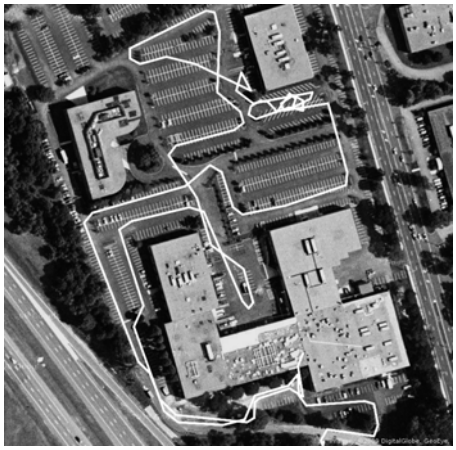

Fig 8 A typical run where the robot followed its target over ~2 km continuously

5.1 Detector Performance

We compared our pedestrian detector to those of Dalal [13] and Zhu [12] using the DET curve as described in [13]. The DET curve describes how changing the system's detection threshold affects the miss rate and False Positives Per Window (FPPW). The published results of Dalal and Zhu, tested on the INRIA pedestrian database, are shown in Fig 9. Dalal achieved the best overall results, but was not suitable for real-time operation. Zhu improved the real-time capabilities of the system, with slight penalties in performance. We use the same methods as Zhu, but make a different performance/speed tradeoff. Whereas Zhu's rejection cascade had 30 levels, we limit ours to only seven levels. Our results are shown in Fig 9 on the same INRIA database. As expected, we achieve lesser detection performance, but benefit from an estimated 36% speed up over Zhu.

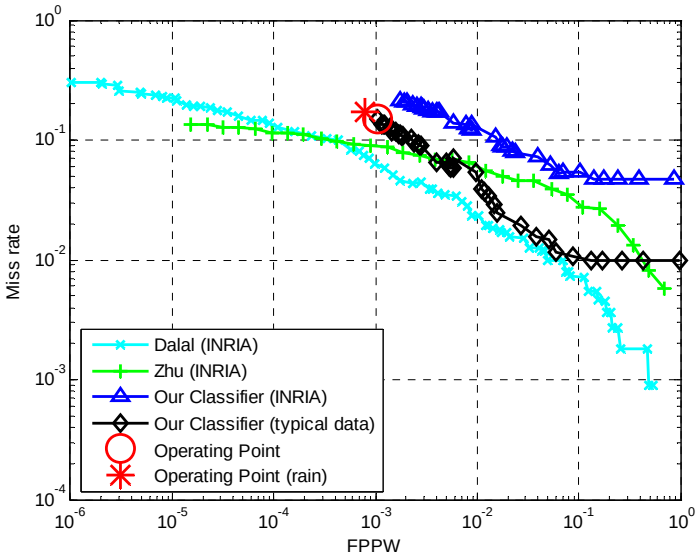

Fig 9 Compared to those of Dalal [13] and Zhu [12], we have a higher miss rate and higher false positive per window score. However, our detector runs an estimated 36% faster

During operation, however, we noticed that the performance was better than that predicted on the INRIA database. Indeed, the INRIA database is more difficult than our typical data. We re-tested our system against our dataset and again produced the DET plot. Although this curve cannot be meaningfully compared to the other curves, it does suggest what level of performance is necessary to do effective person following: we successfully operate at the point circled, at 0.1% FPPW and 15% miss rate. This operating point will be important as we continue our work, allowing us to precisely target our design and retrain future classifiers knowing a minimum required level of performance.

5.2 Following Performance

In order to characterize the ability of the entire system to follow a pedestrian, we compare the estimated track position of the pedestrian to a ground truth position. To do so, we had nine test subjects walk a combined total of about 8.7 km (5.4 miles) and recorded their position relative to the robot as estimated by the tracker. Test subjects were asked to walk at a normal pace (4-5 km/h) and to try to make about the same number of left and right turns. We also logged data from the on-board LIDAR and hand-annotated the data for ground truth. To our knowledge, no previous person following system has proved its success over these distances or with even this many users.

As described in [19], we calculated several statistics comparing the estimated and true tracks. In addition to the mean, median, and standard deviation, we calculate results for the spatial offset of the tracks (the 2D shift that best aligns the tracks), the temporal offset of the tracks (the shift in time that best aligns the tracks), and the combined temporal and spatial offset.

Table 1 The error (in meters) between the estimated and ground truth (hand-annotated from LIDAR) positions for all test subjects

	Mean Error	Median Error	Standard Deviation of Error	Minimum Error	Maximum Error
Uncorrected	0.2837	0.2248	0.29019	0.001835	3.0691
Spatially Corrected	0.24239	0.18314	0.28382	0.001095	3.0915
Temporally Corrected	0.27811	0.2206	0.2994	0.001919	5.7675
Spatio-temporally Corrected	0.23513	0.17688	0.29361	0.001416	5.7499

Table 1 shows the results averaged from all the test subjects. Without any corrections, the system tracked the pedestrian within an average error of 0.2837 m. The average spatial bias was 0.134 m and 0.095 m in the x and y directions, respectively. The average temporal offset was 74 ms, which is less than the pedestrian tracker's frame period (~120 ms).

Since the average temporal offset was less than a cycle of the detection algorithm, it is an acceptable error. To better understand the overall track error, Fig 10 shows a sample plot from one of the test subjects comparing the pedestrian's actual and estimated distance from the robot. The dotted lines show the error bounds provided by Tyzx for our 6 cm-baseline G2 stereo system. As can be seen from the best fit line, the system's estimates are slightly biased (13.8 cm bias at 5 m actual distance), but still fall well within the sensor's accuracy limits. Some errors fall outside the boundaries, but these are caused by cases when the pedestrian exited the camera's field of view and the tracker simply propagated the position estimate based on constant velocity.

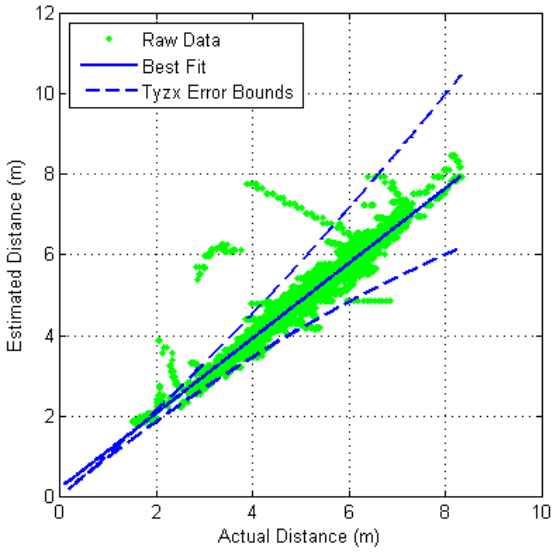

Fig 10 The accuracy of our system while tracking a pedestrian is within the limits of our stereo vision system

Finally, Fig 11 is a 2D histogram of a sample pedestrian's position relative to the robot. The robot is located at (0, 0), oriented to the right; the intensity of the plot shows areas where the pedestrian spent most of his time. As expected, the system was able to place the robot about 5 m behind the pedestrian most frequently. The distribution of the pedestrian's position is reflective both of the pedestrian's dynamics (how fast he walked and turned) and the robot's dynamics (how quickly the robot could respond to changes in the pedestrian's path).

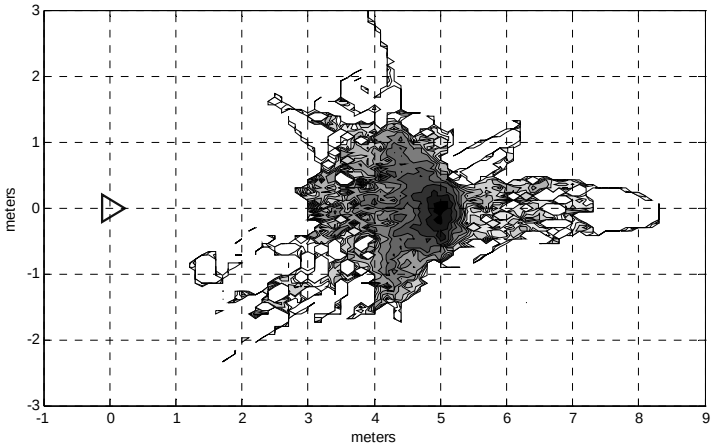

Fig 11 A 2D histogram of the pedestrian's position relative to the robot shows that the system was able to position the robot in the anticipated position, 5 m behind the target, most frequently. The robot is depicted as a triangle at (0, 0), oriented to the right

5.3 Performance in Rain

To characterize our system further, we also performed testing outdoors in a brief rainstorm. During the storm, the NOAA report [22] for the nearest airport measured the average hourly rainfall at 2.5 mm/hr (0.1 in/hr) and described it as "light rain."

Fig 12 The system successfully followed the pedestrian even during a brief rainstorm

The system was able to operate successfully despite a considerable build up of rain on the camera's protective window; the detector was still able to locate the pedestrian in images as blurry as Fig 13. Depth data from the stereo cameras degraded quickly since drops in front of either camera resulted in large gaps in the depth data. Systems which rely primarily on depth data for detection (e.g., rely only stereo depth data) would fail quickly. Our system was robust to this loss of data since only an average of the entire pedestrian's distance was used. The mean track error in rain was comparable to that in dry conditions, and the FPPW was 0.08% with a 17.3% miss rate, as shown in Fig 9.

Fig 13 The detector continued to detect pedestrians even with rain drops partially obscuring the view

6 Conclusions and Future Work

We have demonstrated pedestrian following on a mobile robot using HOG features. The system uses monocular video for detections, stereo depth data for distance, and runs at about 8 Hz on board the robot using 70% of the 1.2 GHz Intel Core 2 Duo. The system is able to follow people at typical walking speeds over flat and moderate terrain.

Unlike some previous work, our follower does not need to learn or be calibrated to follow a new pedestrian. The system uses only vision and does not use depth data for detection. Cameras represent an inexpensive path to deployment and provide rich features not available with range sensors. Our evaluations verify the system's performance and suggest a level of detector performance necessary for person following in our system. Finally, we have successfully demonstrated performance over a significant distance with a variety of targets.

Our next step will be to implement a multiple target tracker. At times, the pedestrian detector will produce persistent detections on other targets (other pedestrians) or false targets (non-pedestrians). For example, if two pedestrians are in the scene, the detector will probably correctly locate both people. On the other hand, occasionally a tree or bush will generate a relatively stable false target. Currently, the tracker must resolve all the detections into a single target; the result is that multiple targets get averaged together. These kinds of errors can be mitigated with a multiple target tracker, such as those used in [15-17].

Acknowledgements. This work was partially supported by DARPA Contract W31P4Q-08-C-0327 and ONR Contract N00014-08-C-0626.

References

1. Satake J, Miura J (2009) Robust Stereo-Based Person Detection and Tracking for a Person Following Robot. In: Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking
2. Loper M, Koenig N, Chernova S, Jenkins C, Jones C (2009) Mobile human-robot teaming with environmental tolerance. In: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction, pp 157-164
3. Kwon H, Yoon Y, Park J, Kak A (2005) Person Tracking with a Mobile Robot using Two Uncalibrated Independently Moving Cameras. In: Proceedings of IEEE International Conference on Robotics and Automation

4. Kleinehagenbrock M, Lang S, Fritsch J, Lomker F, Fink G, Sagerer G (2002) Person Tracking with a Mobile Robot based on Multi-Modal Anchoring. In: Proceedings of IEEE International Workshop on Robot and Human Interactive Communacation, pp 423-429
5. Carballo A, Ohya A, Yuta S (2009) Multiple people detection from a mobile robot using double layered laser range finders. In: Lecture Notes in Electrical Engineering 35, DOI 10.1007/978-3-540-89859-7 22, Springer-Verlag Berlin Heidelberg
6. Yoshimi T, Nishiyama M, Sonoura T, Nakamoto H, Tokura S et al (2006) Development of a Person Following Robot with Vision Based Target Detection. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems
7. Shaker S, Saade J, Asmar D (2008) Fuzzy Inference-Based Person-Following Robot. International Journal of Systems Applications, Engineering and Development, Issue 1, Volume 2
8. Kirby R, Forlizzi J, Simmons R (2007) Natural Person-Following Behavior for Social Robots. In: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction, pp 17-24
9. Topp E, Christensen H (2005) Tracking for following and passing persons. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 70-76
10. Schlegel C, Illmann J, Jaberg K, Schuster M, Worz R (1998) Vision based person tracking with a mobile robot. In: Proceedings of the Ninth British Machine Vision Conference, pp 418-427
11. Tarokh M, Kuo J (2006) Vision based Person tracking and following in unstructured environments. In: Proceedings of IEEE 13th Annual Conf. on Mechatronics and Machine Vision in Practice: Vol. 1., pp 031.1-031.6
12. Zhu Q, Yeh M, Cheng K, Avidan S (2006) Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol 2, No 2, pp 1491-1498
13. Dalal N (2006) Finding People in Images and Videos. Dissertation, Institut National Polytechnique de Grenoble
14. Dalal N, Triggs B (2005) Histograms of Oriented Gradients for Human Detection. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition, pp 886-893
15. Ess A, Leibe B, Schindler K, van Gool L (2008) A Mobile Vision System for Robust Multi-Person Tracking. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition
16. Lau B, Arras K, Burgard W (2009) Multi-model Hypothesis Group Tracking and Group Size Estimation. In: Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking
17. Sullivan J, Nillius P, Carlsson S (2009) Multi-target Tracking on a Large Scale: Experiences from Football Player Tracking. In: Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking

18. Viola P, Jones M (2001) Rapid Object Detection using a Boosted Cascade of Simple Features. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Vol 1, pp 511
19. Needham C, Boyle R (2003) Performance Evaluation Metrics and Statistics for Positional Tracker Evaluation. In: Proceedings of International Conference on Computer Vision Systems, pp 278-289
20. Särkkä S, Vehtari A, Lampinen J (2007) Rao-Blackwellized Particle Filter for Multiple Target Tracking. In: Information Fusion Journal, Vol 8, Issue 1, pp 2-15
21. Schapire R (2001) The boosting approach to machine learning: An overview. In: MSRI Workshop on Nonlinear Estimation and Classification
22. National Oceanic and Atmospheric Administration (2009) Current weather conditions: Bedford Hanscom Field, MA, United States. http://weather.noaa.gov/weather/current/KBED.html. Accessed 21 July 2009
23. iRobot Research Videos. http://www.irobot.com/sp.cfm?pageid=189. “iRobot Tactical Teams / Natural HRI”

Biographies

Jonathan Brookshire obtained his M.S. in Robotics from Carnegie Mellon University (CMU) in 2004 and his Electrical Engineering B.S. from the University of Virginia in 2002. He is currently pursuing his PhD at MIT and is also a researcher at iRobot Corporation, investigating methods of human robot interaction on small, mobile platforms. At iRobot, he developed a complete, integrated system that is now a commercial product: the Mapping Kit provides the operator with fused information that provides (1) real-time 2-D mapping and (2) safeguarded teleoperation capabilities. At CMU, Mr. Brookshire studied sliding autonomy with teams of multiple, heterogeneous robots performing a coordinated assembly task. He also previously worked at MIT Lincoln Laboratory focused on RADAR real-time image processing and high speed data acquisition.