Mobile robot system architecture for people tracking and following applications

Nicolas A. Olmedo, Hong Zhang, and Michael Lipsett

Abstract— **We present the design and implementation of a multi-input and multi-controller system architecture for mobile robots. The proposed system architecture is divided into interchangeable modular subsystems with common interfaces. The paper details the development of a prototype to test the proposed architecture and the implementation of the architecture with the Robot Operating System. Results from indoor and outdoor experiments are discussed. The prototype is used to demonstrate person following for route teach-and-repeat applications with collision avoidance and controller override.**

I. Introduction

The complexity of human-robot interactions has increased with advances in human interaction methods [1] and people tracking and following [2]. Intelligent systems have been developed for a variety of capacities, such as real time tracking [2], assistive home companions [3], and museum tour guides [4]. The computational methods for people detection, tracking and following are still active areas of research, [5], [6], [7], [8], [9], [10]. These advances have allowed robotic systems to conduct tasks of increasing difficulty.

Robotic systems used for multi-task operations have increased complexity in their system architecture and high level controllers. Multi-task operations require multi-controller systems, for example people tracking and following with collision avoidance, autonomous visual navigation, SLAM, route-repeating applications, etc. Various studies have investigated case-specific system architectures but have not aimed to propose a generalized system architecture for multi-task operations [3], [11], [12], [13], [14], [15], [16]. A generalized system architecture would provide reusability, expandability, and possible collaborations between researchers using robotic systems.

Partial teleoperation and partial autonomy are desired characteristics of robots working in industrial settings. This motivates the need of multi-input and multi-controller systems that can perform a variety of tasks. An example of such a system is a robot that is trained by an operator to navigate a predefined route in a dynamic environment. The operator might train the robot by teleoperating it over the route, or by letting the robot follow a person over the route. In many cases, the robot may need to switch between modes of operation in response to the environment or the input from the operator.

N. Olmedo is with the Department of Mechanical Eng., University of Alberta, Edmonton, Canada. olmedo@ualberta.ca

H. Zhang is with the Department of Computing Science, University of Alberta, Edmonton, Canada. zhang@cs.ualberta.ca

M. Lipsett is with the Department of Mechanical Eng., University of Alberta, Edmonton, Canada. mlipsett@ualberta.ca

The aim of this paper is to present a multi-input and multi-controller reusable system architecture for a mobile robot used in a multi-task operation. The requirements and proposed design solution will be discussed, followed by the development of a prototype for people tracking and following using laser range-finders. Next, the implementation using the Robot Operating System (ROS) [17] will be detailed and the results from indoor and outdoor field trails will be discussed. Finally, future work will be identified and conclusions presented.

II. System architecture design requirements

In many robotics applications, a multi-input multi-controller (MIMC) system architecture with modular subsystems is required. MIMC agents are commonly used for tasks with several operating modes. Multi-input systems use transducers to acquire information that is necessary to perform different tasks. In the case of robot teleoperation, the input can be a wireless joystick module used by the robot operator to send velocity commands to the mobile robot, while in the case of autonomous navigation, laser scanners can be used to map an environment, localize the robot, and identify objects of interest such as people. Multi-controller systems use various independent controllers running in parallel to modulate actuator input signals or set-points to lower level controllers. In many cases the controllers modulate the same control variables, such as the velocities of actuators, and therefore must be managed based on the state of the system or operating mode. For example, a mobile robot used in a teach-and-repeat operation typically has several controllers for person follower, collision avoidance and autonomous navigation all requiring the control of the robot.

The subsystems of MIMC agents are required to be modular. Modularity and common interfaces are characteristics that allow the system architecture to be designed, constructed, debugged, and most importantly, reused on a variety of robots and applications. These characteristics allow the robot designers and operators to interchange modules depending on the technique or method used by the robot to complete a task. Modular subsystems are also easy to isolate and test independently and layers of abstraction help manage the complexity of the entire system.

III. Proposed system architecture

We have used the requirements discussed in the previous subsection to design a generalized MIMC system architecture with modular subsystems for navigating a mobile robot (Fig. 1). The system architecture is divided into two main

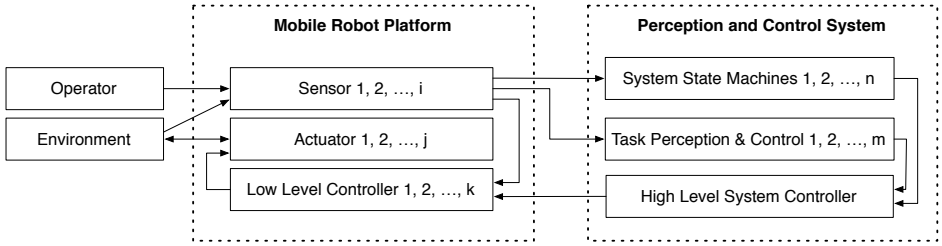

Fig. 1: Proposed system architecture. Three main PCS subsystems: system state machines, task perception and control and a high level system controller. The arrows represent information flow between modules.

components: the mobile robotic platform (MRP) and the perception and control system (PCS), each with subsystem modules.

A. Mobile Robotic Platform

The MRP is composed of sets of transducers and low level controllers. Transducers can be subdivided into sensors and actuators. Sensors detect proprioceptive and exteroceptive signals in one form of energy and convert it to another which can be processed, used or transmitted readily. Proprioceptive signals are internal signals such as joint encoder outputs used for feedback control while exteroceptive signals are external signals from the environment or system operators. Actuators receive energy and convert it to movement, such as DC motors and linear actuators. Electric motor speed controllers and joint position controllers are typical feedback controllers in mobile platforms.

B. Perception and Control System

The main objective of the PCS is to perceive the state of the system in an environment and modulate actuator input signals or set-points to lower level controllers. It is composed of three main subsystems: System State Machines (SSM), Task Perception & Control (TPC) and a High Level System Controller (HLSC).

1) System State Machines: The SSM is composed of a set of modules that maintain a record of the state of the overall system. Each module runs a state machine for a particular set of related states and updates it based on the output signals of the mobile platform sensors. For example, the current operation mode of the mobile platform can be selected using the signals from a set of joystick buttons controlled by a system operator.

2) Task Perception & Control: Independent modules can be used for specific task perception and control. In general, each task requires dedicated information processing and actuator modulation. Examples of such modules are: task-specific actuator controllers, state estimators, perception algorithms, tracking algorithms, etc. These modules operate in parallel, may use the same sensor information, and may modulate the same control variables.

3) High Level System Controller: The HLSC is composed of a set of modules that are used to integrate the information provided by the SSM and TPC subsystems, and output a single command to each control signal. The current state of the system is used to establish the operating mode of the robot. The operating mode determines the priority of the task controllers. For example, a collision avoidance controller may override a person follower controller if another object is too close to the robot, and a teleoperation controller may override the collision avoidance controller in cases when an operator needs to navigate the robot out of a difficult situation. The software logic of the HLSC is responsible of choosing what controller is in command of the robot based on the current systems states. Once a controller is selected, its output is used to modulate actuator input signals or set-points to lower level controllers.

4) System stability: Output of high level task controllers cannot affect low level controllers in such a way that they cause the system to be unstable. In general, low level controllers must have higher performance and bandwidth than high level controllers and the transitions between controllers must be stable. It is important that there not be controller actions at a high level that cause lower level controllers to generate positive reinforcement inputs to high level controllers that would drive the system unstable. The nature of the mobile platform needs to be taken into account in the design of both the low and high level controllers. It is expected that different controller architectures or parameters are required for different platforms, specially if the locomotion characteristics differ such as between aerial, humanoid, and wheeled platforms.

IV. Prototype development

Using the general system architecture described in the previous section, we have implemented a prototype for people tracking and following applications (Fig. 2). The system was designed to be a MIMC agent. The main features we would like to demonstrate with this prototype are:

1) Multi-controller design:
   a) teleoperation,
   b) collision avoidance,
   c) person tracker and follower (front),
   d) person tracker and follower (side), and
   e) autonomous navigation using route repeater.
2) Safety in unknown dynamic environments:
   a) emergency stop when an object is too close to the robot, and
   b) controller overrides

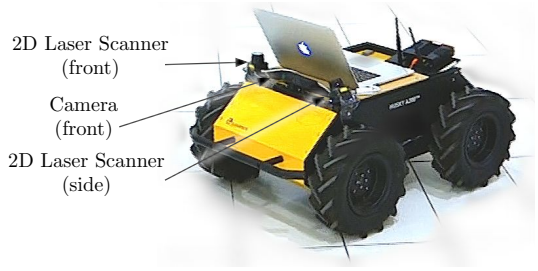

Fig. 2: Prototype mobile platform based on a Husky A200, instrumented with two laser scanners and a camera.

A. Mobile platform

The prototype was built using a Husky A200. The skid-steer platform has two geared DC motors, each driving the wheels of either side of the robot [18]. The motors have built-in encoders and independent motor driver. The platform is controlled by an internal Micro-Controller Unit (MCU) that communicates with an Intel Atom 1.6 GHz on-board computer under Ubuntu 12.04 LTS OS.

The prototype includes two Hokuyo URG-04LX-UG01 Scanning Laser Rangefinders and a digital camera (Fig. 2). Each laser range-finder has an angular resolution of 0.36 degrees, an accuracy of ± 30 mm and a maximum range of 5600 mm [19].

The front laser was used for forward collision avoidance and object tracking and following in front of the robot. It was mounted approximately 45 cm above ground level, 10 cm to the right from the center of the robot, and 20 cm from the front of the robot. It was pointed forwards at 0 degrees from the horizontal plane.

The side laser was used for people tracking and following on the side of the robot. It was mounted 45 cm above the ground, 10 cm to the left from the center of the robot, and 20 cm from the front of the robot. It was pointed to the left side of the robot, and pointed upwards at approximately 30 degrees from the horizontal plane in order to have the laser scan pointing to the torso of a person walking beside the robot.

B. People tracking and following modules

Two object tracking and following modules exist. The top view of the robot with the laser scanners is pictured in Fig. 3. Each module is predefined with a static goal position. In the case of the front laser, the goal is at point $(dx_g, dy_g)_1 = (1.0, 0.10)$ in coordinate system x_1, y_1. For the side laser, the goal is located at point $(dx_g, dy_g)_2 = (1.0, 0.0)$ in coordinate system x_2, y_2. All the goal positions are given in meters. The main task of the object following modules is to control the linear and angular velocity set-points of the robot in order to maintain the tracked object in the goal position. The generalized feedback control design is shown in Fig. 4. For the object positions shown in Fig. 3, the direction of motion required to reduce the distance between the goal and the object is shown with red arrows. Depending on the operation mode of the robot, the front or side object following controller will be in control of the mobile platform.

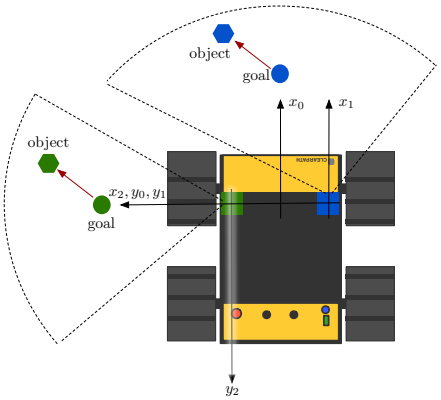

Fig. 3: Top view of the robot with front laser scanner (blue) and side laser scanner (green). The dotted lines represent the approximate field of view of each scanner.

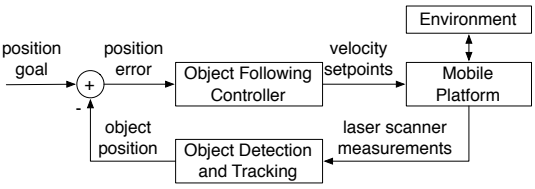

Fig. 4: Feedback control design for object detection, tracking and following.

Abstraction layers are used to simplify the implementation of multiple trackers and controllers. In Fig. 4, the blocks "Object Following Controller" and "Object Detection and Tracking" were designed to be modular and reusable with the following features:

1) single position goal in coordinate system x_0, y_0,
2) generalized Object Following Controller, with input: position error in coordinate system x_0, y_0, and
3) interchangeable Object Detection and Tracking methods, with outputs: positions of virtual objects in coordinate system x_0, y_0.

We use virtual objects to be able to calculate position error in coordinate system x_0, y_0 independently of the coordinate system of the laser used to detect the object, so that only one object following controller is necessary. For the front laser, the virtual object position is calculated by translating the measurements of the laser in coordinate system x_1, y_1, by $dy = 0.1$ in the y_1 direction. For the side laser, a virtual object is created in front of the robot as shown in Fig. 5 with $dx_0 = -dy_2$ and $dy_o = dx_2$.

V. Design Implementation

This section presents the design implementation for the prototype described in section IV. All components of the MIMC system were integrated using ROS. A system diagram is shown in Fig. 6. The different modules have been color

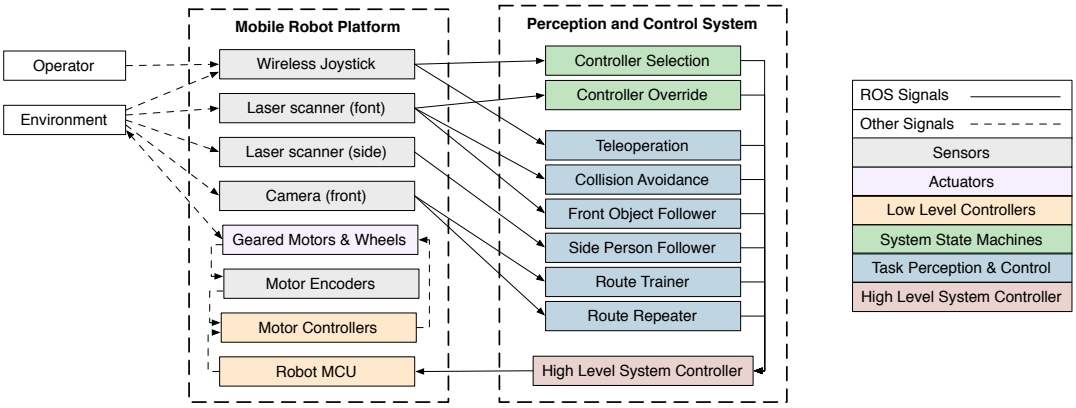

Fig. 6: System diagram of the ROS implementation of the proposed system architecture.

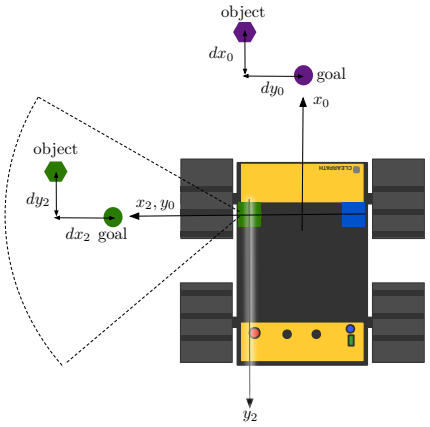

Fig. 5: Top view of the robot with real object detected by side laser scanner (green) and virtual object (purple) created in front of the robot.

coded to represent the different components presented in the generalized system architecture of Fig. 1.

ROS is a software platform commonly used for research robotics. As described in [20] and [21], "ROS is an open source, meta-operating system", meaning that it provides services such as "hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management." ROS works on top of traditional operating systems and "provides tools and libraries for obtaining, building, writing, and running code across multiple computers" [20]. ROS has several advantages such as distributed computation, software reuse and rapid testing/prototyping [21].

A. System modularity with ROS

As outlined in section II, modularity and common interfaces are requirements in a sound system architecture. ROS provides methods and techniques to produce modular systems that can be shared and reused. Computation requirements are organized into interchangeable software modules called nodes. Nodes can communicate between each other by publishing messages to topics. Messages are predefined data structures, while topics are the channels used to broadcast these messages. Nodes can subscribe or publish to topics to transmit and receive information. A complete description of ROS can be found in [17]. Modularity of the proposed system architecture was achieved by organizing the subsystems into nodes or groups of nodes and common interfaces was achieved by the standardization of messages and topics.

B. Mobile Robot Platform: sensors, actuators and low-level controllers

The main components of the MRP are listed in Fig. 6. Two laser scanners, a wireless joystick, and a camera collect the inputs and measurements used by the robot. All sensors are interfaced through ROS nodes running on an on-board Linux computer. Two identical ROS nodes communicate with the Hokuyo laser scanners and publish the range finders measurements to unique ROS topics. Another ROS node publishes the positions and states of the wireless joystick sticks and buttons. The camera module is built with its own ROS node that publishes raw and compressed images. All sensor measurements are timestamped.

A Husky A200 ROS node was provided by the base platform manufacturers. We use the Husky node as a bridge between the on-board computer with the robot MCU. The node subscribes and publishes to ROS topics to receive control commands and publish actuator sensory feedback respectively. The control commands are composed of set-points for the linear and angular velocities of the robot. We use the robot's velocities to calculate the corresponding angular velocities for it's wheels. The on-board computer transmits these angular velocities set-points to the robot MCU which relays them to the two motor controllers. The angular speeds of the wheels are controlled with closed-loop PID controllers using optical encoders on the shaft of the motors. The feedback from the encoders is transmitted to the on-board computer and published in ROS.

C. Task Perception & Control

Our prototype was developed to demonstrate six different tasks. Each task required specific perception and control

subsystems. Teleoperation was accomplished by mapping the feedback from the wireless joystick to the linear and angular velocities set-points of the robot. An ROS node was built for linear mapping between the position of the sticks and the velocities but in the future we can use exponential mapping to reduce the joystick sensitivity for motion at slow speeds.

We built a collision avoidance subsystem for the prototype demonstration. Collisions are avoided by stopping the robot when objects were detected to be closer than 0.3 m to the robot. The front laser scanner is used for this task. We selected this approach because the robot was tested around dynamic objects, such as people walking beside and in front of the robot and we found that people feel safer if the robot stopped if it gets too close, rather than if it would try to navigate around them.

A front Object Following Controller was implemented as described in section IV-B. The front laser scanner is used to estimate the position of the objects ahead of the robot. An Object Detection and Tracking subsystem was built with an ROS node that subscribes to the range finder measurements and uses a nearest point detector to track an object's positions. The estimated position is published to an ROS topic and is used to control the robot's linear and angular velocities. Two PID controllers are used for object following based on the position error. One PID controller modulates the robot's linear velocity set-point while a second PID controller modulates the robot's angular velocity set-point. The PID gains were tunned using the classical Zieglers-Nichols method and were fined tunned manually. The Object Following Controller was packaged into an ROS node.

The Object Following Controller ROS node was reused for the side Person Following Controller. Similarly to the front Object Detection and Tracking subsystem, an ROS node was built to subscribe the the side range finder measurements and publish the estimated object's position. As described in section IV-B, the Object Following Controller ROS node could be reused because of the standardization of coordinate systems, the generalization of the Object Following Controller, and the interchangeability of the Object Detection and Tracking methods.

A Route Trainer and Route Repeater ROS nodes were integrated to the prototype. Images are used for Graph-SLAM to construct maps and use them to navigate the robot autonomously. These modules were used for demonstration purposes of the multi-controller agent and their details are not within the scope of this paper.

D. High Level System Controller and System State Machines

The HLSC was designed to select which of the six Task Controllers described above is in control of the robot. Once a Task Controller is selected, the HLSC transmits the commands of the selected controller to the mobile platform. The HLSC selects the Task Controller based on the state of the Controller Selection and Controller Override state machines.

The Controller Selection state machine tracks the operator's desired Task Controller based on the state of the buttons of the wireless joystick. We designed our prototype so that the operator can quickly change the desired Task Controller. For example, if the object following or route repeating is operational, the operator can quickly switch to manual control by pressing any button of the joystick. Then, autonomous navigation can be resumed by pressing predetermined buttons linked to specific Task Controllers. This allows for short interventions of manual control and easy switching between autonomous controllers.

The Controller Override state machine determines if the Collision Avoidance subsystem needs to take precedence over an autonomous Task Controller such as object following or route repeating. If an object is detected to be too close to the robot, the state is changed so that the HLSC can give priority to the Collision Avoidance subsystem which stops the robot.

The HLSC is required to make the transitions between controllers stable. The smoothness of the transitions might also be important in some applications. In such cases, interpolation between control commands during controller transitions might eliminate sudden jumps in actuator inputs. No high level control action should cause low level controllers to create positive reinforcement to high level controllers which can drive the system unstable.

VI. Applications

To determine the feasibility of real-world application of the proposed architecture, we tested our prototype during the completion of two tasks: side person following for teach-and-repeat route following and outdoor robot following. Figures 7a and 7b show images during the experiments of side person following. Side person following was initially tested in open environments (Fig 7b) and then on a dynamic indoor environment (Fig 7a). During the indoor experiments the collision avoidance and controller selection features of the prototype were successfully demonstrated. The side person follower was used to train the robot to follow a route though a mall. Once the robot was trained over a path, the operator selected the Route Following Task Controller to let the robot autonomously navigate the path. The collision avoidance subsystem was essential to avoid collisions with people and objects that moved into the robot's path as the robot was autonomously following the route.

Front Object Following was used to follow another mobile platform as shown in Figure 7c. Experiments in the Canadian Space Agency's Mars Yard were conducted over various terrains. The object following methods were successfully demonstrated on trajectories that included sharp turns and reverse maneuvers.

VII. Conclusions and Future Work

We have presented the design and implementation of a reusable system architecture for mobile robots. Desired characteristics for the proposed system architecture were identified to be multi-input, multi-controller, modular and with common interfaces. The proposed system architecture is divided into two main components, the Mobile Platform and

(a) Person following in a dynamic indoor environment.

(b) Person following in an outdoors environment.

(c) Robot following experiments at CSA's Mars Yard.

Fig. 7: Indoor and outdoor field trials.

the Perception and Control Systems. The Mobile Platform integrates the sensors, actuators and low level controllers, while the Perception and Control system integrates the system state machines, tasks perception and control modules and high level system controller. System stability issues were addressed. The details of the development of a prototype to test the proposed system architecture were detailed. System modularity and common interfaces were achieved through the ROS implementation of the proposed subsystems. The systems architecture was successfully demonstrated in multi-task operation of people following for route teach-and-repeat applications as well as for robot following in outdoor environments with collision avoidance. Future work will include the addition of extra payloads to the robotic system. Instrumentation for terramechanics and environmental studies will be integrated to the platform within the existing system architecture as a set of actuators, perception and control modules. The new subsystems will be integrated using ROS.

REFERENCES

[1] M. Wang and J.-K. Liu, "A novel teleoperation paradigm for human-robot interaction," in *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, vol. 1, Dec 2004, pp. 13–18 vol.1.

[2] M. Volkhardt, C. Weinrich, and H.-M. Gross, "People tracking on a mobile companion robot," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, Oct 2013, pp. 4354–4359.

[3] H. Gross, C. Schroeter, S. Mueller, M. Volkhardt, E. Einhorn, A. Bley, T. Langner, M. Merten, C. Huijnen, H. van den Heuvel, and A. van Berlo, "Further progress towards a home robot companion for people with mild cognitive impairment," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, Oct 2012, pp. 637–644.

[4] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: a second-generation museum tour-guide robot," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 3, 1999, pp. 1999–2005 vol.3.

[5] S. Zheng, H. Qiao, B. Zhang, and P. Zhang, "The application of intrinsic variable preserving manifold learning method to tracking multiple people with occlusion reasoning," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct 2009, pp. 2993–2998.

[6] M. Hashimoto, T. Konda, Z. Bai, and K. Takahashi, "Identification and tracking using laser and vision of people maneuvering in crowded environments," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, Oct 2010, pp. 3145–3151.

[7] H. Schulz and D. Schulz, "Fast visual people tracking using a feature-based people detector," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 3614–3619.

[8] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele, "Learning people detectors for tracking in crowded scenes," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 1049–1056.

[9] V. Eiselein, H. Fradi, I. Keller, T. Sikora, and J.-L. Dugelay, "Enhancing human detection using crowd density measures and an adaptive correction filter," in *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, Aug 2013, pp. 19–24.

[10] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1815–1821.

[11] N. A. Mai and K. Janschek, "Generic system architecture for behavior-based mobile robot control using fuzzy logic," in *Control, Automation and Information Sciences (ICCAIS), 2012 International Conference on*, Nov 2012, pp. 253–258.

[12] W. Ren, J.-S. Sun, R. Beard, and T. McLain, "Experimental validation of an autonomous control system on a mobile robot platform," *Control Theory Applications, IET*, vol. 1, no. 6, pp. 1621–1629, November 2007.

[13] M.-S. Lim, J. Lim, J. Lim, and S.-R. Oh, "A hybrid system approach to motion control of wheeled mobile robots," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 1, Oct 1998, pp. 210–215 vol.1.

[14] I. Ullah, Q. Ullah, F. Ullah, and S. Shin, "Mobile robot navigation with distance control," in *Robotics and Artificial Intelligence (ICRAI), 2012 International Conference on*, Oct 2012, pp. 61–67.

[15] B. Putra, K. Mutijarsa, and W. Adiprawita, "Design and implementation of software architecture behavioral-based robot control system using active object computing model," in *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, July 2011, pp. 1–6.

[16] S. Hommel, M. Grimm, V. Voges, U. Handmann, and U. Weigmann, "An intelligent system architecture for multi-camera human tracking at airports," in *Computational Intelligence and Informatics (CINTI), 2012 IEEE 13th International Symposium on*, Nov 2012, pp. 175–180.

[17] M. Quigley, K. Conley, B. P. 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*, 2009.

[18] Clearpath-Robotics. (2014, 04) Husky. [Online]. Available: http://www.clearpathrobotics.com/husky/

[19] HOKUYO-AUTOMATIC. (2014, 04) Urg-04lx-ug01. [Online]. Available: http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/urg-04lx-ug01/

[20] O. S. R. Foundation. (2014, 04) Ros. [Online]. Available: http://www.ros.org/

[21] J. M. O'Kane, *A Gentle Introduction to ROS*. Independently published, Oct. 2013, available at http://www.cse.sc.edu/ jokane/agitr/.