On Leader Following and Classification

Procópio Stein, Anne Spalanzani, Vitor Santos, Christian Laugier

▶ **To cite this version:**

Procópio Stein, Anne Spalanzani, Vitor Santos, Christian Laugier. On Leader Following and Classification. IROS 2014, Sep 2014, Chicago, United States. hal-01069567v2

HAL Id: hal-01069567

https://inria.hal.science/hal-01069567v2

Submitted on 12 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

On Leader Following and Classification

Procópio Stein[1], Anne Spalanzani[1,2], Vítor Santos [3] and Christian Laugier[1]

Abstract— **Service and assistance robots that must move in human environment must address the difficult issue of navigating in dynamic environments. As it has been shown in previous works, in such situations the robots can take advantage of the motion of persons by following them, managing to move together with humans in difficult situations. In those circumstances, the problem to be solved is how to choose a human leader to be followed.**

This work proposes an innovative method for leader selection, based on human experience. A learning framework is developed, where data is acquired, labeled and then used to train an AdaBoost classification algorithm, to determine if a candidate is a bad or a good leader, and also to study the contribution of features to the classification process.

I. Introduction

Mobile robots that provide assistance or services to humans must face a series of requirements that set them apart from other types of robots. They are expected to be able to interact with humans, adapt to unpredicted situations, respect social conventions and be able to move in dynamic environments. These requirements are directly related to the success of interactions and human acceptance of service and assistance robots.

To address the issue of robot motion planning in populated environments, modern techniques implement methods based on probabilistic and predictive approaches [1], [2] and on models of social interactions [3]. These techniques create motion plans that allow the robot to avoid trajectories that have a risk of future collision with persons, while at the same time avoiding entering personal and social spaces and causing discomfort to the persons involved.

A drawback of those approaches is that they usually do not take into account changes that people perform in their typical paths to avoid and adapt to other moving persons. This omission, allied to excessive future uncertainty, or densely populated environments, may lead to situations where every generated path leads to collisions, resulting in frozen situations, as shown by [4].

However, humans are able to seamlessly move in populated environments and also to address complex situations and interactions with other humans. The way humans move is the result of information gathering and very complex decision making processes, which is not yet completely understood, although some models have been developed [5] and incorporated in planning algorithms [6], [7].

[1] INRIA Rhône Alpes, France {procopio.silveira-stein, anne.spalanzani, christian.laugier}@inria.fr
[2] Univ. Grenoble Alpes, Grenoble, France
[3] Universidade de Aveiro, Portugal vitor@ua.pt

Fig. 1. Initial configuration of the experiment using a pedestrian simulator. Pedestrians are represented by circles and must reach opposing ends in the corridor. The robot is represented by the dark rectangle and its goal lies on the right region of the corridor.

Based on these observations, it has been proposed that the robot can enhance its navigation capabilities by following humans in such environments, taking advantage of their navigation skills [8], [9], [10]. In these works, a probabilistic framework was used to predict the goal of leader candidates, and then the robot would select as a leader the person moving to the same destination as itself.

Following this line of research, the current work extends the previous studies, creating a learning framework for leader classification, according to their behavior and based on human experience.

The next section shows the advantages of leader following behavior over classic and state-of-the-art motion planning algorithms in a populated corridor. Section III follows with a description on the AdaBoost algorithm and how it can be used to study the contribution of features to the classification problem. After that, Section IV describes the data acquisition and labeling, followed by the training of the classifier and experiments. The analysis and conclusions of this work are presented in section V.

II. Advantages of Leader Following

This experiment uses a simple pedestrian simulator, which was developed based on the social-forces model [11]. The test environment consists on a narrow corridor with two groups of nine agents that have to reach opposing goals in such corridor. A robot that must reach a goal in the right-most region of the test scenario starts behind the group in the left region. Figure 1 depicts the initial configuration of the experiment. On the following images, the complete path traveled by the robot is represented as a dark red line.

In the first test, the A$*$ algorithm is used. As it has been developed to work in a static mode, its planning phase takes place at snapshots of the current environment state. However, due to the motion of the agents, the motion plan is frequently invalidated, prompting the algorithm to constantly replan. This results in a path where the robot changes direction several times, as shown by the dark red line in Figure 2. The constant need of replanning virtually brings the robot to a halt during some instants. Only when the density of

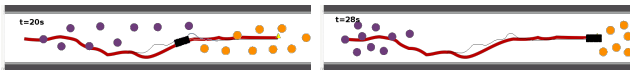

Fig. 2. Navigation using A* motion planning. Due to the motion of agents, the algorithm is constantly replanning, hindering the robot motion.

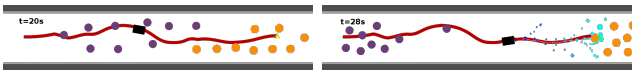

Fig. 3. Navigation using RiskRRT motion planning. The motion prediction uncertainty allied with the high density of agents produces the Freezing Robot Problem, where the algorithm predicts a collision for any possible motion, bringing the robot to a halt.

agents reduces, the algorithm manages to create a plan that the robot is able to follow, until its goal.

The second experiments uses a state-of-the-art algorithm, specially developed for motion planning in dynamic environments called RiskRRT [1], that uses motion prediction to avoid collision with agents. Even though the RiskRRT has a promising performance in sparsely populated environments, as the density of agents grows, this techniques falls into the Freezing Robot Problem (FRP). At some point, the predicted future states of the agents cover all the region in front of the robot, which causes the algorithm to fail to find a solution, delaying its motion, as shown in Figure 3. This is a recurrent problem in prediction-based motion planning algorithms.

Finally, the last experiment avoids using motion planning algorithms and instead uses a set of simple rules to select an agent that will lead the robot in difficult situations (for more details on this approach the reader is referred to [10]). In this way, the robot takes advantage of complex interactions among the surrounding agents (Figure 4), resulting in a smoother and faster path, reducing the time spent to cross the corridor.

Comparing the three techniques, simulations using both the leader following and Dijkstra's algorithm spend a similar amount of time to reach the goal (28 s), but the leader following algorithm provides a smoother path and also respect the social spaces of the agents. The Dijkstra's approach, in the other hand, completely disregards the presence of moving agents, and the only reason it has a similar performance in terms of time spent is that agents reacted to the presence of the robot and gave room for it to pass. Although the RiskRRT approach also respects the social spaces, it is slower to reach the goal and also results in a path with a larger number of detours and oscillations. These tests show how a simple technique is able to have better results in this scenario, resulting in a robot motion that is fast, smooth and respectful toward moving agents.

III. Classification Algorithm

Once shown the benefits of leader following, it is interesting to study what are the underlying criteria used by persons to engage and disengage in leader following behavior, so this can also be applied in other research areas.

Here a technique is proposed in order to capture the individual cues that can be associated with a good or a

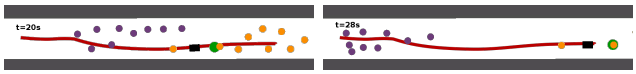

Fig. 4. Navigation using leader following technique. The robot selects a leader according to a set of rules and engage in a group formation, benefiting from the complex interactions of leading agents.

bad leader. The objective is to have a system that can receive different types of measurements (features) of leader candidates and classify them as good or bad leaders, based on examples

It is expected that this approach bring benefits to the leader selection problem, as:

- it is possible to provide individual scores for leader candidates, allowing a better basis for the decision of who to follow;
- a bad leader can be associated with situations of discomfort experienced by the subjects being followed, so the algorithm can stop following a leader, improving its acceptance.

Besides the objective of classifying a good or bad leader candidate, it is also important to understand what are the measurements, or features, that are relevant to that classification (distance, velocity, etc.). Because of this, the machine learning algorithm chosen is the AdaBoost, as this algorithm inherently selects the features that contribute the most to the classification process. This choice addresses two issues: the leader classification problem and the study of what are the features that are most important to that process.

A. AdaBoost Classifier

Boosting is the name of a category of supervised machine learning ensemble algorithm, where several *weak classifiers*, or rule-of-thumbs, are combined in order to create a more accurate predictor, or *strong classifier*. The adaptive Boosting, or AdaBoost [12], is a variation of the basic Boosting algorithm where more emphasis is given in the correct classification of previously misclassified samples, adapting the weights of samples in the training set.

In the algorithm workflow, the confidence parameter α, is responsible for the adaptive capability of the algorithm. This parameter is used in the update of the weights of the training set, and its value is based on the classification error ϵ:

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$$

If a training sample is correctly classified, its weight is reduced, whereas if the classification fails, the weight of this sample is increased. The new weight distribution will now affect the choice of the *weak classifier* on the next iteration, favoring one that manages to correctly classify the difficult cases.

Among the advantages of the AdaBoost algorithm is the exponential convergence of the training error to zero and good generalization properties, besides the capacity of the algorithm to perform feature selection.

B. Features Study

According to [13], AdaBoost has a feature selection built into it when using decision stumps as weak learners. As in each iteration, a weak classifier is associated with only one feature, the contribution parameter α and error ϵ are measurements of the importance of that feature in the classification problem. This provides a form of quantitative evaluation that can be used as a criteria for feature selection.

It also let us draw conclusions about why some features are more important than others, as this can help the understanding of the criteria used by humans in following behaviors.

According to [14] and [15], the contribution of each feature in the general classification problem can be measured by summing the number of times each feature is used, weighted by the α_t of the weak classifier that uses the feature in question. Different from the aforementioned works, here this sum will also be normalized by the sum of all α_t, to obtain the ratio of the contribution of each feature.

$$C(f_i) = \frac{\sum_{t=1}^{T} |\alpha_t| \delta[F(h_t) - f_i]}{\sum_{t=1}^{T} |\alpha_t|}$$

$$\text{where} \quad \delta[n] = \begin{cases} 0, & n = 0 \\ 1, & n \neq 0 \end{cases}$$

Where $C(f_i)$ is the contribute ratio of feature $f_i \in \{1, 2, 3, ..., i\}$, α_t is the confidence parameter and $F(h_t)$ is a function that returns the identification (number) of the feature chosen by the weak learner h_t. Finally, δ is the Kronecker delta, which outputs 1 in the case the weak learner h_t was created using feature f_i and 0 otherwise. As a result, $C(f_i)$ will be the weighted sum of the number of times that the feature of interest f_i has been used by a weak learner. And this will be the measurement used to evaluate features contribution.

IV. Experiments

This section will present the experiments associated with the methods described in Section III. Tests will begin with data acquisition and labeling process. After that, the classification capabilities of the algorithm will be evaluated, together with the study of the most relevant features used in the classification process. All tests were performed using the modular architecture of Robot Operating System (ROS) [16].

A. Data Acquisition

Data collection was performed with a small car-like robot (Fig. 5), measuring approximately $0.4 \times 0.75m$. A laser scanning range finder was installed on the robot, providing distance measurements up to $30m$ and with a field of view of 270^o. Also a firewire camera fitted with a wide-angle lens was present, providing a way to acquire images with a large field of view. In this way, videos could be taken during tests and then associated with the laser scans, as shown in Fig. 6. This allowed a better understanding of the experiment situations, together with precise measurements of the surrounding area.

Fig. 5. The robot used for data acquisition when following pedestrians.

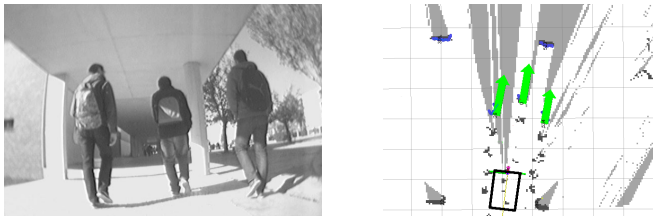

Fig. 6. (a) Image from a wide-angle camera. (b) Representation of the laser measurements, showing the three tracked subjects and their orientation as the green vectors, the robot is represented as the black rectangle.

The robot was teleoperated by a person that was hidden from the subjects being followed, in order to create the illusion that the robot was autonomous.

In total 47 tests of the robot following persons or group of persons were recorded, with the mean duration of about 20 seconds each. Tests were conducted in an open corridor, about $3m$ wide, that links several University buildings. The objective of the robot operator was to position the robot behind the pedestrians and try to maintain the a constant speed and distance from the person being followed. In the case the leader stopped, moved aside or was too fast, the operator should resume a stand-alone navigation, simulating the situation where an algorithm would abandon a leader.

In the setup used for data acquisition a module receives the laser range finder measurements and another the camera images. Readings from both sensors are stored in a ROS *bag* file. Isolated from these modules, there is also a program that receives commands from a gamepad and passes them to a module responsible for a low level interface with the robot's motors.

B. Extraction of Features

Only the laser range finder measurements were used to obtain the descriptors of each leader subject. Initially, the *bag* containing measurements from the test have its messages published to another module that clusters and tracks moving objects, using the motion tracker developed by [17].

As the laser measurements are obtained from the robot's point of view (robot's frame), static objects will appear as moving ones, as the robot moves. To account for that, the transformation of the robot frame w.r.t. the global frame (or map) must be computed, as this will allow the laser measurements to be transformed to the global frame. The

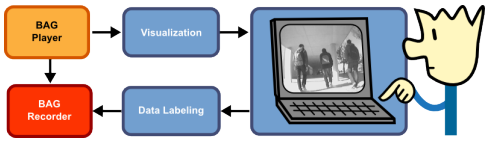

Fig. 7. Diagram of the data labeling process.

task of finding the transformation *global frame* $\rightarrow$ *robot*, is accomplished using the Hector SLAM tool [18], which is able to localize the robot in an environment based only on laser measurements.

With the laser measurements transformed to the global frame, a motion tracker [17] is used to manage the identification and tracking of moving targets. This algorithm also implements a Kalman Filter (KF), to cope with temporary obstructions of tracked humans.

After this, another module proceeds to the feature extraction of moving targets, which are then stored in a *.txt* file that will be further used during the training process.

- candidate's absolute velocity v_{c_i};
- relative heading b/t. the robot and the candidate α_{rc_i};
- candidate's relative velocity w.r.t. the robot reference frame vx_{rc_i} and vy_{rc_i};
- angle between the robot heading and the candidate's position β_{c_i};
- distance between the candidate and the robot d_{c_i};
- lateral and sagittal displacement of the candidate ld_{c_i} and sd_{c_i} ;

As the tracker implements a KF, all the measurements incorporate a temporal dimension, as the filter uses previous states to compute the current one. Besides the filtered features, their derivatives and standard deviations were also computed. The total number of features extracted is 24.

C. Data Labeling

In many data labeling situations, the decision to create a label on the dataset is clear and objective. Examples are faces in images, or pedestrians in laser scans. However, this is not the case in this work. Here the intention is to capture how humans decide when to start or stop following someone, or in other words, when someone is a good or bad leader. In order to do so, participants must create labels based on their *feeling* about someone being a good leader or not.

The labeling process also needs to be simple, as the input of persons that were unfamiliar with the process was required. This lead to a decision of creating a binary labeling system, with candidates identified either as good or bad leader. Besides that, for the sake of simplicity, the data presented to the volunteers should always have a single good/bad leader transition.

In the labeling process, depicted in Fig. 7, the volunteers should press a button whenever they *felt* a transition from good to bad leader occurred, while watching a video of persons being followed by the robot. A ROS module captured the keyboard input that gets recorded, together with all the information from the original bag file, to a new one.

Before the labeling began, examples of different classes of good and bad leader situations were shown to the volunteers. Later they were asked to tell which of these situations occurred in each experiment. Although each test has its own peculiarities, the experiments can roughly fit one of the following classes:

- good leader (gd) - leader(s) maintained their speed and orientation, withouth changing their behavior while being followed;
- bad leader, moved aside (as) - identifies situations where leaders gave room for the robot to pass, generally moving aside, while keeping their original motion direction;
- bad leader, far or fast (fr) - this occurred whenever the distance between the leader and the robot grew to a point where it was not advantageous to keep following them;
- bad leader, stopped (st) - when the person being followed stops moving;

Three persons participated in this process, and the final label was computed as the mean time of the three labels instants. In the case that labels were not close to each other (timewise), only the two closest were used. This analysis was aided by the typical situation identified by the labelers. For example, if one had found that the transition occurred because the leader was too far, but the other two thought that the transition was due to the leader moving aside, only the label of the latter two was used, as they agreed in the situation.

Besides the four typical situations mentioned before, two other situations were recorded, which did not required manual labeling. These situations refer to cases when candidates were not moving, but rather standing close to the robot's path (nm), and when persons were moving to the opposite direction compared to the robot's (od).

In total, 12911 samples were obtained, with the proportion of 37% of bad leader labels and 63% of good leader labels. This is equivalent to 451 seconds of tests, divided in 47 experiments and distributed in the following classes: 9 good leaders; 7 leader moved aside; 5 leader too far or fast; 9 leader stopped; 7 candidate not moving; 10 candidate coming from opposite direction.

D. Training

To evaluate the classification, two measurements were made. The **false good leader**, where the classifier labeled a sample as *good leader* but the ground truth has a *bad leader* label. This is the most critical error, because by following someone that should not be followed, the robot may be disturbing the leader or find itself in unwanted situations. The second measurement is the **false bad leader**, that occurred when the classifier output a *bad leader* and the ground truth is labeled as *good leader*. Although this is also a mismatch in the classification, it is much less critical then the previous measurement, because if a robot does not follow a potential leader, it is only loosing an opportunity.

To create a dataset to evaluate the performance of the classifiers, two datasets (except only one far/fast) of each

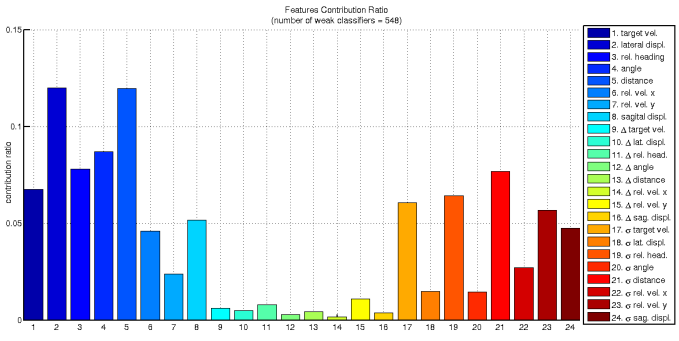

Fig. 8. Feature contribution ratio using 24 features, number of weak learners: 548.

TABLE I
CONTRIBUTION RATIO OF FEATURES

	size of feature set			
features	24	16	12	8
σ distance	0.077	**0.085**	**0.104**	**0.181**
lateral displacement	**0.120**	**0.130**	**0.118**	**0.169**
distance	**0.120**	**0.131**	**0.125**	**0.122**
angle to robot	**0.087**	0.080	0.090	0.119
sagittal displacement	0.052	0.053	0.078	0.118
relative heading	0.078	0.083	0.081	0.113
target velocity	0.068	0.068	0.082	0.093
σ relative velocity in y	0.057	0.062	0.075	0.085

situation were randomly chosen, making a total of 2715 samples with 39% cases of bad leader. The remaining sets were grouped into the training dataset, totaling 8504 samples with 34% cases of bad leader.

During the Adaboost training, there was no limitation in the number of the weak learners used (iterations), and the stop criteria was the error. In all cases, it reached 0 after different numbers of iterations.

E. Results in Classification Error and Feature Selection

The objectives of these tests were to evaluate the how different sets of features affected the train and performance of the resulting classifiers. After each test, the features that provided the less contribution were removed, and a new test was conducted. In this way it was possible to determine if the contribution of the most important features were maintained, and how that affected the classification error. This error was computed comparing the output of the classifier with the labeled ground truth. In the first test, all the 24 features were used, and their contribution result is shown in Fig. 8.

In this image it is possible to distinguish the three groups of features used in the experiments: the directly measured features, their derivatives and their standard deviation. It is clear that the derivatives group have almost no role at all in the classification, with individual contribution ratios usually smaller than 1%. A possible explanation for this is that this operation added noise into the data, reducing its usefulness. Looking at the first group (directly measured features), the most important features are clearly the lateral displacement and the distance between the robot and the leader, while the less important ones being the relative velocities, specially along the y axis. Finally focusing on the last group, the most important features were the standard deviation of the distance, of the relative heading and of the target velocity.

Following the investigation of the most important features for this classification problem other tests were conducted, using the same training sets, but with a smaller number of features. In each test, the less important features were removed and then the AdaBoost algorithm was retrained and evaluated again. This resulted in three other tests, using 16, 12 and 8 features. Table I shows the final 8 most important features, and how their contribution ratio evolved from previous tests.

According to these results, the **standard deviation of the distance between the robot and the leader** and the **lateral displacement of the leader** are the two most important features in the leader classification. Together they contribution ratio is approximately 35% when using only the eight most important features. Parallel to the investigation of the most relevant features for the leader classification, after each training of the AdaBoost classifier, its performance was evaluated regarding its classification error, which was divided according to the distinct leader situations, for a better analysis.

The results of these tests are shown in Table II, according to the size of the feature set used. Regarding the performance of the classifier in each situation, the *false good leader* relative error is particularly large at the *move aside* situations (*as I* and *as II*). However, the error was much smaller in the remaining situations, and the overall performance of the classifier was remarkable, with the total relative error across all the situations in the order of 3%. Regarding the results in the light of the size of the features set, it can be seen that although the results are very similar, the smaller feature set of size 8 had the best performance in the critical classification of a good leader. Even though its performance on the misclassification of bad leaders was worst than when using larger features sets, that was not a critical error. In conclusion, the classifier trained with the smaller feature set had the most promising results.

V. CONCLUSIONS

In this work, a leader classification system based on a learning framework was presented. A machine learning algorithm was trained using real data, that was labeled by humans. The use of the AdaBoost algorithm allowed the study of the contribution of the features used to train the classifier, giving an insight about how humans decide who is a good or bad leader.

The tests provided very interesting results about the importance of the lateral displacement and of the distance to the leader (and its standard deviation), which were the features that contributed most to the classification task, across different numbers of features used.

TABLE II
RELATIVE ERROR IN LEADER CLASSIFICATION

		leader situations											
	features	st I	st II	gd I	gd II	as I	as II	fr I	nm I	nm II	od I	od II	total
false good	24	0.062	0.000	0.000	0.000	0.098	0.145	0.012	0.066	0.000	0.000	0.000	0.036
	16	0.062	0.000	0.000	0.000	0.098	0.145	**0.005**	0.044	0.000	0.000	0.000	0.033
	12	0.062	0.000	0.000	0.000	0.071	0.145	**0.005**	**0.017**	0.000	0.000	0.000	0.028
	8	**0.046**	0.000	0.000	0.000	**0.047**	0.145	**0.005**	0.039	0.000	0.000	0.000	**0.026**
false bad	24	**0.000**	**0.110**	**0.000**	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.011**
	16	**0.000**	0.120	**0.000**	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.011**
	12	**0.000**	0.124	**0.000**	**0.004**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.011**
	8	0.069	0.129	0.014	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.015

Both the lateral displacement and the distance are spatial features, that do not take into account the dynamics of the leader. This means that the position of someone w.r.t. the robot plays a very important role when deciding among leader candidates and deciding when to stop following someone.

One interpretation of the importance of the standard deviation of the distance is that it represents the stability of the distance between the leader and the robot during a test, while the lateral displacement of the leader may be related to a human giving room for the robot to pass.

Several classifiers were trained, using different numbers of features. In overall, the trained classifiers produced good results, with the total relative error smaller than 5%, and the total error for false good leader classifications, which is the most critical error, smaller than 4%, according to the training set and the number of features used.

Looking individually at the different situations, the worst performance of the classifier was in the *move aside* situation. This was a difficult condition, that the classifier could not properly identify with the given features. This issue requires further investigation, specially concerning the exploration of a richer feature set, using different sensors as cameras and higher level informations, as features extracted based on a pre-planned robot path, for example.

Finally, there was no significant differences among the classifiers while using different number of features. Although promising, these results are obtained from a sum of instant classifications, and they can only be properly evaluated when integrated with a navigation strategy and higher level layers of reasoning, which will be conducted in the future works.

REFERENCES

[1] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems*, Nice, France, 2008.

[2] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research*, vol. 24, no. 1, p. 31, 2005.

[3] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2011, pp. 2014–2019.

[4] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2010, pp. 797–803.

[5] K. Mombaur, A. Truong, and J. Laumond, "From human to humanoid locomotion - an inverse optimal control approach," *Autonomous Robots*, vol. 28, pp. 369–383, Dec. 2009. [Online]. Available: http://www.springerlink.com/content/c571474575ux1t52/

[6] D. Althoff, D. Wollherr, and M. Buss, "Safety assessment of trajectories for navigation in uncertain and dynamic environments," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 5407–5412.

[7] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2010, pp. 981–986.

[8] P. Stein, A. Spalanzani, C. Laugier, and V. Santos, "Leader selection and following in dynamic environments," in *2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*, 2012, pp. 124–129.

[9] P. Stein, A. Spalanzani, V. Santos, and C. Laugier, "Robot navigation taking advantage of moving agents," in *IROS 2012 Workshop on Assistance and Service robotics in a human environment*, 2012.

[10] P. Stein, V. Santos, A. Spalanzani, and C. Laugier, "Navigating in populated environments by following a leader," in *International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2013.

[11] D. Helbing, I. J. Farkas, P. Molnar, and T. Vicsek, "Simulation of pedestrian crowds in normal and evacuation situations," in *Pedestrian and Evacuation Dynamics*, S. D. Sharma and M. Schreckenberg, Eds. Springer Berlin, 2002, pp. 21–58.

[12] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002200009791504X

[13] T. Susnjak, "Accelerating classifier training using AdaBoost within cascades of boosted ensembles," Science in Computer Sciences, Massey University, Auckland, New Zealand, 2009. [Online]. Available: http://muir.massey.ac.nz/handle/10179/1002

[14] M. Tsuchiya and H. Fujiyoshi, "Evaluating feature importance for object classification in visual surveillance," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 2, 2006, pp. 978–981.

[15] M. Drauschke, "Feature subset selection with adaboost and adtboost," Tech. rept. Department of Photogrammetry, University of Bonn, Tech. Rep., 2008.

[16] M. Quigley, B. Gerkeyy, K. Conleyy, J. Fausty, T. Footey, J. Leibsz, E. Bergery, R. Wheelery, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[17] J. Almeida, "Target tracking using laser range finder with occlusion," Master's thesis, Universidade de Aveiro, Aveiro, Portugal, 2010.

[18] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.