# Human-Aware Mobile Robot Exploration and Motion Planner

Fei Xia, Louae Tyoan, Zhongtao Yang, Ikenna Uzoije, Linze Zhao,
Guangcong Zhang and Patricio Antonio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA
pvela@gatech.edu

***Abstract*—Conventionally, robot exploration problem assumes a static environment. However, casting the exploration problem with the presence of humans brings new possibilities for interactive exploration tasks including robot rescue, etc. In this paper, we present a mobile robot system designed to explore an unknown environment with humans. The system is able to perform the robot localization and mapping, and produce the positions of humans in the environment at the same time. The system combines techniques of simultaneous localization and mapping (SLAM), motion planning, and human recognition. Our approach maximizes the exploration efficiency by designing optimal exploration paths. Moreover, the robot is able to perform human-aware behavior during the exploration, which complies with human safety, preferences and actions. Both simulation and field experiment results in real world settings are presented.**

***Keywords*—Human-aware exploration, autonomous navigation, human-robot interaction**

## I. Introduction

Navigation and exploration are highly autonomous behaviors of robots. A variety of efforts have been made in this area. Particularly, the interdisciplinary area of robot navigation and human-robot interaction has seen great progress in recent years. Some human-aware robot navigation and motion planning methods [1] [2] [3] [4] [5] [6] have been proposed. Most human-aware navigation papers take human's presence into consideration, for example, the path planning considers constraints of safety and human comfort. Among them, [1] and [3] are two representative works. [1] summarises previous work related to human-aware navigation. It discussed different approaches including use case based methods and cost function based methods. However, they are restricted to navigation tasks without mentioning human-aware explorations. [3] demonstrates a sampling based costmap planner. However, it also assumes a known workspace so this method can be used only for navigation but not for exploration.

On the other hand, robot exploration is a popular topic in the robot community and has various applications. Apart from traditional frontier exploration methods [7], some researches focus on collaborative robot explorations [8]. Those exploration algorithms are used for searching in a room and building a static map.

The motivation of this research is to combine human-aware techniques with robot exploration for real-world applications.

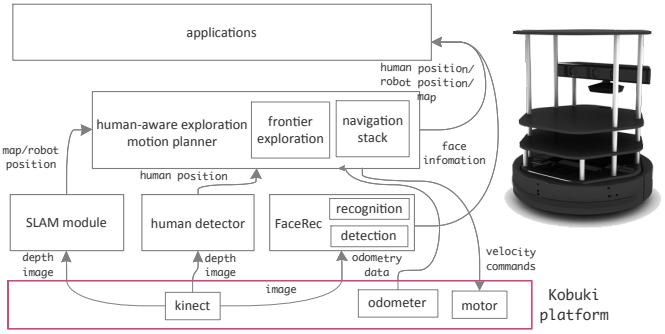



Fig. 1: Architecture of TurtleSeek system.

Such an exploration system is important because the robots often have to work with the presence of human beings in real applications. The exploration system can avoid collision with human while perform favorable path planning. Also the mapping result is with not only static scene points but also semantic information of humans. Such a system remains a knowledge gap in the literature, and thus we propose a mobile robot system that can explore an area with the presence of humans.

The rest of this paper is organized as follows: Section II describes the architecture of the whole system; Section III shows the human-aware techniques integrated in the system; Section IV focuses on the navigation and exploration components; Section V presents simulation and real scene experiments of our system, and Section VI concludes the work and discusses the future work for this project.

## II. System Architecture

We leverage open-source solutions as the infrastructure of our system. Our system is built on the Turtlebot [9] with Robot Operating System (ROS) [10]. Our system mainly includes five modules: human detection module, face detection and recognition module, SLAM module, navigation module, and exploration motion planner module. Our system is dubbed as **TurtleSeek**, indicating it is built on Turtlebot and can *seek* for humans.

Figure 1 illustrates the architecture and data flow of the system.

The Turtlebot is based on Kobuki platform by Yujin Robot, which provides odometer, motors and sensors including Kinect. This is shown in the rectangle in Figure 1. The small rectangles in that red one stand for hardware driver nodes that serve as interfaces from the hardware to our system.

Kinect provides data for the human detection module to estimate the position of a human and for the SLAM module for mapping and localization. The position of humans, SLAM results, as well as the laser scan data are used to do the path planning. Then velocity commands are sent to the platform to drive robot following the designed path. Lastly, our system is equipped with face recognition module, which makes it able to recognize people.

As for implementing the modules, we build each module on ROS. Each module is implemented as an ROS *package*. Each package has at least one or more *nodes*, which are processes that perform computation. Nodes communicate with each other by passing *messages* under a given *topic*. A *message* is data structure of a certain type. For example the kinect driver node publishes the depth image to the topic '\depth\image_raw', and the SLAM node subscribes that topic. Then the depth image message is passed from the former node to the latter node. The rest of the system in Firgure 1 is connected using the similar protocol.

## III. Human-aware techniques

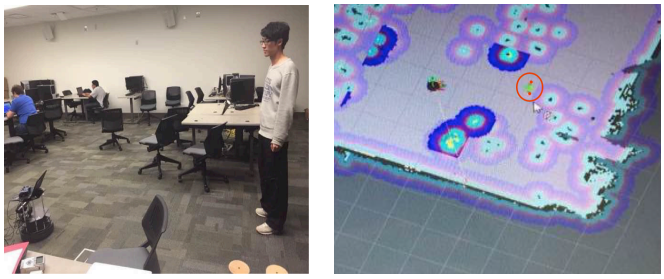
### A. human detection

Considering the height of the robot, human detection of our system is achieved by leg detection. This component is implemented with the leg detector from the ROS package *people* [11]. The laser scans are provided by the Kinect by extracting the rows at the bottom of depth image. The leg detector takes laser scans as input and uses a random forest trained classifier to detect groups of readings as possible legs.

The readings of laser are clustered into clusters. The training and testing samples are sets of points of those clusters. For each set of points, the following features are considered.

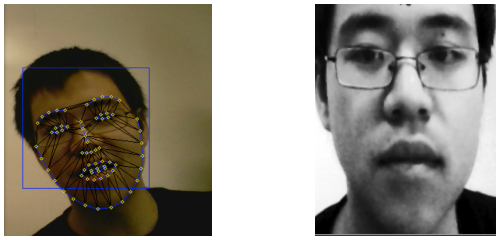
- *Statistics related features*: The standard deviation and median absolute deviation of the data points are selected as statistics related features.
- *Jump related features*: The distance from selected cluster to previous cluster and next cluster, and the width of the selected cluster are selected as jump related features.
- *Linearity and circularity related features*: We fit the data points with line and arc. Then we get the RMSE (rooted mean square error) of the fitting process as linearity and circularity features. The radius of the arc is selected as another circularity related feature.
- *Boundary related features*: We get the boundary of the data points. Length of the boundary and average curvature of the boundary are selected as features.

The classifier is learned by using random forest. We use the implementation from the machine learning library of *OpenCV*.



(a) Real scene of human detection. (b) Output of the module.

Fig. 2: Example of our human detection module. Left: in the real scene, a human is standing in front of the robot. Right: In the output, red dots indicate legs, the green dot is the estimation of the center of human. The background is the cost map of other obstacles.



(a) Output of face detection and alignment. (b) Result after normalization and correction.

Fig. 3: Example output of the face detection and recognition module.

The detection algorithm can also pair individual legs together by finding pairs of legs within the max distance, and estimate the center of one person.

An example of our human detection module is shown in Figure 2.

### B. face detection and recognition

The face detection and recognition module is implemented as a subsystem called *FaceRec*. It performs face detection, face alignment, training set building, and finally face recognition.

The face detection is implemented using AdaBoost learning algorithm [12]. After this step, the rough position of the face is found. With the position, face alignment is conducted via Subspace Constrained Mean-Shifts [13]. In aligned faces, a bilinear transformation is used to correct the skew and rotation of the faces. The face data after correction is added to a face archive. Figure 3 shows the aligned faces, where the rotation of the face was corrected.

Finally, sparse representations [14] are used for the face recognition. The accuracy in real scene is comparable to the state of the art algorithms.

## IV. Navigation and Exploration

### A. Navigation and exploration framework

The navigation system is built upon the built-in navigation stack [15] of Turtlebot. The exploration algorithm used is frontier based algorithm [7], with implementation of ROS frontier exploration package [16]

### B. Human-aware exploration

To make the exploration human-aware, we first determine the position and orientation of human; then the position and orientation are used for the *costmap* to meet the safety and comfortable criteria, which in turn adjusts the exploration.

Denote the position of a human as $(x_h, y_h)$, the position of robot as $(x_r, y_r)$, and the orientation of the robot as $\theta_r$. We define the safety criteria and comfortable(visibility) criteria by the costs of safety $C_s$ and the cost of comfort $C_c$:

*a) Safety criteria:* The safety criteria mainly prevent the robot to come too close to human. The cost is defined as

$$C_s = \frac{1}{||(x_h, y_h) - (x_r, y_r)||^2} \tag{1}$$

*b) Comfortable criteria:* The comfortable criteria mainly consider whether the robot is in the vision. If the robot is not in the vision. human will feel more comfortable if the robot is not too close to him.

$$C_c = \frac{1}{|\text{atan2}(y_h - y_r, x_h - x_r) - \theta_r|^2/\pi^2 + \beta}, \tag{2}$$

where $\beta > 0$ is a controlling factor. We use atan2 to obtain the relative rotation of robot with respect to human. In the experiments, $\beta$ is set to 1 in order to make the $C_c \in [\frac{1}{2}, 1]$.

*c) Final criteria:* The overall cost is defined by the linear combination of the previous two costs along with the cost from other scene obstacles:

$$C = C_o + \lambda_1 C_s + \lambda_2 C_c, \tag{3}$$

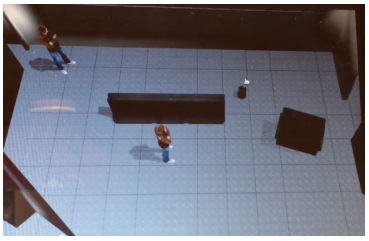
where $C_o$ is the cost generated by other obstacles; $\lambda_1$ and $\lambda_2$ are parameters to control the weights of costs. This overall cost is used in the exploration. In the experiments, the parameters are chosen empirically. Without losing generality, we set $\lambda_1 = \lambda_2 = 1$. These parameters can be adjusted according to specific tasks.

Based on the cost function, the global path is generated by the *global_planner* module in the navigation stack, and the local path is generated by the *dwa_local_planner* module in navigation stack [15].

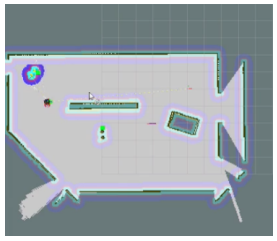
## V. Experiments

### A. Simulation experiment

In order to prove the concept, we build a simulation environment based on Gazebo [6], which is an open-source multi-robot simulation environment. In the simulated scenario, two men are standing in the room. The Turtlebot comes in without any a priori knowledge of the room. In the experiment, the turtlebot succeeds in mapping the room, finding the two people, and then visiting them in order. The result is shown in Figure 4. Figure 4(a) shows the simulation environment. It is a room with humans and furnitures inside. The robot enters this room without pre-built map of the room. Then the robot explores the room and interacts with each human. Figure 4(b) shows the exploration results. The map is built and the location of humans are labelled with green squares.



(a) The simulation scene.



(b) Mapping and localization results.

Fig. 4: Simulation experiment.

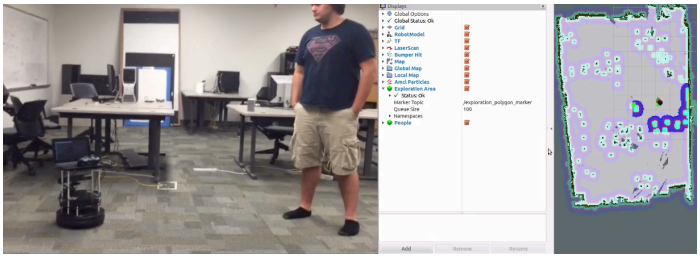


Fig. 5: Field experiment.

### B. Field experiment

We further tested the system in a real scene.

The real scene experiment is conducted as follows. Unlike the simulation part, the exploration is done with a pre-built map. First, the boundary of the exploration is set by drawing a polygon in the map. Then the robot begins to search for humans as well as update the costmap in this area. After it finishes searching, it revisites each human it met during the search and do face recognition. The experiment video is available for public [1].

An instance during the experiment is shown in Fig 5, where robot successfully did the exploration and face recognition.

### C. Analysis of experiments

In both simulation and field experiment, the robot successfully explores the room and find the person in the room. In the field experiment, the robot does further interaction with the human (face recognition). This shows the effectiveness of our system.

From observation, the exploration of the room is without redundant scanning of explored areas. So we can conclude the exploration is efficient.

[1]The *demonstration video* is available online: https://www.youtube.com/watch?v=OXM6cRdTb24

## VI. Conclusion and Future Work

In this paper, we presented a system capable of exploration in an environment with human. It can not only do the exploration and map the environment, but also perform human-aware actions, including avoiding collision with human, planning paths that are comfortable to human, and interacting with human.

The future work includes extending the single robot system to multi-robot system for cooperative exploration. Also for human-aware exploration part, more factors can be taken into account to improve the comfortable criteria. We will also consider the effect of different human behaviors on the costmap. For example when a human is talking with another one.

## Acknowledgment

This work is from the final project of Fall 2014 course ECE4580 (Computational Computer Vision) in Georgia Institute of Technology. We would like to thank the instructor, teaching assistants and classmates for valuable discussions. Fei Xia would also like to thank China Scholarship Council for supporting his exchange study in Georgia Institute of Technology.

## References

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[2] C.-P. Lam, C.-T. Chou, K.-H. Chiang, and L.-C. Fu, "Human-centered robot navigationtowards a harmoniously human–robot coexisting environment," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 99–112, 2011.

[3] J. Mainprice, E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon, "Planning human-aware motions using a sampling-based costmap planner," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5012–5017.

[4] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[5] S. Tranberg Hansen, M. Svenstrup, H. J. Andersen, and T. Bak, "Adaptive human aware navigation based on motion pattern analysis," in *IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2009, pp. 927–932.

[6] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2004, pp. 2149–2154.

[7] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*. IEEE, 1997, pp. 146–151.

[8] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 2000, pp. 476–481.

[9] WillowGarage, "TurtleBot open source hardware."

[10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[11] WillowGarage, "ROS people package, http://wiki.ros.org/people."

[12] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[13] J. M. Saragih, S. Lucey, and J. F. Cohn, "Face alignment through subspace constrained mean-shifts," in *International Conference on Computer Vision*. IEEE, 2009, pp. 1034–1041.

[14] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[15] WillowGarage, "ROS navigation stack, http://wiki.ros.org/navigation."

[16] "ROS frontier exploration package, http://wiki.ros.org/frontier_exploration."