A Situationally Aware Voice-Commandable Robotic Forklift Working Alongside People in Unstructured Outdoor Environments

Matthew R. Walter, Matthew Antone, Ekapol Chuangsuwanich, Andrew Correa, Randall Davis, Luke Fletcher, Emilio Frazzoli, Yuli Friedman, James Glass, Jonathan P. How, Jeong hwan Jeon, Sertac Karaman, Brandon Luders, Nicholas Roy, Stefanie Tellex, Seth Teller

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
mwalter@csail.mit.edu

Abstract

One long-standing challenge in robotics is the realization of mobile autonomous robots able to operate safely in human workplaces, and be accepted by the human occupants. We describe the development of a multi-ton robotic forklift intended to operate alongside people and vehicles, handling palletized materials within existing, active outdoor storage facilities.The system has four novel characteristics. The first is a multimodal interface that allows users to efficiently convey task-level commands to the robot using a combination of pen-based gestures and natural language speech. These tasks include the manipulation, transport, and placement of palletized cargo within dynamic, human-occupied warehouses. The second is the robot's ability to learn the visual identity of an object from a single user-provided example and use the learned model to reliably and persistently detect objects despite significant spatial and temporal excursions. The third is a reliance on local sensing that allows the robot to handle variable palletized cargo and navigate within dynamic, minimally-prepared environments without GPS. The fourth concerns the robot's operation in close proximity to people, including its human supervisor, pedestrians who may cross or block its path, moving vehicles, and forklift operators who may climb inside the robot and operate it manually. This is made possible by interaction mechanisms that facilitate safe, effective operation around people.

This paper provides a comprehensive description of the system's architecture and implementation, indicating how real-world operational requirements motivated key design choices. We offer qualitative and quantitative analyses of the robot operating in real settings and discuss the lessons learned from our effort.

Figure 1: (left) The mobile manipulation platform is designed to safely coexist with people within unstructured environments while performing material handling under the direction of humans.

1 Introduction

Robots are increasingly being seen not only as machines used in isolation for factory automation, but as aides that work with and alongside people, be it in hospitals, long-term care facilities, manufacturing centers, or our homes. Logistics is one such area in which there are significant benefits to having robots capable of working alongside people. Among the advantages is improved safety by reducing the risks faced by people operating heavy machinery. This is particularly true in disaster relief scenarios and for military applications, the latter of which motivates the work presented in this paper. It is not uncommon for soldiers operating forklifts on forward operating bases (FOBs) or elsewhere in theater to come under fire. Automating the material handling promises to take soldiers out of harm's way. More generally, robots that can autonomously load, unload, and transport cargo for extended periods of time offer benefits including increased efficiency and throughput, which extend beyond applications in military logistics.

The military domain raises two primary challenges for material handling that are common to more general manipulation scenarios. Firstly, the domain provides limited structure with dynamic, minimally-prepared environments in which people are free to move about and the objects to be manipulated and interacted with vary significantly and are unknown a priori. Secondly, any solution must afford effective command and control mechanisms and must operate in a manner that is safe and predictable, so as to be usable and accepted by existing personnel within their facilities. Indeed, a long-standing challenge to realizing robots that serve as our partners is developing interfaces that allow people to efficiently and reliably command these robots as well as interaction mechanisms that are both safe and accepted by humans.

Motivated by a desire for increased automation of logistics operations, we have developed a voice-commandable autonomous forklift (Fig. 1) capable of executing a set of commands to approach, engage, transport and place palletized cargo in minimally-structured outdoor settings. Rather than carefully preparing the environment to make it amenable to robot operation, we designed and integrated capabilities that allow the robot to operate effectively alongside people within existing unstructured environments, such as military Supply Support Activities (outdoor warehouses). The robot has to operate safely outdoors on uneven terrain, without specially-placed fiducial markers, guidewires or other localization infrastructure, alongside people on foot, human-driven vehicles,

and eventually other robotic vehicles, and amidst palletized cargo stored and distributed according to existing conventions. The robot also has to be commandable by military personnel without burdensome training. Additionally, the robot needs to operate in a way that is acceptable to existing military personnel and consistent with their current operational practices and culture. There are several novel characteristics of our system that enable the robot to operate safely and effectively despite challenging operational requirements, and that differentiate our work from existing logistic automation approaches. These include:

- Autonomous operation in dynamic, minimally-prepared, real-world environments, outdoors on uneven terrain without reliance on precision GPS, and in close proximity to people;
- Speech understanding in noisy environments;
- Indication of robot state and imminent actions to bystanders;
- Persistent visual memories of objects in the environment;
- Multimodal interaction that includes natural language speech and pen-based gestures grounded in a world model common to humans and the robot; and
- Robust, closed-loop pallet manipulation using only local sensing.

This paper presents a comprehensive review of the design and integration of our overall system in light of the requirements of automating material handling for military logistics. We present each of the different components of the system in detail and describe their integration onto our prototype platform. We focus in particular on the capabilities that are fundamental to our design approach and that we feel generalize to a broader class of problems concerning human-commandable mobile manipulation within unstructured environments. We evaluate the performance of these individual components and summarize the results of end-to-end tests of our platform within model and active military supply facilities. Some of the capabilities that we detail were originally presented within existing publications by the authors (Correa et al., 2010; Teller et al., 2010; Walter et al., 2010; Karaman et al., 2011; Tellex et al., 2011; Walter et al., 2012). The contribution of this paper is to provide a comprehensive, unified description of our overall system design, including its successful implementation within the target domain, together with an in-depth discussion of the lessons learned from our three year effort.

The remainder of the paper is organized out as follows. Section 2 describes existing work related to the general problem of material handling and the specific research areas that are fundamental to our approach. Section 3 discusses the requirements of automating military logistics and their influence on our design approach. Section 4 introduces the prototype forklift platform, including its sensing, actuation, and computing infrastructure. Section 5 describes in detail the different capabilities that comprise our solution and their integration into the overall system. Section 6 analyzes the performance of the key components of the system and summarizes the results of end-to-end deployments of the platform. Section 7 reflects on open problems that we feel are fundamental to realizing robots capable of effectively working alongside people in the material handling domain. Section 8 offers concluding remarks.

2 Related Work

There has been significant interest in automating material handling for mining (Nebot, 2005), heavy industries (Tews et al., 2007), and logistics (Durrant-Whyte et al., 2007; Wurman et al., 2008; Hilton, 2013). The state-of-the-art in commercial warehouse automation (Wurman et al., 2008; Hilton, 2013) is comprised of systems designed for permanent storage and distribution facilities. These indoor environments are highly prepared with flat floors that include fiducials for localization, substantial prior knowledge of the geometry and placement of objects to be manipulated, and clear separation between people and the robots' workspace. The structured nature of the facilities allows multiagent solutions that involve an impressively large number of robots operating simultaneously, backed by centralized resource allocation. In contrast, the military and disaster relief groups operate storage and distribution centers outdoors on uneven terrain, often for no more than a few months at a time. The facilities offer little preparation, precluding the use of guidewires, fiducials, or other localization aides. The objects in the environment are not standardized and the robot must manipulate and interact with different pallets and trucks whose geometry, location, and appearance are not known a priori. Further, people are free to move unconstrained throughout the robot's workspace on foot, in trucks, or in other manually-driven forklifts.

More closely related to our approach are solutions to automating forklifts and other autonomous ground vehicles that emphasize the use of vision (Cucchiara et al., 2000; Seelinger and Yoder, 2006; Kelly et al., 2007; Pradalier et al., 2010) and LIDAR (Bostelman et al., 2006; Lecking et al., 2006) to mitigate the lack of structure. Of particular note is the work by Kelly et al. who proposed vision-based solutions for localization, part rack detection, and manipulation that allow material handling vehicles to function autonomously within indoor environments with little-to-no additional structure (Kelly et al., 2007). Our system similarly emphasizes local sensing over external infrastructure, using vision for object recognition and LIDARs to estimate pallet and truck geometry and to detect people, obstacles, and terrain hazards. Whereas Kelly et al. have known CAD models of the objects to be manipulated, we assume only a rough geometric prior, namely that the pallets have slots and the height of the truck beds is within a common range. Unlike Kelly et al., whose system is capable of stacking part racks with the aid of fiducials and unloading enclosed tractor trailers, we only consider loading pallets from and to the ground and flatbed trailers, albeit in less-structured outdoor environments.

The most notable distinction between our system and the existing state-of-the-art is that our method is intended to work with and alongside people. To that end, we developed methods that allow users to command the robot using pen-based gestures and speech, and designed the system so that its actions are both safe and predictable, so as to be acceptable by military personnel. In the remainder of this section, we place in context our work in vision-based object detection, multimodal interface design, and human-robot interaction that enable the robot to work alongside people. For a description of work related to other aspects of design, we refer the reader to our earlier work (Teller et al., 2010; Walter et al., 2010).

2.1 Persistent Visual Memories

We endowed the robot with the ability to reliably and persistently recognize objects contained in its operating environment using vision. As we demonstrate, this capability enables people to command the robot to interact with cargo and trucks simply by referring to them by name. A key challenge is to develop an algorithm that can recognize objects across variations in scale, viewpoint, and lighting that result from operations in unstructured, outdoor environments.

Visual object recognition has received a great deal of attention over the past decade. Much of the literature describes techniques that are robust to the challenges of viewpoint variation, occlusions, scene clutter, and illumination. Generalized algorithms are typically trained to identify abstract object categories and delineate instances in new images using a set of exemplars that span the most common dimensions of variation. Training samples are further diversified through variations in the instances themselves, such as shape, size, articulation, and color. The current state-of-the-art (Savarese and Fei-Fei, 2007; Hoiem et al., 2007; Liebelt et al., 2008) involves learning relationships among constituent object parts represented using view-invariant descriptors. Rather than *recognition* of generic categories, however, the goal of our work is the *reacquisition* of specific previously observed objects. We still require invariance to camera pose and lighting variations, but not to intrinsic within-class variability, which allows us to build models from significantly fewer examples.

Some of the more effective solutions to object instance recognition (Lowe, 2001; Gordon and Lowe, 2006; Collet et al., 2009) learn 3D models of the object from different views that they then use for recognition. Building upon their earlier effort (Lowe, 2001), Gordon and Lowe (Gordon and Lowe, 2006) perform bundle adjustment on Scale Invariant Feature Transform (SIFT) features (Lowe, 2004) from multiple uncalibrated camera views to first build a 3D object model. Given the model, they employ SIFT matching, Random Sample and Consensus (RANSAC) (Fischler and Bolles, 1981), and Levenberg-Marquardt optimization to detect the presence of the object and estimate its pose. Collet et al. take a similar approach, using the Mean Shift algorithm in combination with RANSAC to achieve more accurate pose estimates that they then use for robot manipulation (Collet et al., 2009). These solutions rely upon an extensive offline training phase in which they build each object's representation in a "brute-force" manner by explicitly acquiring images from the broad range of different viewing angles necessary for bundle adjustment. In contrast, our one-shot algorithm learns the object's 2D appearance rather than its 3D structure and does so online by opportunistically acquiring views while the robot operates.

With respect to detecting the presence of specific objects within a series of images, our reacquisition capability shares similar goals with those of visual tracking. In visual tracking, an object is manually designated or automatically detected and its state is subsequently tracked over time using visual and kinematic cues (Yilmaz et al., 2006). General tracking approaches assume small temporal separation with limited occlusions or visibility loss, and therefore slow visual variation, between consecutive observations (Comaniciu et al., 2003). These trackers tend to perform well over short time periods but are prone to failure when an object's appearance changes or it disappears from the camera's field-of-view. To address these limitations, "tracking-by-detection" algorithms adaptively model variations in appearance online based upon positive detections (Lim

et al., 2004; Collins et al., 2005). These self-learning methods extend the tracking duration, but tend to "drift" as they adapt to incorporate the appearance of occluding objects or the background. This drift can be alleviated using self-supervised learning to train the model online using individual unlabeled images (Grabner et al., 2008; Kalal et al., 2010) or multiple instances (Babenko et al., 2009). These algorithms improve robustness and thereby allow an object to be tracked over longer periods of time despite partial occlusions and frame cuts. However, they are still limited to relatively short, contiguous video sequences. Although we use video sequences as input, our approach does not rely on a temporal sequence and is therefore not truly an object "tracker"; instead, its goal is to identify designated objects over potentially disparate views.

More closely related to our reacquisition strategy is the recent work by Kalal et al. that combines an adaptive tracker with an online detector in an effort to improve robustness to appearance variation and frame cuts (Kalal et al., 2009). Given a single user-provided segmentation of each object, their Tracking-Modeling-Detection algorithm utilizes the output of a short-term tracker to build an appearance model of each object that consists of image patch features. They employ this model to learn an online detector that provides an alternative hypothesis for an object's position, which is used to detect and reinitialize tracking failures. The algorithm maintains the model by adding and removing feature trajectories based upon the output of the tracker. This allows the method to adapt to appearance variations while removing features that may otherwise result in drift. While we do not rely upon a tracker, we take a similar approach of learning an object detector based upon a single supervised example by building an image-space appearance model online. Unlike Kalal et al.'s solution, however, we impose geometric constraints to validate additions to the model, which reduces the need to prune the model of erroneous features.

2.2 Multimodal User Interface

A significant contribution of our solution is the interface through which humans convey task-level commands to the robot using a combination of natural language speech and pen-based gestures. Earlier efforts to develop user interfaces for mobile robots differ with regards to the sharing of the robot's situational awareness with the user, the level of autonomy given to the robot, and the variety of input mechanisms available to the user.

The PdaDriver system (Fong et al., 2003) allows users to teleoperate a ground robot through a virtual joystick and to specify a desired trajectory by clicking waypoints. The interface provides images from a user-selectable camera for situational awareness. Other interfaces (Keskinpala et al., 2003) additionally project LIDAR and sonar returns onto images and allow the user to switch to a synthesized overhead view of the robot, which has been shown to facilitate teleoperation when images alone may not provide sufficient situational awareness (Ferland et al., 2009). Similarly, our interface incorporates the robot's knowledge of its surroundings to improve the user's situational awareness. Our approach is different, in that we render contextual knowledge at the object level (e.g., pedestrian detections) as opposed to rendering raw sensor data, which subsequent user studies (Keskinpala and Adams, 2004) have shown to add to the user's workload during teleoperation. A fundamental difference, however, is that our approach explicitly avoids teleoperation in favor of a task-level interface; in principle, this enables a single human supervisor to command multiple

robots simultaneously.

Skubic et al. provide a higher level of abstraction with a framework in which the user assigns a path and goal positions to a team of robots within a coarse user-sketched map (Skubic et al., 2007). Unlike our system, the interface is exclusive to navigation and supports only pen-based gesture input. Existing research related to multimodal robot interaction (Holzapfel et al., 2004; Perzanowski et al., 2001) exploits a combination of vision and speech as input. Perzanowski et al. introduce a multimodal interface that, in addition to pen-based gestures, accommodates a limited subset of speech and hand gestures to issue navigation-related commands (Perzanowski et al., 2001). Our approach is analogous as it combines the supervisor's visual system (for interpretation of the robot's surroundings) with speech (Glass, 2003) and sketch (Davis, 2002) capabilities. However, we chose to design a multimodal interface that uses speech and sketch as complementary, rather than as mutually disambiguating modes.

Our interface accompanies pen-based gestural interactions with the ability to follow commands spoken in natural language. The fundamental challenge to interpreting natural language speech is to correctly associate the potentially diverse linguistic elements with the robot's model of its state and action space. The general problem of mapping language to corresponding elements in the external world is known as the symbol grounding problem (Harnad, 1990). Recent efforts propose promising solutions to solving this problem in the context of robotics for the purpose of interpreting natural language utterances (Skubic et al., 2004; MacMahon et al., 2006; Dzifcak et al., 2009; Kollar et al., 2010; Tellex et al., 2011, 2012; Matuszek et al., 2010, 2012). Skubic et al. present a method that associates spoken references to spatial properties of the environment with the robot's metric map of its surroundings (Skubic et al., 2004). The capability allows users to command the robot's mobility based upon previous spoken descriptions of the scene. That work, like others (MacMahon et al., 2006; Dzifcak et al., 2009), models the mapping between the natural language command and the resulting plan as deterministic. In contrast, our method learns a distribution over the space of groundings and plans by formulating a conditional random field (CRF) based upon the structure of the natural language command. This enables us to learn the meanings of words and to reason over the likelihood of inferred plans (e.g., an indication of potential ambiguity), and provides a basis for performing human-robot dialog (Tellex et al., 2012). In similar fashion, other work has given rise to discriminative and generative models that explicitly account for uncertainty in the language and the robot's world model in the context of following route directions given in natural language (Kollar et al., 2010; Matuszek et al., 2010).

2.3 Predictable Interaction with People

Our robot is designed to operate in populated environments where people move throughout, both on foot and in other vehicles. It is important not only that the robot's actions be safe, which is not inconsequential for a 2700 kg vehicle, but that they be predictable. There is an extensive body of literature that considers the problem of conveying knowledge and intent for robots that have human-like forms. Relatively little work exists for non-anthropomorphic robots, for which making intent transparent is particularly challenging. The most common approach is to furnish the robots with additional hardware that provide visual cues regarding the robot's indented actions. These

include a virtual eye that can be used to indicate the direction in which the robot intends to move or a projector that draws its anticipated path on the ground (Matsumaru et al., 2005). We similarly use several visual means to convey the current state of the robot and to indicate its immediate actions. Additionally, we endow the robot with the ability to verbally announce its planned activities.

In addition to conveying the robot's intent, an important factor in people's willingness to accept its presence is that its actions be easily predictable (Klein et al., 2004). Several researchers (Alami et al., 2006; Takayama et al., 2011; Dragan and Srinivasa, 2013) have addressed the problem of generating motions that help to make a robot's intent apparent to its human partners. In particular, Takayama et al. show how techniques from animation can be used to facilitate a user's ability to understand a robot's current actions and to predict future actions. Dragan and Srinivasa describe a method that uses functional gradient optimization to plan trajectories for robot manipulators that deliberately stray from expected motions to make it easier for humans to infer the end effector's goal (Dragan and Srinivasa, 2013). In our case, we use the same visual and audible mechanisms that convey the robot's state to also indicate its actions and goals to any people in its surround. Having indicated the goal, the challenge is to generate motion trajectories that are consistent with paths that bystanders would anticipate the vehicle to follow. This is important not only for predictability, but for safety as well. We make the assumption that paths that are optimal in terms of distance are also predictable and use our anytime optimal sample-based motion planner to solve for suitable trajectories. A growing body of literature exists related optimal sample-based motion planning (Karaman and Frazzoli, 2010a,b, 2011; Marble and Bekris, 2011; Jeon et al., 2011) and we refer the reader to the work of Karaman and Frazzoli for a detailed description of the state-of-the-art in this area.

3 Design Considerations

In this section, we outline the fundamental characteristics of our system design in light of the demands of automating material handling for military logistics. The process of identifying these requirements involved extensive interaction with military personnel. We made repeated visits to several active military warehouses, where we interviewed personnel ranking from forklift operators to supervisors, and observed and recorded their operations to better understand their practices. Military and civilian logisticians also made several visits to MIT early and throughout the project where they operated and commented on our system. These interactions led us to identify requirements that are not specific to this application, but are instead general to mobile manipulation within dynamic, unstructured, human-occupied environments. In particular, the system must

- depend minimally on GPS or other metric global localization and instead emphasize local sensing;
- operate outdoors on uneven terrain, with little preparation;
- manipulate variable, unknown palletized cargo from arbitrary locations;
- load and unload flatbed trucks of unknown geometry;

- afford efficient command and control with minimal training;
- operate in a manner that is predictable and adheres to current practices so as to be accepted by existing personnel;
- be subservient to people, relinquishing command or asking for help in lieu of catastrophic failure;
- operate safely in close proximity to bystanders and other moving vehicles.

The military has a strong interest in reducing the reliance of their robotic platforms on GPS for localization. This stems from a number of factors, including the threat of signal jamming faced by systems that are deployed in theater. Additionally, achieving highly accurate positioning typically requires GPS/INS systems with price points greater than the cost of the base platform. An alternative would be to employ Simultaneous Localization and Mapping (SLAM) techniques, localizing against a map of the environment, however the warehouse is constantly changing as cargo is added and removed by other vehicles. Instead, we chose a framework that used intermittent, low-accuracy GPS for coarse, topological localization. In lieu of accurate observations of the robot's global pose, we employ a state estimation methodology that emphasizes local sensing and dead-reckoning for both manipulation and mobility. We also developed the robot's ability to automatically formulate maps of the environment that encode topological and semantic properties of the facility based upon a narrated tour provided by humans, thereby allowing people with minimal training to generate these maps.

The forklift must function within existing facilities with little or no special preparation. As such, the robot must be capable of operating outdoors, on packed earth and gravel while carrying loads of which the mass may vary by several thousand kilograms. Thus, we chose to adopt a non-planar terrain representation and a full 6-DOF model of chassis dynamics. We use laser scans of the terrain to detect and avoid hazards and combine these scans with readings from an IMU to predict and modulate the maximum vehicle speed based upon terrain roughness.

The forklift must be capable of detecting and manipulating cargo of which the location, geometry, appearance, and mass are not known a priori. We use an IMU to characterize the response of the forklift to acceleration, braking, and turning along paths of varying curvature when unloaded and loaded with various masses, in order to ensure safe operation. We designed a vision-based algorithm that enables the robot to robustly detect specific objects in the environment based upon a single segmentation hint from a user. The method's effectiveness lies in the ability to recognize objects over extended spatial and temporal excursions within challenging environments based upon a single training example. Given these visual detections, we propose a coupled perception and control algorithm that enables the forklift to subsequently engage and place unknown cargo to and from the ground and truck beds. This algorithm is capable of detecting and estimating the geometry of arbitrary pallets and truck beds from single laser scans of cluttered environments and uses these estimates to servo the forks during engagement and disengagement.

The robot must operate in dynamic, cluttered environments in which people, trucks, and other forklifts (manually-driven or autonomous) move unencumbered. Hence, the forklift requires full-

surround sensing for obstacle avoidance. We chose to base the forklift's perception on LIDAR sensors, due to their robustness and high refresh rate. We added cameras to provide situational awareness to a (possibly remote) human supervisor, and to enable vision-based object recognition. We developed an automatic multi-sensor calibration method to bring all LIDAR and camera data into a common coordinate frame.

Additionally, existing personnel must be able to effectively command the robot with minimal training, both remotely over resource-constrained networks and from positions nearby the robot. This bandwidth and time delay requirements of controlling a multi-ton manipulator preclude teleoperation. Additionally, the military is interested in a decentralized, scalable solution with a duty cycle that allows one person to command multiple vehicles, which is not possible with teleoperation. Instead, we chose to develop a multimodal interface that allows the user to control the robot using a combination of speech and simple pen-based gestures made on a handheld tablet computer.

There has been tremendous progress in developing a robot's ability to interpret completely free-form natural language speech. However, we feel that the challenge of understanding commands of arbitrary generality within noisy, outdoor environments are beyond the scope of current speech recognition, sensing, and planning systems. As a result, we chose to impose on the human supervisor the burden of breaking down high-level commands into simpler sub-tasks. For example, rather than command the robot to "unload the truck," the user would give the specific directives to "take the pallet of tires on the truck and place them in storage alpha," "remove the pallet of pipes and put them in storage bravo," etc. until the truck was unloaded. In this manner, the atomic user commands include a combination of summoning the forklift to a specific area; picking up cargo and placing it at a specified location. We refer to this task breakdown as "hierarchical task-level autonomy." Our goal is to reduce the burden placed on the user over time by making the robot capable of carrying out directives at ever-higher levels (e.g., completely unloading a truck pursuant to a single directive).

We recognize that an early deployment of the robot would not match the capability of an expert human operator. Our mental model for the robot is as a "rookie operator" that behaves cautiously and asks for help with difficult maneuvers. Thus, whenever the robot recognizes that it can not make progress at the current task, it can signal that it is "stuck" and request supervisor assistance. When the robot is stuck, the human supervisor can either use the remote interface to provide further information or abandon the current task, or any nearby human can climb into the robot's cabin and guide it through the difficulty via ordinary manned operation. The technical challenges here include recognizing when the robot is unable to make progress, designing the drive-by-wire system to seamlessly transition between unmanned and manned operation, and designing the planner to handle mixed-initiative operation.

Humans have a lifetime of prior experience with one another, and have built up powerful predictive models of how other humans will behave in almost any ordinary situation (Mutlu et al., 2009). We have no such prior models for robots, which in our view is part of the reason why humans are uncomfortable around robots: we do not have a good idea of what they will do next. However, the ability for robots to convey their understanding of the environment and to execute actions that make their intent transparent have often been cited as critical to effective human-robot collabo-

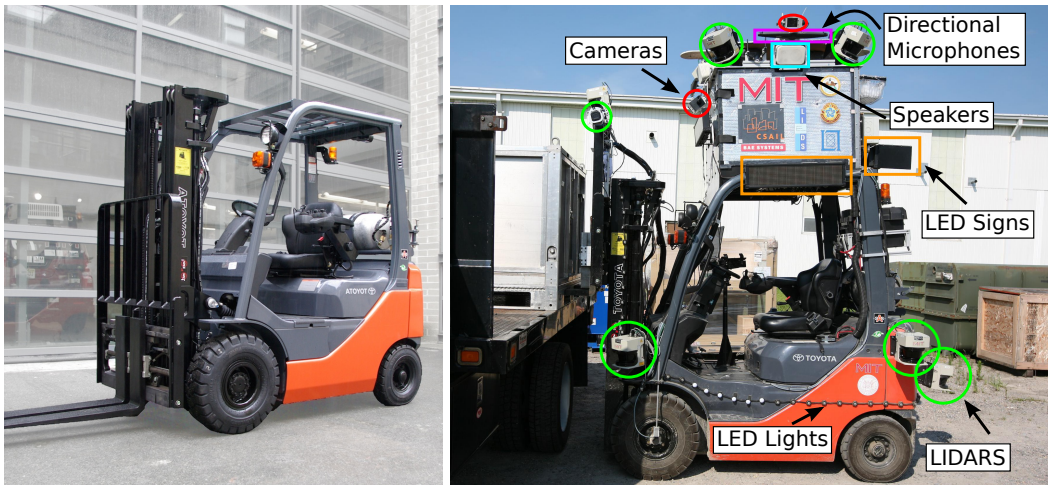


Figure 2: The platform is based upon (left) a stock 2700 kg Toyota lift truck. We modified the vehicle to be drive-by-wire and equipped it with LIDARs, cameras, encoders, an IMU, and directional microphones for perception; and LED signage, lights, and speakers for annunciation. The compartment on the roof houses three laptop computers, a network switch, and power distribution hardware.

ration (Klein et al., 2004). A significant design priority is thus the development of subsystems to support cultural acceptance of the robot. We added an *annunciation* subsystem that uses visible and audible cues to announce the near-term intention of the robot to any human bystanders. The robot also uses this system to convey its own internal state, such as the perceived number and location of bystanders. Similarly, people in military warehouse settings expect human forklift operators to stop whenever a warning is shouted. We have incorporated a continuously-running *shouted warning detector* into the forklift, which pauses operation whenever a shouted stop command is detected, and stays paused until given an explicit go-ahead to continue.

4 Mobile Manipulation Platform

We built our robot based upon a stock Toyota 8FGU-15 manned forklift (Fig. 2), a rear wheel-steered, liquid-propane fueled lift truck with a gross vehicle weight of 2700 kg and a lift capacity of 1350 kg. The degrees of freedom of the mast assembly include tilt, lift, and sideshift (lateral motion). We chose the Toyota vehicle for its relatively compact size and the presence of electronic control of some of the vehicle's mobility and mast degrees of freedom, which facilitated our drive-by-wire modifications.

4.1 Drive-by-Wire Actuation

We devised a set of electrically-actuated mechanisms involving servomotors to bring the steering column, brake pedal, and parking brake under computer control. A solenoid serves to activate the release latch to disengage the parking brake. Putting the parking brake under computer control

is essential, since OSHA regulations (United States Department of Labor Occupational Safety & Health Administration, 1969) dictate that the parking brake be engaged whenever the operator exits the cabin; in our setting, the robot sets the parking brake whenever it relinquishes control to a human operator. We interposed digital circuity into the existing forklift wiring system in order to control the throttle, mast, carriage, and tine degrees of freedom. Importantly, we integrated the digital acquisition devices in a manner that allows the robot to detect any control actions made by a human operator, which we use to seamlessly relinquish control.

4.2 Sensor Allocation

Fundamental to our design approach is the system's reliance on local sensing in lieu of assuming accurate global navigation. As such, we configured the forklift with a heterogeneous suite of proprioceptive and exteroceptive sensors that include cameras, laser range finders, encoders, an IMU, and a two-antenna GPS for periodic absolute position and heading fixes. We selected the sensor type and their placement based upon the requirements of the different tasks required of the vehicle.

For the sake of obstacle and pedestrian detection, we mounted five Sick LMS-291 planar LIDARs roughly at waist height to the side of the forklift, two at the front facing forward-left and -right, and three at the rear facing left, right, and rearward (Fig. 2). We positioned each LIDAR in a *skirt* configuration, but pitched them slightly downward such that the absence of a ground return would be meaningful. We oriented each sensor such that its field-of-view overlaps with at least one other LIDAR. Additionally, we mounted a Hokuyo UTM-30LX at axle height under the carriage looking forward in order to perceive obstacles when the forklift is carrying cargo that occludes the forward-facing skirts.

The robot operates on uneven terrain and must be able to detect and avoid hazards as well as to regulate its velocity based upon the roughness of the terrain. For this purpose, we positioned four Sick LIDARs on the roof facing front-left and -right, and rear-left and -right. We mounted them in a *pushbroom* configuration with a significant downwards canter (Fig. 2). As with the skirt LIDARs, we oriented the sensors such that their fields-of-view overlap with at least one other.

Our approach to engaging palletized cargo and placing it on and picking it up from truck beds relies upon laser range finders to detect and estimate the geometry of the palletized cargo and trucks. In order to servo the lift truck into the pallet slots, we placed one Hokuyo UTM-30LX LIDAR in a horizontal configuration on each of the two fork tines, scanning a half-disk parallel to and slightly above the tine (in practice, we used only one sensor). Additionally, we mounted two UTM-3OLX LIDARs on the outside of the carriage, one on each side with a vertical scanning plane, to detect and estimate the geometry of truck beds.

We mounted four Point Grey Dragonfly2 color cameras on the roof of the vehicle facing forward, rearward, left, and right, offering a 360° view around the forklift (Fig. 2). We utilize the camera images to perform object recognition. The system also transmits images at a reduced rate and resolution to the handheld tablet to provide the supervisor with a view of the robot's surround.

Finally, we equipped the forklift with four beam-forming microphones facing forward, rearward, left, and right (Fig. 2). The robot utilizes the microphones to continuously listen for spoken commands and for shouted warnings.

Our reliance on local sensing and our emphasis on multi-sensor fusion requires that we have accurate estimates for the body-relative pose of the many sensors on the robot. For each LIDAR and camera, we estimate the 6-DOF rigid-body transformation that relates the sensor's reference frame to the robot's body frame (i.e., "extrinsic calibration") through a chain of transformations that include all intervening actuatable degrees of freedom. For each LIDAR and camera mounted on the forklift body, this chain contains exactly one transformation; for LIDARs mounted on on the mast, carriage, or tines, the chain has as many as four transformations. For example, the chain for the tine-mounted Hokuyo involves changing transformations for the tine separation, carriage sideshift, carriage lift, and mast tilt degrees of freedom. We employ several different techniques to estimate each of these transformations, including bundle adjustment-like optimization (Leonard et al., 2008) for LIDAR-to-body calibration and multi-view, co-visibility constraint optimization (Zhang and Pless, 2004) to estimate the camera-to-LIDAR calibration.

4.3 Annunciation

In order to facilitate a bystander's ability to predict the robot's actions, we endowed it with multiple means by which to make its world model and intent transparent. Among these, we mounted four speakers to the roof, facing forward, rearward, left, and right (Fig. 2). The forklift uses these speakers to announce ensuing actions (e.g., "I am picking up the tire pallet"). In order to account for environment noise and to provide for less intrusive notification, we affixed four LED signs to the roof of the forklift. The signs display the robot's current operational state (e.g., "Active" or "Manual") as well as its immediate actions. We also ran LED lights around the body of the forklift, which we use to indicate the robot's state (i.e., by color) as well as as a *reflective display* to indicate its knowledge of people in its surround.

4.4 Computing Infrastructure

The software architecture includes several dozen processes that implement obstacle tracking, object detection, motion planning, control, and sensor drivers. The processes are distributed across four quad-core 2.53 GHz laptops running GNU/Linux, three located in the equipment cabinet on the roof and one affixed to the carriage (Fig. 2). An additional laptop located near the seat serves as a development interface. We employed a publish-subscribe model for inter-process communication (Huang et al., 2010) over a local Ethernet network. A commercial off-the-shelf 802.11g wireless access point provides network connectivity with the human supervisor's handheld tablet. The rooftop cabinet also houses a network switch, power supplies and relays, as well as digital acquisition (DAQ) hardware for drive-by-wire control.

The supervisor's tablet, a Nokia N810 handheld computer, constitutes a distinct computational resource. In addition to providing a visual interface through which the user interacts with the robot, the tablet performs pen-based gesture recognition and rudimentary speech recognition onboard.

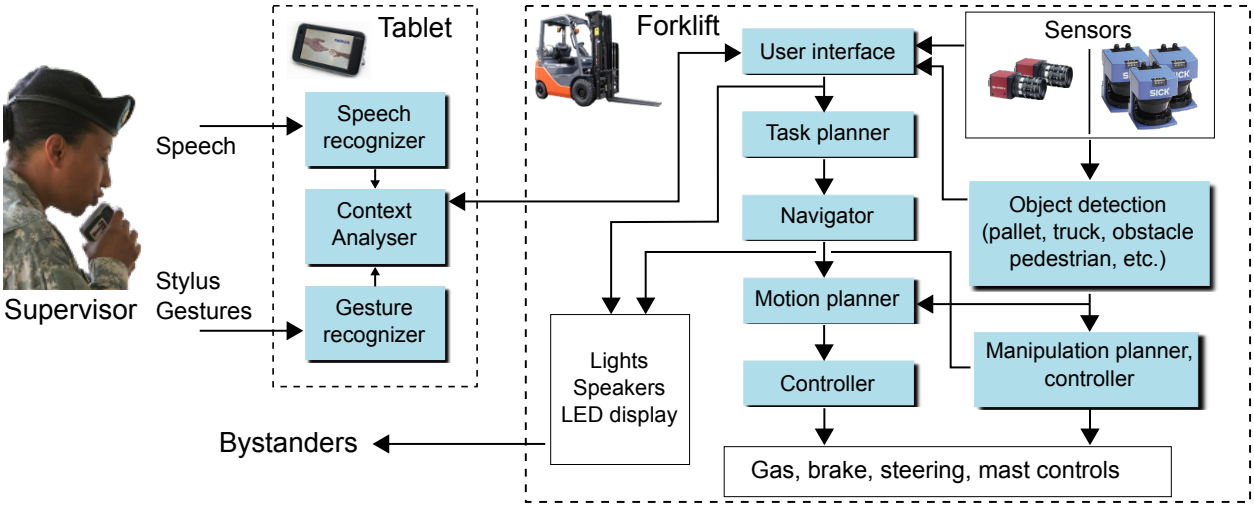


Figure 3: High-level system architecture.

The tablet offloads more demanding natural language understanding to the robot.

4.5 Power Consumption

The power to each of the systems onboard the forklift is supplied by an after-market alternator capable of supplying 1920 W. Devices requiring AC input, including the laptops, LED signs and lights, and network hardware are powered by a 600 W inverter. The remaining DC hardware is driven directly from the alternator via step-up and step-down regulators. The primary consumers of power are the five laptops (165 W continuous), the LED signs and lights (140 W), the speaker amplifier (100 W continuous), the three drive-by-wire motors (145 W continuous), and the Sick LIDARs (180 W continuous). In total, the continuous power consumption is approximately 1050 W. While the alternator is more than sufficient to drive the system under continuous load, it nears maximum capacity when the three drive-by-wire motors are at peak draw.

5 System Architecture

In this section, we outline a number of the components of our system that are critical to the robot's effective operation within unstructured environments. In similar fashion to our bottom-up design strategy, we start with the low-level, safety-critical capabilities and proceed to describe more advanced functionality.

5.1 Software

Our codebase is built upon middleware that we initially developed as part of MIT's participation in the DARPA Urban Challenge competition (Leonard et al., 2008). This includes the Lightweight Communications and Marshalling (LCM) utility (Huang et al., 2010), a low-level message passing

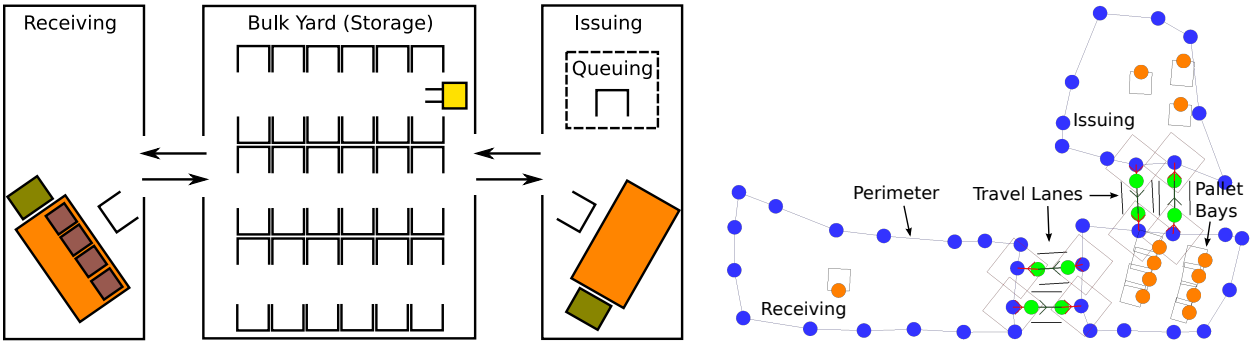
and data marshalling library that provides publish-subscribe inter-process communication among sensor handlers, perception modules, task and motion planners, controllers, interface handlers, and system monitoring and diagnostic modules (Fig. 3). As part of the project, we developed and make heavy use of the Libbot suite (Huang et al., 2014), a set of libraries and applications whose functionality includes 3D data visualization, sensor drivers, process management, and parameter serving, among others.

5.2 Robot System Integrity

The architecture of the forklift is based on a hierarchy of increasingly complex and capable layers. At the lowest level, kill-switch wiring disables ignition on command, allowing the robot or a user to safely stop the vehicle when necessary. Next, a programmable logic controller (PLC) uses a simple relay ladder program to enable the drive-by-wire circuitry and the actuator motor controllers from their default (braking) state. The PLC requires a regular heartbeat signal from the higher-level software and matching signals from the actuator modules to enable drive-by-wire control.

Higher still, the software architecture is a set of processes distributed across four of the networked computers and is composed of simpler device drivers and more complex application modules. At this level, there are many potential failure modes to anticipate, ranging from issues related to PC hardware, network connectivity, operating systems, sensor failures, up to algorithmic and application logic errors. Many of these failure modes, however, are mitigated by a software design methodology that emphasizes redundancy and multiple safety checks. This stems from our use of LCM message passing for inter-process communication, which is UDP-based and therefore contains no guarantees regarding message delivery. The consequence is that the application programmer is forced to deal with the possibility of message loss. What seems onerous actually has the significant benefit that many failure modes reduce to a common outcome of message loss. By gracefully handling the single case of message loss, the software becomes tolerant to a diverse range of failure types. As such, the software architecture is designed with redundant safety checks distributed across each networked computer that, upon detecting a fault, cause the robot to enter a *paused* state. These safety checks include a number of inter-process heartbeat messages that report the status of each of the sensors, communication bandwidth and latency, and clock times, among others. Higher-level algorithmic and logic errors are less obvious, particularly as their number and complexity compound as the system grows. We identify and detect the majority of these failures based upon designer input and extensive unit and system-wide testing.

A single process manages the robot's run-state, which takes the form of a finite state machine that may be *active* (i.e., autonomous), *manual* (i.e., override), or *paused*, and publishes the state at 50 Hz. If this process receives a fault message from any source, it immediately changes the state of the robot into the quiescent paused state. All processes involved in robot actuation listen to the run-state message. If any actuator process fails to receive this message for a suitably small duration of time, the process will raise a fault condition and go into the paused state. Similarly, the motion planning process will raise faults if timely sensor data is not received. Under this configuration, failures that arise as a result of causes such as the run-state process terminating, network communications loss, or one of the other sources identified above induce the robot into a



(a) Notional Warehouse (b) Topological Map

Figure 4: Renderings of (a) a notional military warehouse and (b) the topological map for a particular facility, each with storage, receiving, and issuing areas that are connected by lanes of travel (arrows).

safe state.

The only logic errors that this system does not address are those for which the robot appears to be operating correctly yet has an undetected error. Sanity checks by different software modules can help mitigate the effect of such errors but by definition some of these failures may go undetected. As far as we are aware, only branch, sub-system, and system-level testing can combat these kinds of failures. In an effort to better model potential failure modes, we developed and make extensive use of introspective and unit testing tools, in addition to field trials. The unit tests involve environment, sensor, and dynamic vehicle simulators that publish data of the same type and rate as their physical counterparts. Additionally, we log all inter-process messages during field and simulation-based tests. In the event of a failure, this allows us to more easily isolate the modules involved and to validate changes to these subsystems. While testing helps to significantly reduce the number of undetectable failures, we are not able to guarantee system integrity and instead rely upon user-level control to stop the vehicle in the event of catastrophic failure.

5.3 Local and Global State Estimation

The forklift operates in outdoor environments with minimal physical preparation. Specifically, we assume only that the warehouse consists of adjoining regions that we notionally model as delivery ("receiving") and pickup ("issuing") areas, as well as a "storage" area with bays labeled using a phonetic alphabet (e.g., "alpha bravo"). The robot performs state estimation and navigation within the environment using a novel coupling of two 6-DOF reference frames that are amenable to simultaneously integrating locally- and globally-derived data (Moore et al., 2009). The first is the *global frame* with respect to which we maintain coarse, infrequent (on the order of 1 Hz) estimates of the robot's absolute pose within the environment. In our system, these estimates follow from periodic GPS fixes, though they may also be the result of a SLAM implementation.

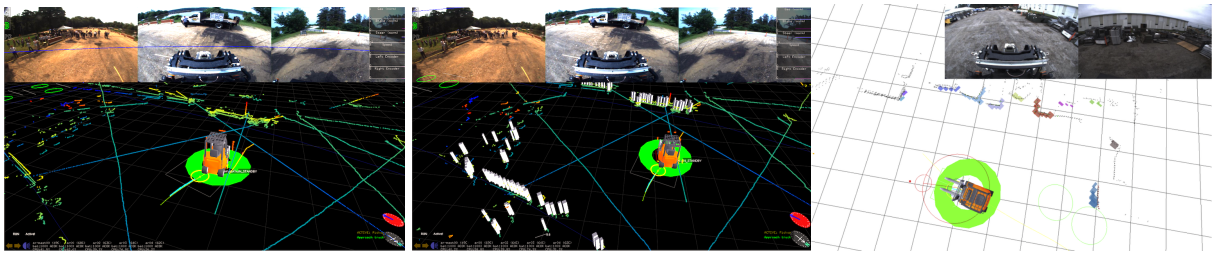
The second and most widely used is the *local frame*, a smoothly varying Euclidean reference frame with arbitrary initial pose about which we maintain high-resolution, high-rate pose estimates. The

local frame is defined to be the reference frame relative to which the vehicle's dead-reckoned pose is assumed to be correct (i.e., not prone to drift). The local frame offers the advantage that, by definition, the vehicle pose is guaranteed to move smoothly over time rather than exhibiting the abrupt jumps that commonly occur with GPS. State estimates that are maintained relative to the local frame are accurate for short periods of time but tend to exhibit drift in absolute pose over extended durations. The majority of the robot's subsystems favor high-accuracy, high-rate estimates of the robot's local pose over short time scales and easily tolerate inaccurate absolute position estimates. For that reason, we use the local frame to fuse sensor data for obstacle detection, pallet estimation and manipulation, and to plan the robot's immediate motion (i.e., upwards of a minute into the future). Some tasks (e.g., summoning to "receiving"), however, require coarse knowledge of the robot's absolute position in its environment. For that purpose, we maintain an estimate of the coordinate transformation between the local and global frames that allows the system to project geo-referenced data into the local frame (e.g., waypoints as Section 5.5 describes).

We model the robot's environment as a topology (Leonard et al., 2008) with nodes corresponding to key locations in the warehouse (e.g., the location of "receiving") and edges that offer the ability to place preferences on the robot's mobility (Fig. 4(b)). For example, we employ edges to model lanes in the warehouse (Fig. 4(a)) within which we can control the robot's direction of travel. With the exception of these lanes, however, the robot is free to move within the boundary of the facility. In order to make it easier for soldiers to introduce the robot to new facilities, we allow the forklift to learn the topological map during a guided tour. The operator drives the forklift through the warehouse while speaking the military designation (e.g., "receiving," "storage," and "issuing") of each region, and the system binds these labels with the recording of their GPS positions and boundaries. We then associate locations relevant to pallet engagement with a pair of "summoning points" that specify a rough location and orientation from which the robot may engage pallets (e.g., those in storage bays). The topological map of these GPS locations along with the GPS waypoints that compose the simple road network are maintained in the global frame and projected into the local frame as needed. Note that the specified GPS locations need not be precise; their purpose is only to provide rough goal locations for the robot to adopt in response to summoning commands, as a consequence of our navigation methodology. Subsequent manipulation commands are executed using only local sensing, and thus have no reliance on GPS.

5.4 Obstacle and Hazard Detection

Critical to ensuring that the robot operates safely is that it is able to detect and avoid people, other moving vehicles, and any stationary objects in its surround. For that purpose, the system includes modules for detecting and tracking obstacles and hazards (e.g., non-traversable terrain) in the environment, which are based upon our efforts developing similar capabilities for MIT's entry in the DARPA Urban Challenge (Leonard et al., 2008). The obstacle and hazard detection processes (within the "Object detection" block of Fig. 3) take as input range and bearing returns from the five planar skirt LIDARs positioned around the vehicle. The processes output the position and spatial extent of static obstacles detected within the environment, the location and extent of ground hazards, and the position, size, and estimated linear velocity of moving objects (e.g., people and other vehicles). We refer to the latter as *tracks*.



(a) Raw LIDAR Scans (b) Clustering into Chunks (c) Clustering into Groups

Figure 5: Obstacle detection operates by taking (a) the current set of LIDAR returns and first spatially clustering them into (b) *chunks*. These chunks are then grouped over space and time to form (c) *groups* to which we associate a location, size, and velocity estimate.

In order to improve the reliability of the detector, we intentionally tilted each LIDAR down by 5 degrees, so that they will generate range returns from the ground when no object is present. The existence of "infinite" range data enables the detector to infer environmental properties from failed returns (e.g., from absorptive material). The downward pitch reduces the maximum range to approximately 15 m, but still provides almost 8 seconds of sensing horizon for collision avoidance, since the vehicle's speed does not exceed 2 m/s.

The range and bearing returns from each LIDAR are first transformed into the smoothly-varying local coordinate frame based upon the learned calibration of each sensor relative to a fixed body frame. The system then proceeds to classify the returns as being either *ground*, *obstacles*, or *outliers*. The ground classification stems, in part, from the fact that it is difficult to differentiate between laser returns that emanate from actual obstacles and those that result from upward-sloping, yet traversable terrain. It is possible to distinguish between the two by using multiple LIDARs with scanning planes that are (approximately) parallel and vertically offset (Leonard et al., 2008). However, the five skirt LIDARs on the forklift are configured in a manner that does not provide complete overlap in their fields-of-view. Instead, we assume an upper bound on the ground slope and classify as obstacles any returns whose height is inconsistent with this bound. The one exception is for regions of sensor overlap when a second LIDAR with a higher scan plane does not get returns from the same (x, y) position.

Given a set of local frame returns from each of the five skirt LIDARs (Fig. 5(a)), we first perform preliminary spatiotemporal clustering on all returns classified as being non-ground, as a preprocessing step that helps to eliminate false positives. To do so, we maintain a 100 m $\times$ 100 m grid with 0.25 m cells that is centered at the vehicle. We add each return along with its associated timestamp and LIDAR identifier to a linked list that we maintain for each cell. Each time we update a cell we remove existing returns that are older than a maximum age (we use 33 ms, which corresponds to 2.5 scans from a Sick LMS-291). We classify cells with returns from different LIDARs or different scan times as candidate obstacles and pass the returns on to the obstacle clustering step.

Next, obstacle clustering groups these candidate returns into *chunks*, collections of spatially-close range and bearing returns (Fig. 5(b)). Each chunk is characterized by its center position in the local frame along with its (x, y) extent, which is restricted to be no larger than 0.25 m in any direction

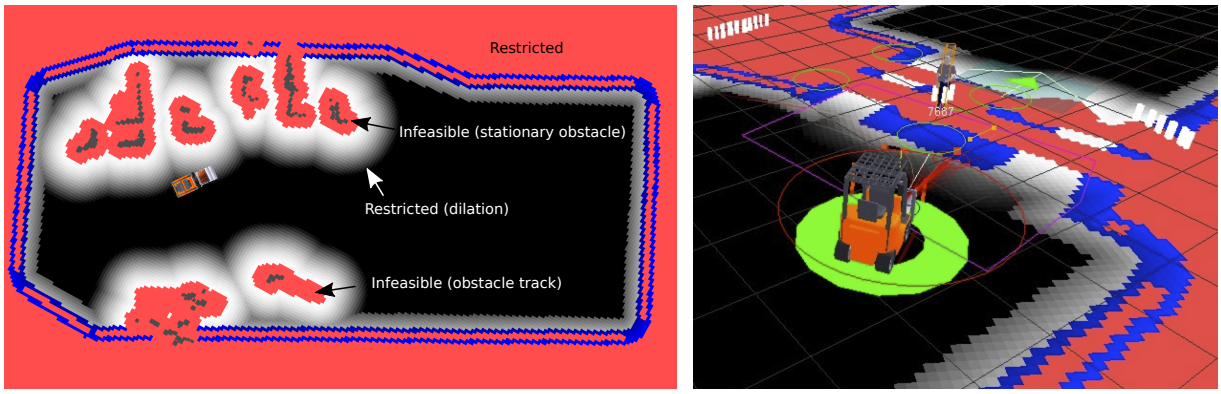


Figure 6: Drivability maps that model the costs associated with driving (left) near stationary and dynamic obstacles, and outside the robot's current operating region, as well as (right) in the vicinity of pedestrians.

to keep chunks small. This bound is intentionally smaller than the size of most obstacles in the environment. Given a return filtered using the preprocessing step, we find the closest existing chunk. If there is a match whose new size won't exceed the 0.25 m bound, we add the return to the chunk and update its position and size accordingly. Otherwise, we instantiate a new chunk centered at the return. After incorporating each of the new returns, we remove chunks for which a sufficiently long period of time has passed since they were last observed. We have empirically found 400 ms to be suitable given the speed at which the forklift travels. The next task is then to cluster together chunks corresponding to the same physical object into *groups*. We do so through a simple process of associating chunks whose center positions are within 0.3 m apart. Figure 5(c) demonstrates the resulting groups.

Next, we cluster groups over time in order to estimate the velocity of moving objects. At each time step, we attempt to associate the current set of groups with those from the previous time step. To do so, we utilize the persistence of chunks over time (subject to the 400 ms update requirement). As chunks may be assigned to different groups with each clustering step, we employ voting whereby each chunk in the current group nominates its association with the group from the previous time step. We then compare the spatial extents of the winning group pair between subsequent time steps to get a (noisy) estimate of the object's velocity. We use these velocity estimates as observations in a Kalman filter to estimate the velocity of each of the group's member chunks. These estimates yield a velocity *track* for moving obstacles. For a more detailed description of the spatiotemporal clustering process, we refer the reader to our earlier work (Leonard et al., 2008).

We integrate obstacle detections and estimated vehicle tracks into a *drivability map* that indicates the feasibility of positioning the robot at different points in its surround. The drivability map takes the form of a 100 m $\times$ 100 m, 0.20 m resolution grid map centered at a position 30 m in front of the vehicle that we maintain in the local frame. Each cell in the map expresses a 0–255 cost of the vehicle being at that location. The map encodes three different types of regions, those that are deemed *infeasible*, those that are *restricted*, and those that are *high-cost*. Infeasible regions denote areas in which the vehicle can not drive, most often resulting from the detection of obstacles. Areas classified as restricted are those for which there is a strong preference for the robot to avoid, but that

the robot can drive in if necessary. For example, open areas that lie outside the virtual boundary of the warehouse environment are deemed restricted, since the robot can physically drive there though we prefer that it doesn't. High-cost regions, meanwhile denote areas where there is an increased risk of collision and follow from a spatial dilation of obstacle detections. We use these regions to account for uncertainties that exists in the LIDAR data, our obstacle detection capability, and the robot's trajectory controller. Individually, these uncertainties are typically small, but they can compound and lead to a greater risk of collision. The high-risk regions allow for us to add a level of risk aversion to the robot's motion.

We populate the drivability map as follows. We start with a map in which each cell is labeled as restricted. We then "carve out" the map by assigning zero cost to any cell that lies within the robot's footprint or within the zone in the topological map (e.g., "Receiving," Fig. 4(b)) in which it is located. Next, we update the map to reflect the location of each stationary and moving obstacle group by assigning maximum cost to any cell that even partially overlaps with an obstacle's footprint. In the case of moving obstacles, we additionally use its velocity track to label as restricted each cell that the obstacle's footprint is predicted to overlap over the next 2.0 s. We chose 2.0 s due to the relatively slow speed at which our forklift operates and because other vehicles change their speed and direction of travel fairly frequently, which would otherwise invalidate our constant-velocity model. Next, we dilate cells that are nearby obstacles by assigning them a cost that scales inversely with their distance from the obstacle, resulting in the high-cost labeling. Figure 6 presents an example of a drivability map and a sampled motion plan. The drivability map is rendered in this manner at a frequency of 10 Hz or immediately upon request by another process (e.g, the motion planner), based upon the most recently published obstacle information.

Pedestrian safety is central to our design. Though LIDAR-based people detectors exist (Hähnel et al., 2003; Cui et al., 2007; Arras et al., 2007), we opt to avoid the risk of misclassification by treating all objects of suitable size as potential humans. The system applies a larger dilation to obstacles that are classified as being pedestrians. For pedestrians that are stationary, this results in a greater reduction of the vehicle's speed when in their vicinity. For pedestrians that are moving, we employ a greater look-ahead time when assigning cost to areas that they are predicted to occupy. When pedestrians cross narrow areas such as the lanes between regions, they become restricted and the robot will stop and wait for the person to pass before proceeding (Fig. 6). Pedestrians who approach too closely cause the robot to pause.

5.5 Planning and Control

The most basic mobility requirement for the robot is to move safely from a starting pose to its destination pose. The path planning subsystem (Fig. 3) consists of two distinct components: a *navigator* that identifies high-level routes through the map topology and a lower-level kinodynamic motion planner. Adapted from MIT's DARPA Urban Challenge system (Leonard et al., 2008), the navigator is responsible for identifying the shortest sequence of waypoints through the warehouse route network (maintained in the global frame) and for tracking and planning around blockages in this network. It is the job of the navigator to respect mobility constraints encoded in the map topology, such as those that model the preference for using travel lanes to move between warehouse

regions. Given a desired goal location in the topology, the navigator performs A* search (Hart et al., 1968) to identify the lowest cost (shortest time) route to the goal while respecting known blockages in the topology. The navigator maintains the sequence of (global frame) waypoints and publishes the local frame position of the next waypoint in the list for the local kinodynamic planner.

Given the next waypoint in the local frame, the goal of the motion planner is to quickly find a cost-efficient path that respects the dynamics of the vehicle and avoids obstacles as indicated by the drivability map. The challenge is that any motion planning method meant for practical deployment on a robot must be capable of operating within limited real-time computational resources. It also must tolerate imperfect or incomplete knowledge of the robot's operating environment. In the context of the forklift, the robot spends no more than a few seconds to plan a path (e.g., while changing gears) before driving towards the goal, which may take several minutes. In this setting, it would be useful if the robot were able to utilize available computation time as it moves along its trajectory to improve the quality of the remaining portion of the planned path. Furthermore, as the robot executes the plan, its model of the environment will change as vehicles and people move and new parts of the surround come into view. The estimate of the robot's state will also change, e.g., due to the unobservable variability of the terrain (e.g., wheel slip).

To address these challenges, we developed a motion planner that exhibits two key characteristics. First, the algorithm operates in an *anytime* manner: it quickly identifies feasible, though not necessarily optimal, motion plans and then takes advantage of available execution time to incrementally improve the plan over time towards optimality. Secondly, the algorithm repeatedly replans whereby it incorporates new knowledge of the robot state and the environment (i.e., the drivability map) and re-evaluates its existing set of plans for feasibility.

Our anytime motion planner (Fig. 3) was originally presented in our earlier paper (Karaman et al., 2011) and is based upon the RRT* (Karaman and Frazzoli, 2010a), a sample-based algorithm that exhibits the anytime optimality property, i.e., almost-sure convergence to an optimal solution with guarantees on probabilistic completeness. The RRT* is well-suited to anytime robot motion planning. Like the RRT, it quickly identifies an initial feasible solution. Unlike the RRT, however, the RRT* utilizes any additional computation time to improve the plan toward the optimal solution. We leverage this quality by proposing modifications to the RRT* that improve its effectiveness for real-time motion planning.

5.5.1 The RRT* Algorithm

We first describe a modified implementation of the RRT* and then present extensions for on-line robot motion planning. Let us denote the dynamics of the forklift in the general form $\dot{x}(t) = f(x(t), u(t))$, where the state $x(t) \in X$ is the position (x, y) and orientation θ, and $u(t) \in U$ is the forward velocity and steering input. Let X_{obs} denote the *obstacle region*, and $X_{\text{free}} = X \setminus X_{\text{obs}}$ define the *obstacle-free space*. Finally, let $X_{\text{goal}} \subset X$ be the goal region that contains the local frame position and heading that constitute the desired waypoint.

The RRT* algorithm solves the optimal motion planning problem by building and maintaining a tree $\mathcal{T} = (V, E)$ comprised of a vertex set V of states from X_{free} connected by directed edges

$E \subseteq V \times V$. The manner in which the RRT* generates this tree closely resembles that of the standard RRT, with the addition of a few key steps that achieve optimality. The RRT* algorithm uses a set of basic procedures, which we describe in the context of kinodynamic motion planning.

Sampling: The `Sample` function uniformly samples a state $x_{\text{rand}} \in X_{\text{free}}$ from the obstacle-free region of the state space. We verify that the sample is obstacle-free by querying the drivability map and using a threshold to determine whether the sample is collision-free.

Nearest Neighbor: Given a state $x \in X$ and the tree $\mathcal{T} = (V, E)$, the $v = \texttt{Nearest}(T, x)$ function returns the nearest node in the tree in terms of Euclidean distance.

Near Vertices: The $\texttt{Near}(V, x)$ procedure returns the set of all poses in V that lie within a ball of volume $O((\log n)/n)$ centered at x, where $n := |V|$.

Steering: Given two poses $x, x' \in X$, the $\texttt{Steer}(x, x')$ procedure returns a path $\sigma : [0, 1] \to X$ that connects x and x', i.e., $\sigma(0) = x$ and $\sigma(1) = x'$. Assuming a Dubins model (Dubins, 1957) for the vehicle kinematics, we use a steering function that generates curvature-constrained trajectories. The dynamics take the form

$$\begin{aligned} \dot{x}_D &= v_D \cos(\theta_D) \\ \dot{y}_D &= v_D \sin(\theta_D) \\ \dot{\theta}_D &= u_D, \qquad |u_D| \leq \frac{v_D}{\rho}, \end{aligned}$$

where (x_D, y_D) and θ_D specify the position and orientation, u_D is the steering input, v_D is the velocity, and ρ is the minimum turning radius. Six types of paths characterize the optimal trajectory between two states for a Dubins vehicle, each specified by a sequence of left, straight, or right steering inputs (Dubins, 1957). We use four path classes for the forklift and choose the steering between two states that minimizes cost.

Collision Check: The $\texttt{CollisionFree}(\sigma)$ procedure verifies that a specific path σ does not come in collision with obstacles in the environment, i.e., $\sigma(\tau) \in X_{\text{free}}$ for all $\tau \in [0, 1]$. We evaluate collisions through computationally efficient queries of the drivability map that determine the collision cost of traversing a particular path with the forklift footprint. Because the drivability map values are not binary, we impose a threshold to determine the presence of a collision.

Lists and Sorting: We employ a list L of triplets (c_i, x_i, σ_i), sorted in ascending order according to cost.

Cost Functional: Given a vertex x of the tree, we let $\texttt{Cost}(x)$ be the cost of the unique path that starts from the root vertex x_{init} and reaches x along the tree. With a slight abuse of notation, we denote the cost of a path $\sigma : [0, 1] \to X$ as $\texttt{Cost}(\sigma)$ for notational simplicity.

The RRT* follows the general structure shown in Algorithm 1 using the above functions. The algorithm iteratively maintains a search tree through four key steps. In the first phase, the RRT* algorithm samples a new robot pose x_{new} from X_{free} (Line 3), and computes the set X_{near} of all vertices that are close to x_{new} (Line 4). If X_{near} is an empty set, then X_{near} is updated to include

Algorithm 1: The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
    if $X_{\text{near}} = \emptyset$ **then**
        $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
    $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{new}})$;
    $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
    if $x_{\text{parent}} \neq \texttt{NULL}$ **then**
        $V.\texttt{add}(x_{\text{new}})$;
        $E.\texttt{add}(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
        $E \leftarrow \texttt{RewireVertices}(E, L_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

Algorithm 2: $\texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{new}})$

$L_{\text{near}} \leftarrow \emptyset$;
for $x_{\text{near}} \in X_{\text{near}}$ **do**
    $\sigma_{\text{near}} \leftarrow \texttt{Steer}(x_{\text{near}}, x_{\text{new}})$;
    $c_{\text{near}} \leftarrow \texttt{Cost}(x_{\text{near}}) + \texttt{Cost}(\sigma_{\text{near}})$;
    $L_{\text{near}}.\texttt{add}(\,(c_{\text{near}}, x_{\text{near}}, \sigma_{\text{near}})\,)$;
$L_{\text{near}}.\texttt{sort}()$;
return L_{near};

Algorithm 3: $\texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$

for $(c_{\text{near}}, x_{\text{near}}, \sigma_{\text{near}}) \in L$ **do**
    if $\texttt{CollisionFree}(\sigma_{\text{near}})$ **then**
        return x_{near};
return NULL

Algorithm 4: $\texttt{RewireVertices}(E, L_{\text{near}}, x_{\text{new}})$

for $(c_{\text{near}}, x_{\text{near}}, \sigma_{\text{near}}) \in L$ **do**
    if $\texttt{Cost}(x_{\text{new}}) + c(\sigma_{\text{near}}) < \texttt{Cost}(x_{\text{near}})$ **then**
        if $\texttt{CollisionFree}(\sigma_{\text{near}})$ **then**
            $x_{\text{oldparent}} \leftarrow \texttt{Parent}(E, x_{\text{near}})$;
            $E.\texttt{remove}(\,(x_{\text{oldparent}}, x_{\text{near}})\,)$;
            $E.\texttt{add}(\,(x_{\text{new}}, x_{\text{near}})\,)$;
return E

the vertex in the tree that is closest to x_{new} (Lines 5-6).

In the second phase, the algorithm calls the `PopulateSortedList`$(X_{\text{near}}, x_{\text{new}})$ procedure (Line 7). This procedure, given in Algorithm 2, returns a list of sorted triplets of the form $(c_{\text{near}}, x_{\text{near}}, \sigma_{\text{near}})$, for all $x_{\text{near}} \in X_{\text{near}}$, where (i) σ_{near} is the lowest cost path that connects x_{near} and x_{new} and (ii) c_{near} is the cost of reaching x_{new} by following the unique path in the tree that reaches x_{near} and then following σ_{near} (see Line 4 of Algorithm 2). The triplets of the returned list are sorted according to ascending cost. Note that at this stage, the paths σ_{near} are *not* guaranteed to be collision-free.

In the third phase, the RRT* algorithm calls the `FindBestParent` procedure, given in Algorithm 3, to determine the minimum-cost collision-free path that reaches x_{new} through one of the vertices in X_{near}. With the vertices presented in the order of increasing cost (to reach x_{near}), Algorithm 3 iterates over this list and returns the first vertex x_{near} that can be connected to x_{new} with a collision-free path. If no such vertex is found, the algorithm returns `NULL`.

If the `FindBestParent` procedure returns a non-`NULL` vertex x_{parent}, the final phase of the algorithm inserts x_{new} into the tree as a child of x_{parent}, and calls the `RewireVertices` procedure to perform the "rewiring" step of the RRT*. In this step, the `RewireVertices` procedure, given in Algorithm 4, iterates over the list L_{near} of triplets of the form $(c_{\text{near}}, x_{\text{near}}, \sigma_{\text{near}})$. If the cost of the unique path that reaches x_{near} along the vertices of the tree is higher than reaching it through the new node x_{new}, then x_{new} is assigned as the new parent of x_{near}.

5.5.2 Extensions to Achieve Anytime Motion Planning

The RRT* is an anytime algorithm in the sense that it returns a feasible solution to the motion planning problem quickly, and given more time it provably improves this solution toward the optimal one. Next, we describe extensions that significantly improves the path quality during execution.

The first extension is to have the planner *commit* to an initial portion of the trajectory while allowing the planner to improve the remaining portion of the tree. Upon receiving the goal region, the algorithm starts an *initial planning phase* in which the RRT* runs until the robot must start moving toward its goal. This time is on the order of a few seconds, which corresponds to the time required to put the forklift in gear. Once the initial planning phase is completed, the online algorithm goes into an *iterative planning phase*, in which the robot starts to execute the initial portion of the best trajectory in the RRT* tree. In the process, the algorithm focuses on improving the remaining part of the trajectory. Once the robot reaches the end of the portion that it is executing, the iterative phase is restarted by picking the current best path in the tree and executing its initial portion.

More precisely, the iterative planning phase occurs as follows. Given a motion plan $\sigma : [0, T] \to X_{\text{free}}$ generated by the RRT* algorithm, the robot starts to execute an initial portion $\sigma : [0, t_{\text{com}}]$ until a given commit time t_{com}. We refer to this initial path as the *committed trajectory*. Once the robot starts executing the committed trajectory, the algorithm deletes each of its branches from the committed trajectory and makes the end $x(t_{\text{com}})$ the new tree root. This effectively shields the committed trajectory from any further modification. As the robot follows the committed trajec-

tory, the algorithm continues to improve the motion plan within the new (i.e., uncommitted) tree of trajectories. Once the robot reaches the end of the committed trajectory, the procedure restarts, using the initial portion of what is currently the best path in the RRT* tree to define a new committed trajectory. The iterative phase repeats until the robot reaches the desired waypoint region.

5.5.3 Branch-and-Bound and the Cost-to-Go Measure

We additionally include a branch-and-bound mechanism to more efficiently build the tree. The basic idea behind the branch-and-bound heuristic is that the cost of any feasible solution to the minimum-cost trajectory problem provides an upper bound on the optimal cost, and the lowest of these upper bounds can be used to prune certain parts of the search tree.

For an arbitrary state $x \in X_{\text{free}}$, let c_x^* be the cost of the optimal path that starts at x and reaches the goal region. A *cost-to-go function* $\texttt{CostToGo}(x)$ associates a real number between 0 and c_x^* with each $x \in X_{\text{free}}$ that provides a lower bound on the optimal cost to reach the goal. We use the minimum execution time as the cost-to-go function, i.e., the Euclidean distance between x and X_{goal} (neglecting obstacles) divided by the robot's maximum speed.

The branch-and-bound algorithm works as follows. Let $\texttt{Cost}(x)$ denote the cost of the unique path that starts from the root node and reaches x through the edges of $\mathcal{T}$. Let $x_{\min} \in X_{\text{goal}}$ be the node currently in the tree with the lowest-cost trajectory to the goal. The cost of its unique trajectory from the root gives an upper bound on cost. Let V' denote the set of nodes for which the cost to get to x plus the lower bound on the optimal cost-to-go is more than the upper bound $\texttt{Cost}(x_{\min})$, i.e., $V' = \{x \in V \mid \texttt{Cost}(x) + \texttt{CostToGo}(x) \geq \texttt{Cost}(x_{\min})\}$. The branch-and-bound algorithm keeps track of all such nodes and periodically deletes them from the tree.

In addition to the anytime characteristic, an important property of the algorithm is its use of replanning to accommodate uncertain, dynamic environments. With each iteration, the current committed trajectory is checked for collisions by evaluating the most recent drivability map, which is updated as new sensor information becomes available. If the committed plan is found to be in collision, the robot will come to a stop. The algorithm then reinitializes the tree from the robot's current location. Additionally, since it is computationally infeasible to check the entire tree, we perform *lazy checks* of what is currently the best path to the goal, and prune it (beyond the end of the committed trajectory) when it is found to be in collision.

5.6 Closed-loop Pallet Manipulation

A fundamental capability of our system is the ability to engage pallets, both from truck beds and from the ground. Pallet manipulation is initiated with a local frame volume of interest containing the pallet or truck. As we discuss shortly, this volume of interest can come from an image-space segmentation provided by the user or derived from an autonomous vision-based detection. In either case, image segmentation together with known camera calibration, results in a prior over the pallet or truck's location.

Figure 7: The robot detects the presence of the truck bed and pallet and maintains estimates of their geometry throughout engagement using LIDARs mounted to the tines and carriage.

The challenge to picking up palletized cargo is to accurately control the dynamics of the non-holonomic forklift moving on uneven terrain while inserting the tines into the slots of the pallet. Further, the pallet and truck geometry (e.g., height, width) vary, as do their cargo. In order to operate safely, the forklift must maintain accurate estimates of their geometry using limited onboard sensing. However, the physical structure of the pallet is sparse, which results in few LIDAR returns from the pallet itself and most from the (unknown) load and the pallet's supporting surface. The surfaces of the truck bed similarly provide few returns. Further complicating the estimation problem is the fact that, while the carriage and tines to which the LIDARs are mounted are rigid, they are not rigidly attached to the forklift, making extrinsic calibration difficult.

Unlike many approaches to small-object manipulation, it isn't possible to exploit the compliance of the manipulator or to use feedback control strategies to ease insertion. At 2700 kg, the forklift can easily exert significant force. Attempting to insert the tines incorrectly can damage the cargo before there is an opportunity to detect and correct for the error. Further, the tines are rigid and cannot be instrumented with tactile sensors necessary to enable feedback control.

Walter et al. originally presented our pallet estimation and manipulation capabilities, which address these challenges through a coupled perception and control strategy (Walter et al., 2010). At the core of our perception capability is a general technique for pattern recognition that quickly identifies linear structure within noisy 2D LIDAR scans. We use this algorithm to build a classifier that identifies pallets and trucks among clutter. The system feeds positive detections to a set of filters that maintain estimates for the pallet and truck poses throughout the process of engagement. We use these estimates in a steering controller that servos the pose of the vehicle and tines.

5.6.1 Fast Closest Edge Fitting for Pallet and Truck Detection

Most pallets and trucks have distinctive features, namely linear segments forming the sides and two slots of the pallet and the flat horizontal and vertical faces of the truck bed. Our strategy is to identify these features in individual scans from the tine- and carriage-mounted LIDARs and classify them as corresponding to a pallet, truck, or as outliers. The challenge is to detect these edges despite the noise present in the laser range finder data. Inspired by kernel-based solutions

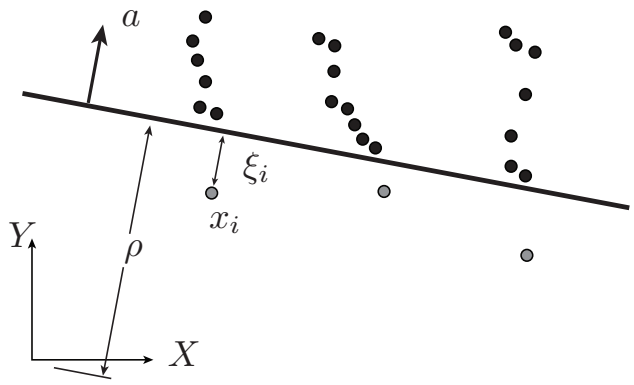


Figure 8: A graphical representation of the closest edge detection problem for 2D laser returns from a pallet face (e.g., Fig. 9). The three grey points are outliers with respect to the line (a, ρ).

to similar problems in machine learning (Schölkopf and Smola, 2002), we formulate the problem as a linear program for which the dual form can be solved in $O(n \min\{\log n, \nu\})$ time, where n is the number of points and ν is a problem-specific parameter. This is particularly advantageous for real-time applications when compared with the $O(n^{3.5})$ worst-case cost of the standard interior point solution (Boyd and Vandenberghe, 2004).

Let $\mathcal{X} = \{x_i\}_{i \in \mathcal{I}}$, where $\mathcal{I} = \{1, 2, \ldots, n\}$, be the set of 2D points from the current scan (Fig. 8). Without loss of generality, assume that the sensor lies at the origin and denote the orientation by the normal vector $a \in \mathbb{R}^2$. Assuming that the orientation is known, the problem is to find the distance ρ from the origin to the line that separates all data points, except a few outliers, from the origin. More precisely, for all points $x_i \in \mathcal{X}$, except a few outliers, $\langle a, x_i \rangle \geq \rho$ holds, where $\langle \cdot, \cdot \rangle$ denotes the dot product. Let $\xi_i = \max(\rho - \langle a, x_i \rangle, 0)$ be the distance from a point x_i to the separating line (projected along a) for outliers, and zero for inliers.

Given a line described by a normal a and distance ρ, a point x_i with $\xi_i > 0$ is an *outlier* with respect to the line (a, ρ). We formulate the *closest edge detection problem* as the maximization of $\rho - C \sum_{i \in \mathcal{I}} \xi_i$, where C is a constant problem-dependent parameter. This seeks to maximize the distance ρ of the separating line to the origin while minimizing the total distance $\sum_{i \in \mathcal{I}} \xi_i$ of the outliers to the line. Notice that $C = 0$ results in each data point being an outlier, while $C \to \infty$ allows no outliers in a feasible solution. We allow for the presence of outliers by formulating the problem as follows:

$$\text{maximize} \quad \rho - \frac{1}{\nu} \sum_{i \in \mathcal{I}} \xi_i, \tag{2a}$$

$$\text{subject to} \quad d_i \geq \rho - \xi_i, \qquad \forall i \in \mathcal{I}, \tag{2b}$$

$$\xi_i \geq 0, \qquad \forall i \in \mathcal{I}, \tag{2c}$$

where $\rho \in \mathbb{R}$ and $\xi_i \in \mathbb{R}$ are the decision variables, $\nu \in \mathbb{R}$ is a parameter that serves as an upper bound on the number of outliers, and $d_i = \langle a, x_i \rangle$ is the distance of point x_i to the origin when projected along a.

For computational purposes, we consider the dual of the linear program (2):

$$\text{minimize} \quad \sum_{i \in \mathcal{I}} d_i \lambda_i, \tag{3a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}} \lambda_i = 1, \qquad \forall i \in \mathcal{I}, \tag{3b}$$

$$0 \leq \lambda_i \leq \frac{1}{\nu}, \qquad \forall i \in \mathcal{I}, \tag{3c}$$

where λ_i are called the dual variables. Let $(\rho^*, \xi_1^*, \ldots, \xi_n^*)$ be the optimal solution to the linear program (2) and $(\lambda_1^*, \ldots, \lambda_n^*)$ be the optimal solution of the dual linear program (3). The optimal primal solution is recovered from the dual solution as $\rho^* = \sum_{i \in \mathcal{I}} \lambda_i^* d_i$.

Algorithm 5, DUALSOLVE solves the dual linear program in time $O(n \min\{\log n, \nu\})$. The algorithm employs two primitive functions: SORT considers a set $\{y_i\}_{i \in \mathcal{I}}$ and returns a sorted sequence of indices $\mathcal{J}$ such that $y_{\mathcal{J}(j)} \leq y_{\mathcal{J}(j+1)}$, and MIN returns the index j of the minimum element in a given set. The elementary operations in DUALSOLVE require only additions and multiplications, without the need to compute any trigonometric functions, which makes it computationally efficient in practice. We use this algorithm in the DISTFIND$(\nu, a, \mathcal{X})$ procedure (Algorithm 6) to solve the original linear program (2). Clearly, DISTFIND also runs in time $O(n \min\{\log n, \nu\})$.

Algorithm 5: DUALSOLVE $(\nu, a, \mathcal{X})$

for *all* $i \in \mathcal{I}$ **do**
  $\lambda_i := 0$;
for *all* $i \in \mathcal{I}$ **do**
  $d_i :=< a, x_i >$;
$\mathcal{D} := \{d_i\}_{i \in \mathcal{I}}$;
if $\log |\mathcal{D}| < \nu$ **then**
  $\mathcal{J} :=$ SORT$(\mathcal{D})$;
  for $j := 1$ *to* $\lfloor \nu \rfloor$ **do**
    $\lambda_{\mathcal{J}(j)} := 1/\nu$;
  $\lambda_{\mathcal{J}(\lfloor \nu \rfloor + 1)} := 1 - \lfloor \nu \rfloor / \nu$;
else
  for $i := 1$ *to* $\lfloor \nu \rfloor$ **do**
    $j :=$ MIN$(\mathcal{D})$;
    $\lambda_j := 1/\nu$;
    $\mathcal{D} := \mathcal{D} \setminus \{d_j\}$;
  $j :=$ MIN$(\mathcal{D})$;
  $\lambda_j := 1 - \lfloor \nu \rfloor / \nu$;
return $\{\lambda_i\}_{i \in \mathcal{I}}$

Now, we relax the assumption that the orientation is known. We employ DUALSOLVE a constant number of times for a set $\{a_i\}_{i \in \{1,2,\ldots,N\}}$ of normal vectors that uniformly span an interval of

Algorithm 6: DISTFIND$(\nu, a, \mathcal{X})$

for *all* $i \in \mathcal{I}$ **do**
  $d_i :=< a, x_i >$;
$\{\lambda_i\}_{i\in\mathcal{I}} :=$ DUALSOLVE$(\nu, a, \mathcal{X})$;
$\rho := \sum_{i\in\mathcal{I}} \lambda_i d_i$

possible orientations (Algorithm 7). After each invocation of DUALSOLVE, the method computes a weighted average z_i of the data points using the dual variables returned from DUALSOLVE as weights. Using a least squares method, a line segment is fitted to the resulting points $\{z_i\}_{i\in\{1,2,\ldots,N\}}$ and returned as the closest edge as the tuple (z', a', w'), where z' is the position of the mid-point, a' is the orientation, and w' is the width of the line segment.

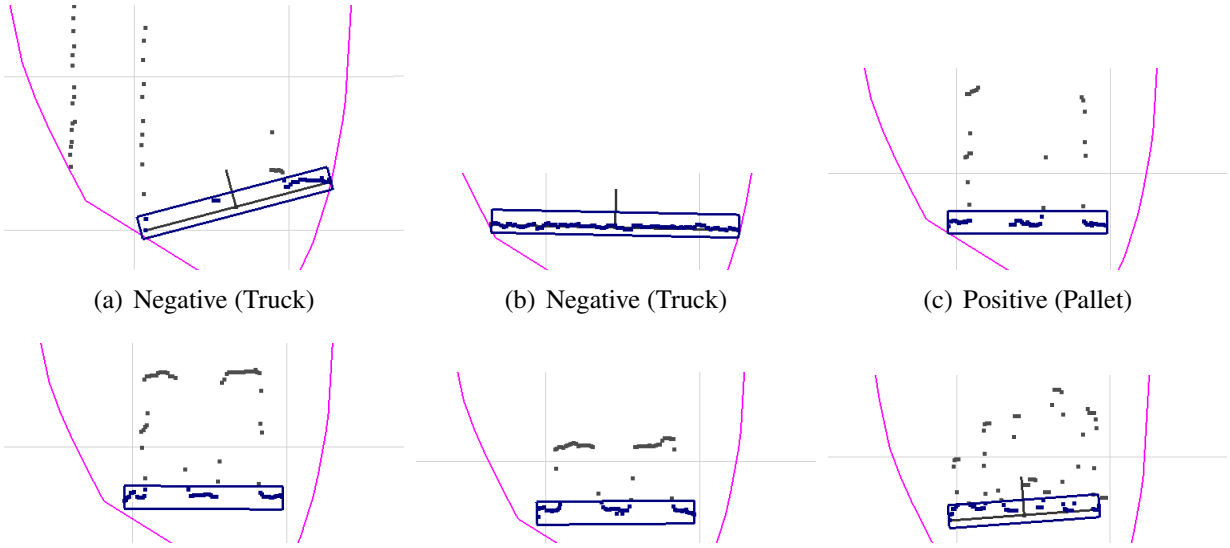
Algorithm 7: EDGEFIND$(\nu, \mathcal{X}, \theta_{\min}, \theta_{\max}, N)$

for $j := 1$ *to* N **do**
  $\theta := \theta_{\min} + (\theta_{\max} - \theta_{\min})j/N$;
  $a := (\cos(\theta), \sin(\theta))$;
  $\{\lambda_i\}_{i\in\mathcal{I}} :=$ DUALSOLVE$(\nu, a, \mathcal{X})$;
  $z_j := \sum_{i\in\mathcal{I}} \lambda_i x_i$;
$(z', a', w') :=$ LINEFIT$(\{z_j\}_{j\in\{1,2,\ldots,N\}})$;
return (z', a', w')

5.6.2 Pallet and Truck Classification and Estimation

Pallet and truck perception algorithms run DISTFIND or EDGEFIND over sets $\{\mathcal{X}_k\}_{k\in\mathcal{K}}$ of data points, one for each tine- and carriage-mounted LIDAR, to identify linear segments within the volume of interest. We then use these segments to compute several features that we use as input to a classifier to identify positive detections of pallets and truck beds. These features include relative distances and orientations between linear segments that encode different geometric properties of pallets and truck beds (e.g., width, slot width, depth, height, etc.). These features are calculated based upon single scans for pallets and pairs of scans from the left- and right-mounted vertical LIDARS for trucks. For a more detailed description of these features and the way in which they are computed, we refer the reader to our earlier work (Walter et al., 2010).

Features are fed into a classifier that, upon detecting a pallet within a scan (Fig. 9) or a truck in a scan pair (Fig. 10), yields an estimate of the object's pose and geometry. This includes the position, orientation and the location and the slot width of pallets, and the 2D position, orientation, and height off the ground of the truck bed. The method then uses these estimates to initialize a Kalman filter with a conservative prior. The filter employs subsequent detections as observations to track the pose and geometry of the truck and pallets online.



(a) Negative (Truck) (b) Negative (Truck) (c) Positive (Pallet)

(d) Positive (Pallet) (e) Positive (Pallet) (f) Negative (Load)

Figure 9: Output of the pallet detection algorithm as a pallet on a truck bed is being actively scanned, sorted by increasing height. (a-b) LIDAR returns from the undercarriage and the truck bed are rejected as pallet candidates. (c-e) LIDAR returns from the pallet face are identified as the pallet. (f) The load on the pallet is correctly ruled out as a candidate pallet face.

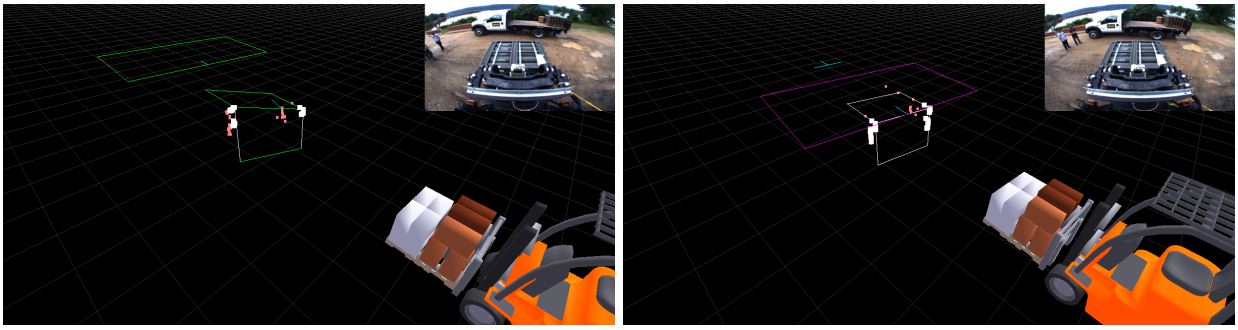


Figure 10: The system classifies pairs of vertical scans as being negative (left, in green) or positive (right, in pink) detections of a truck bed, and uses the latter to estimate the truck's pose.

5.6.3 Manipulation Controller

Given the filter estimates for the truck and pallet pose, the manipulation controller steers the robot from an initial position and heading to a final position and heading. The algorithm is tailored and tuned for precise pallet engagement operations.

Let z_{initial} and a_{initial} be the robot's initial position and orientation, where z_{initial} is a coordinate Euclidean plane and a_{initial} is a normalized two-dimensional vector. Similarly, let z_{final} and a_{final} be the desired final position and orientation of the robot. (In our application, z_{final} and a_{final} represent the pallet position and orientation.) Without loss of generality, let $z_{\text{final}} = (0, 0)$ and $a_{\text{final}} = (1, 0)$ be oriented toward the X-axis (Fig. 11). Similarly, let e_y be the distance between z_{initial} and z_{final}

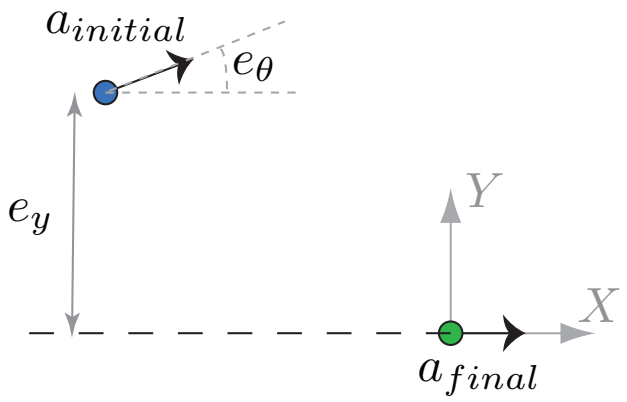


Figure 11: Illustration of the controller algorithm.

along the direction orthogonal to a_{final}, and let e_θ be the angle between the vectors a_{initial} and a_{final}, $e_\theta = \cos^{-1}(a_{\text{initial}} \cdot a_{\text{final}})$. Finally, let δ be the steering control input to the robot. In this work, we use the following steering control strategy for pallet engagement operations:

$$\delta = K_y \tan^{-1}(e_y) + K_\theta e_\theta, \tag{4}$$

where K_y and K_θ are controller parameters. Assuming a Dubins vehicle model (Dubins, 1957) of the robot

$$\dot{z} = (\cos\theta, \sin\theta) \tag{5a}$$

$$\dot{\theta} = \tan^{-1}(\delta), \tag{5b}$$

the nonlinear control law (4) can be shown to converge such that $e_y \to 0$ and $e_\theta \to 0$ holds, if $-\pi/2 \leq e_\theta \leq \pi/2$ is initially satisfied (Hoffmann et al., 2007).

5.7 Object Reacquisition

Critical to the effectiveness of the forklift is its ability to understand and execute long task sequences that the user commands using natural language speech (e.g., "Pick up the pallet of tires and put them on the truck"). This requires that the robot be able to robustly detect the presence of objects in the environment (e.g., "the pallet of tires" and "the truck") over long excursions in both space and time. Achieving the level of recall necessary to persistently reacquire objects is challenging for the forklift and other robots that operate with imprecise knowledge of their absolute location within dynamic, uncertain environments.

We developed a one-shot appearance learning algorithm (Walter et al., 2012) that automatically builds and maintains a model of the visual appearance of each user-indicated object in the environment. This enables robust object recognition under a variety of different lighting, viewpoint, and object location conditions. The user provides a single manual segmentation by circling it in an image from one of the forklift's cameras shown on the tablet interface. The system bootstraps on this single example to automatically generate a multiple-view, feature-based object model that

captures variations in appearance due to changes in viewpoint, scale, and illumination. This automatic and opportunistic model learning enables the robot to recognize the presence of objects to which the user referred, even for viewpoints that are temporally and spatially far from those of the first training example.

Given a segmentation cue, the algorithm constructs a *model* $\mathcal{M}_i$ that represents the visual appearance of an object i as a collection of views, $\mathcal{M}_i = \{v_{ij}\}$. We define a *view* v_{ij} as the appearance of the given object at a single viewpoint and time instant j (i.e., as observed by a camera with a particular pose at a particular moment). A view consists of a 2D constellation of SIFT keypoints (Lowe, 2004) comprised of an image pixel position and a local descriptor. The system initializes a new model $\mathcal{M}_i$ for each indicated object, using the set of SIFT features that fall within the gesture in that particular frame to form the new model's first view v_{i1}. The method then searches new camera images for each model and produces a list of visibility hypotheses based on visual similarity and geometric consistency of keypoint constellations. New views are automatically added over time as the robot moves; thus the collection of views opportunistically captures variations in object appearance due to changes in viewpoint and illumination.

5.7.1 Single-View Matching

As the robot acquires new images of the environment, the system (Algorithm 8) continuously searches for instances of each model $\mathcal{M}_i$ within the scene, producing a visibility hypothesis and associated likelihood for the presence and location of each view. For each view v_{ij}, our algorithm matches the view's set of descriptors $\mathcal{F}_{ij}$ with those in the image at time t $\mathcal{F}_t$ to produce a set of point-pair correspondence candidates $\mathcal{C}_{ijt}$ (Line 2). We evaluate the similarity s_{pq} between a pair of features p and q as the normalized dot product between their descriptor vectors f_p and f_q, $s_{pq} = \sum_k (f_{pk} f_{qk}) / \|d_p\| \|d_q\|$. We exhaustively compute all similarity scores and collect in $\mathcal{C}_{ijt}$ at most one pair per feature in $\mathcal{F}_{ij}$, subject to a minimum threshold. Table 1 enumerates the parameter settings that the forklift uses for reacquisition.

Since many similar-looking objects may exist in a single image, $\mathcal{C}_{ijt}$ may contain a significant number of outliers and ambiguous matches. We therefore enforce geometric consistency on the constellation by means of random sample consensus (RANSAC) (Fischler and Bolles, 1981) with a plane projective homography H as the underlying geometric model (Hartley and Zisserman, 2004). Our particular robot employs wide-angle camera lenses that exhibit noticeable radial distortion. Before applying RANSAC, we correct the distortion of the interest points (Line 3) to account for deviations from standard pinhole camera geometry, which enables the application of a direct linear transform to estimate the homography.

With each RANSAC iteration, we select four distinct (un-distorted) correspondences $\hat{\mathcal{C}}^u_{ijt} \in \mathcal{C}^u_{ijt}$ with which we compute the induced homography $\hat{H}$ between the current image and the view v_{ij} (Line 7). We then apply the homography to all matched points within the current image, re-distort the result, and classify each point as an inlier or outlier according to its distance from its image counterpart using a pre-specified threshold t in pixel units (Lines 12 and 12). As the objects are non-planar, we use a loose value for this threshold in practice to accommodate deviations from planarity due to motion parallax.

Algorithm 8: Single-View Matching

Input : A model view v_{ij} and camera frame $\mathcal{I}_t$
Output: $\mathcal{D}_{ijt} = \left(H^\star_{ij}, c^\star_{ij}\right)$

1 $\mathcal{F}_t := \{(x_p, f_p)\} \leftarrow \texttt{SIFT}(\mathcal{I}_t)$;
2 $\mathcal{C}_{ijt} := \{(s_p, s_q)\} \leftarrow \texttt{FeatureMatch}(\mathcal{F}_t, \mathcal{F}_{ij})\ s_p \in \mathcal{F}_t, s_q \in \mathcal{F}_{ij}$;
3 $\forall\, x_p \in \mathcal{C}_{ijt},\ x^u_p \leftarrow \texttt{UnDistort}(x_p)$;
4 $\mathcal{H}^\star_{ijt} := \{H^\star_{ijt}, d^\star_{ijt}, \tilde{\mathcal{C}}^\star_{ijt}\} \leftarrow \{\}$;
5 **for** $n = 1$ *to* N **do**
6 Randomly select $\hat{\mathcal{C}}^u_{ijt} \in \mathcal{C}^u_{ijt}$, $|\hat{\mathcal{C}}^u_{ijt}| = 4$;
7 Compute homography $\hat{H}$ from (x^u_p, x^u_q) in $\hat{\mathcal{C}}^u_{ijt}$;
8 $\mathcal{P} \leftarrow \{\}, \hat{d} \leftarrow 0$;
9 **for** $\left(x^u_p, x^u_q\right) \in \mathcal{C}^u_{ijt}$ **do**
10 $\hat{x}^u_p \leftarrow \hat{H} x^u_p$;
11 $\hat{x}_p \leftarrow \texttt{Distort}(\hat{x}^u_p)$;
12 **if** $d_{pq} = |x_q - \hat{x}_p| \leq t_d$ **then**
13 $\mathcal{P} \leftarrow \mathcal{P} + (x_p, x_q)$;
14 $\hat{d} \leftarrow \hat{d} + \min(d_{pq}, t_d)$;
15 **if** $\hat{d} < d^\star_{ij}$ **then**
16 $\mathcal{H}^\star_{ijt} \leftarrow \{\hat{H}, \hat{d}, \mathcal{P}\}$;
17 $c^\star_{ijt} = |\tilde{\mathcal{C}}^\star_{ijt}|/(|v_{ij}| \min(\alpha|\tilde{\mathcal{C}}^\star_{ijt}|, 1)$;
18 **if** $c^\star_{ijt} \geq t_c$ **then**
19 $\mathcal{D}_{ijt} \leftarrow \left(H^\star_{ijt}, c^\star_{ijt}\right)$;
20 **else**
21 $\mathcal{D}_{ijt} \leftarrow ()$;

RANSAC establishes a single best hypothesis for each view v_{ij} that consists of a homography $H^\star_{ijt}$ and a set of inlier correspondences $\tilde{\mathcal{C}}^\star_{ijt} \in \mathcal{C}_{ijt}$. We assign a confidence value to the hypothesis $c_{ijt} = |\text{inliers}|/(|v_{ij}| \min(\alpha|\text{inliers}|, 1)$ (Line 17) that compares the number of inliers to total points in v_{ij}. If the confidence is sufficiently high per a user-defined threshold t_c, we output the hypothesis.

5.7.2 Multiple-View Reasoning

The single-view matching procedure may produce a number of match hypotheses per image and does not prohibit detecting different instances of the same object. Each object model possesses one or more distinct views, and it is possible for each view to match at most one location in the image. To address this, the algorithm reasons over these matches and their associated confidence scores to resolve potential ambiguities, thereby producing at most one match for each model and reporting its associated image location and confidence.

Table 1: Reacquisition Parameter Settings.

Parameter	Description	Setting
$s_{pq}^{\min}$	Minimum dot product (i.e., maximum allowable distance) between SIFT feature matches (Alg. 8, line 2).	0.9
N	Number of RANSAC iterations for single-view matching (Alg. 8, line 5).	600
t_d	Maximum distance in pixels of projected interest points for RANSAC inliers (Alg. 8, line 12).	10.0 px
t_c	Minimum confidence threshold for homography validation (Alg. 8, line 18).	0.10
t_c^{match}	Minimum confidence threshold for a visible model match.	0.10
$h_{\min}$	Minimum scale variation between an existing model view and a new view for model augmentation.	1.20
$d_{\min}$	Minimum displacement of the robot between new and existing views for model augmentation.	0.50 m

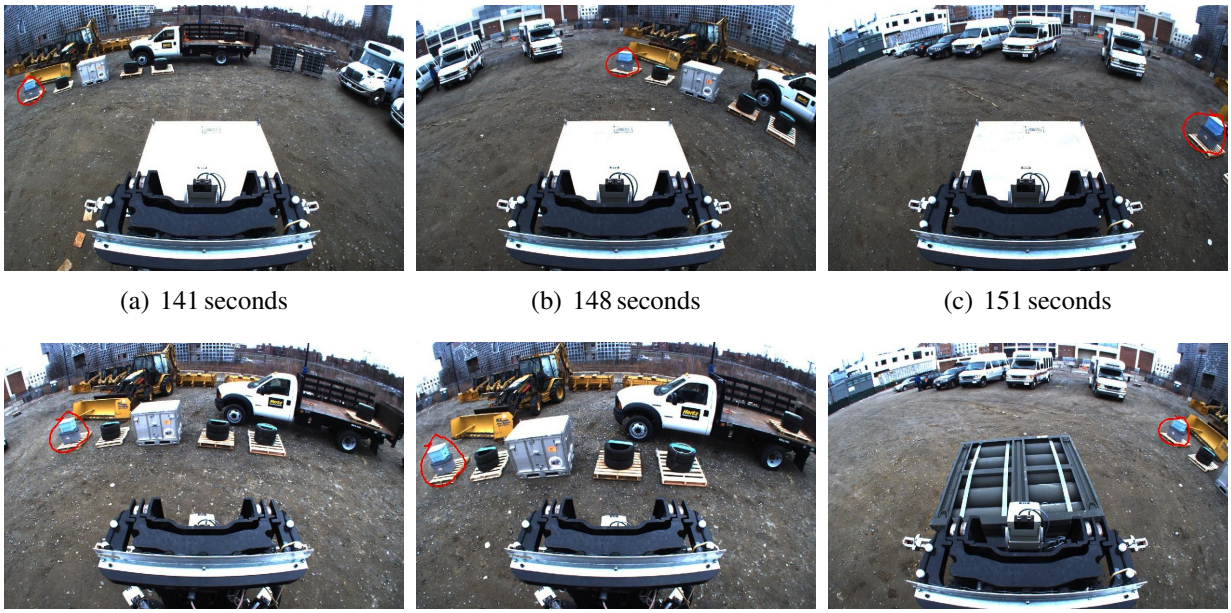
First, all hypotheses are collected and grouped by object model. To each *active* model (i.e., a model for which a match hypothesis has been generated), we assign a confidence score equal to that of the most confident view candidate. If this confidence is sufficiently high as specified by a threshold t_c^{match}, we consider the model to be visible and report its current location, which is defined as the original 2D gesture region transformed into the current image by the match homography associated with the hypothesis.

While this check ensures that each model matches no more than one location in the image, we do not impose the restriction that a particular image location match at most one model. Indeed, it is possible that running the multiple-view process on different models results in the same image location matching different objects. However, we have not found this to happen in practice, which we believe to be a result of surrounding contextual information captured within the user gestures.

5.7.3 Model Augmentation

As the robot navigates within the environment, an object's appearance changes due to variations in viewpoint and illumination. Furthermore, the robot makes frequent excursions—for example, moving cargo to another location in the warehouse—that result in extended frame cuts. When the robot returns to the scene, it typically observes objects from a different vantage point. Although SIFT features tolerate moderate appearance variability due to some robustness to scale, rotation, and intensity changes, the feature and constellation matches degenerate with more severe scaling and 3D perspective effects.

In order to retain consistent object identity over longer time intervals and larger displacements, the algorithm periodically augments each object model by adding new views whenever an object's



(a) 141 seconds (b) 148 seconds (c) 151 seconds

(d) 288 seconds (e) 292 seconds (f) 507 seconds

Figure 12: New views of an object annotated with the corresponding reprojected gesture. New views are added to the model when the object's appearance changes, typically as a result of scale and viewpoint changes. Times shown indicate the duration since the user provided the initial gesture. Note that the object is out of view during the periods between (c) and (d), and (e) and (f), but is reacquired when the robot returns to the scene.

appearance changes significantly. In this manner, the method opportunistically captures the appearance of each object from multiple viewing angles and distances. This increases the likelihood that new observations will match one or more views with high confidence and, in turn, greatly improves the overall robustness of reacquisition. Figure 12 depicts views of an object that are automatically added to the model based upon appearance variability.

The multiple-view reasoning signals a positive detection when it determines that a particular model $\mathcal{M}_i$ is visible in a given image. We then examine each of the matching views v_{ij} for that model and consider both the robot's motion and the geometric image-to-image change between the view v_{ij} and the associated observation hypothesis. In particular, we evaluate the minimum position change $d_{\text{min}} = \min_j \|p_j - p_{\text{cur}}\|$ between the robot's current position p_{cur} and the position p_j associated with the j^{th} view. We also consider the minimum 2D geometric change $h_{\text{min}} = \min_j \text{scale}(H_{ij})$ corresponding to the overall 2D scaling implied by match homography H_{ij}. If both d_{min} and h_{min} exceed pre-specified thresholds, signifying that no current view adequately captures the object's current image scale and pose, then we create a new view for the model $\mathcal{M}_i$ using the hypothesis with the highest confidence score.

In practice, the system instantiates a new view by generating a *virtual gesture* that segments the object in the image. SIFT features from the current frame are used to create a new view, and this view is then considered during single-view matching (Section 5.7.1) and during multiple-view reasoning (Section 5.7.2).

Figure 13: The handheld tablet provides one means for the user to interact with the robot. The tablet enables the user to visualize the robot's situational awareness and to convey task-level directives through a combination of pen-based gestures and natural language utterances.

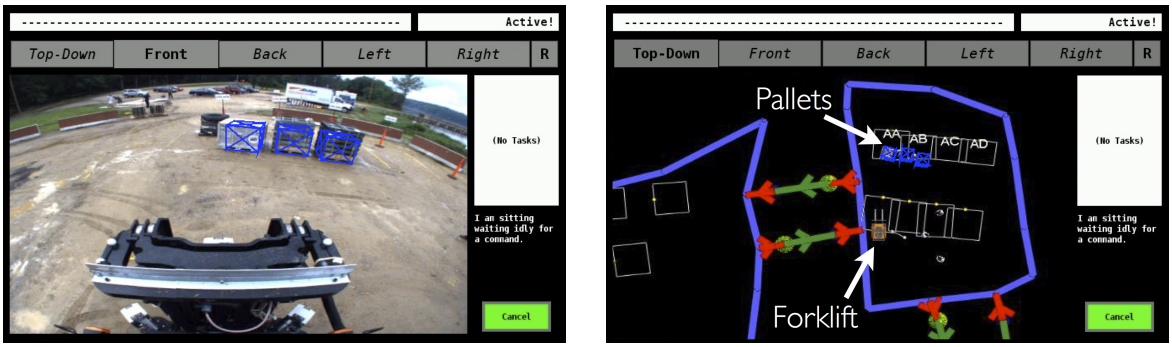
5.8 User Interface

One of the most fundamental design requirements for our system is that it must afford an intuitive interface that allows existing personnel to quickly and efficiently command the robot with minimal training. To that end, we worked extensively with soldiers and civilians within military logistics throughout the iterative design process. Taking into account their feedback and the results of their tests, we developed a multimodal command interface that enables humans to issue high-level tasks to the robot using a combination of simple pen-based gestures made on a handheld tablet, and utterances spoken using natural language.

5.8.1 Graphical User Interface

Our interface (Correa et al., 2010) operates on a Nokia N810 Internet Tablet (Fig. 13) that provides a built-in microphone for speech input, alongside a touchscreen and stylus. The graphical user interface presents images from one of the forklift's four cameras that are annotated with the robot's object-level knowledge about its surround (e.g., obstacles, pedestrians, and pallets), thus providing the user a 360° view of the area around the robot. Additionally, the interface offers a rendered overhead view of the robot's local environment that depicts its location in the topological map along with an indication of the robot's awareness of nearby objects. For both the overhead and camera views, we deliberately choose to provide a high-level abstraction of the robot's world model over rendering raw sensor data (as others have done) to minimize the cognitive burden on the user that results from having to interpret the data.

In addition to providing an indication of the robot's situational awareness, the tablet lists the queue of tasks that the forklift is to perform. The user can click on and cancel any of these tasks if needed. Additionally, the interface indicates the current status of the robot. Text boxes at the top of the screen display a variety of information, including the system's recognition of the latest



(a) Forward-Facing Camera Image

(b) Rendered Overhead View

Figure 14: The interface presents images from (a) each of the robot's four cameras as well as (b) an overhead view, each augmented with the robot's knowledge of objects in its surround.

utterance, the robot's operating mode (i.e., "Active," "Manual," or "Paused"), and a description of the robot's current action. For example, when the robot is approaching a pallet on the back of a truck as part of a pick-up task, the interface lists "Approaching truck" and "Active: Pickup."

5.8.2 Drawing on the World

By drawing on the canvas, the user can command the robot to move, pick up, and place pallets throughout the environment. Gestures have different meanings depending on their context. For example, circling a pallet is an instruction to pick it up, while circling a location on the ground or on the back of a truck is an instruction to put the pallet at the circled location. Drawing an "X" or a dot on the ground is an instruction to go there, while drawing a line is an instruction to follow the path denoted by the line.

The interface first classifies the shape of the user's sketch into one of six types: a dot, an open curve, circle, an "X", a pair of circles, or a circled "X". It then infers the context-dependent meaning of the shape, upon which we then refer to it as a *gesture*. The system recognizes shapes as in traditional sketch recognition systems: it records the timestamped point data that makes up each stroke and uses heuristics to compute a score for each possible shape classification based on stroke geometry. It then classifies the stroke as the highest-scoring shape. In order to classify shapes as gestures, the system must consider both what was drawn and what it was drawn on. We define the scene (i.e., the "context") as the collection of labeled 2D boxes that bound the obstacles, people, and pallets visible in the camera view. Reasoning over the scene is what differentiates this approach from ordinary sketch recognition.

Figure 15 shows an example of the use of context to disambiguate a stroke. In this example, the stroke (Fig. 15(a)) could be either a circular path gesture that avoids objects (Fig. 15(b)), a pallet pickup command (Fig. 15(c)), or a pallet placement command (not shown). The interpretation of the stroke depends upon its geometry as well as the context in which it was drawn. For example, when the projected size is too large to indicate a pallet drop-off location (or when the forklift isn't carrying a pallet), the system interprets the gesture as suggesting a circular path. We further incor-

porate scene context into the classification process so as to not rely solely upon stroke geometry for interpretation, making the algorithm less sensitive to gesture errors.

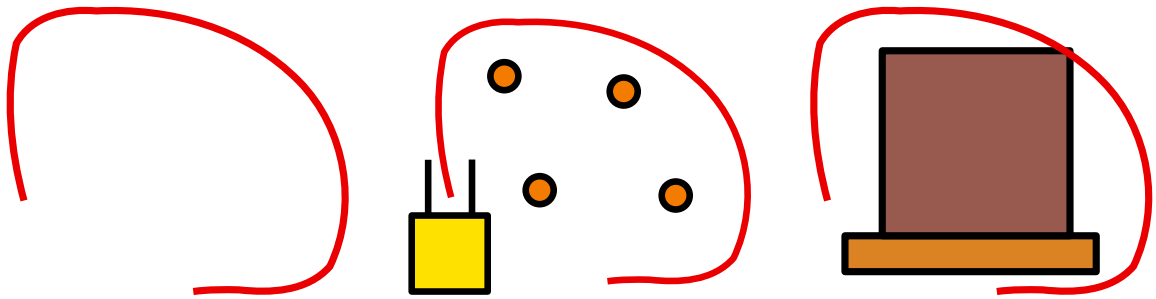


Figure 15: (a) A circular stroke alone is ambiguous: it could be either (b) a circular path or (c) a pallet manipulation command. Context determines the gesture's meaning.

This ability to disambiguate shapes into different gestures allows us to use fewer distinct shapes. As a result, the geometrical sketch recognition task is simplified, leading to higher gesture classification accuracy and robustness. The smaller lexicon of simple gestures also allows the user to interact with the system more easily (Correa et al., 2010).

5.8.3 Natural Language Understanding

In addition to context-aware gesture recognition, we developed and implemented a framework that enables people to issue commands using natural language speech (Tellex et al., 2011). For example, as an alternative to summoning the robot to the storage bay, circling the pallet to be picked up, summoning it to issuing, and then circling a placement location on the truck, the user can simply say "Pick up the tire pallet and put it on the truck." The tablet performs speech recognition onboard using the PocketSUMMIT library (Hetherington, 2007). The robot then uses the language understanding algorithm (running on the robot) to interpret the resulting text, and the vision-based reacquisition and pallet manipulation capabilities to execute the command. This directive requires a few seconds of the user's time at the outset after which the robot carries out the task, which may take tens of minutes. In contrast, the aforementioned gesture-based interaction requires that the user be involved throughout, albeit for only a few seconds at a time.

A challenge to understanding commands spoken in natural language is to correctly *ground* (i.e., map) the linguistic elements to their referents in the robot's model of its state and action space. We address this problem by learning a probabilistic model over the space of groundings for the linguistic constituents in the command. The task of interpreting the utterance is then one of performing inference in this model to identify the most likely plan. Underlying the model is the Generalized Grounding Graph (G^3), a probabilistic graphical model that we instantiate for each command based upon the hierarchical structure of natural language (Tellex et al., 2011). The model encodes the relationship between linguistic elements that refer to places and objects in the environment (e.g., "receiving" or "tire pallet"), spatial relations (e.g., "on the truck"), and actions that the robot

can perform (e.g., "pick up"), and the robot's model of its environment and available actions. We learn these relations by training the model on a corpus of natural language commands that are paired with hand-labeled groundings in the robot's state and action space. The task of interpreting a new command is then one of building the G^3 graph and performing inference on the unobserved random variables (i.e., the set of objects, places, relations, and actions in the robot's world model). For more information about the specific operation of the G^3 framework, we refer the reader to our previous work (Tellex et al., 2011).

5.9 Operation in Close Proximity to People

The robot must be able to operate safely in close proximity to people, whether it is a supervisor or bystanders who move unconstrained within the warehouse. To ensure that the robot operates safely and that its presence is accepted by people, we endowed the robot with a number of failsafe behaviors. By design, all potential robot trajectories conclude with the robot coming to a complete stop (even though this leg of the trajectory may not always be executed, particularly if another trajectory is chosen). Consequently the robot moves more slowly when close to obstacles (conservatively assumed to be people). The robot also signals its internal state and intentions, in an attempt to make people more accepting of its presence and more easily able to predict its behavior (Correa et al., 2010).

Humans can infer a great deal of information from the behavioral cues of others. If people are to accept the presence of a 2700 kg robot, they must be able to infer its current state and intent with similar ease. We allow the forklift to convey a subset of similar cues to people in its surround by equipping it with LED signage, strings of LED lights circling the body and mast, as well as speakers mounted to the roof.

5.9.1 Annunciation of Intent

The marquee lights that encircle the forklift encode the robot's state as colors, and imminent motion as moving patterns. For example, a chasing light pattern renders the intended direction of the robot, strobing forward when it is about to move forward and backward just before it is going to move in reverse. The LED signage displays short text messages describing the robot's current state (e.g., "Active," "Paused," or "Manual"), the current task (e.g., "Summon: Storage"), and any imminent actions (e.g., forward motion or mast lifting). When the forklift is transitioning to autonomous mode, the speakers sound a warning while the LED signs spell out the time remaining till the robot is autonomous (e.g., "Active in ... 3, 2 ,1"). Subsequently, the speakers announce tasks right before the robot is about to perform them. In our experience, we found that it is useful to be more verbose with signage and verbal annunciation at the outset and, once people become comfortable with the robot's presence, to reduce the rate at which the robot vocalizes its state and intent.

5.9.2 Awareness Display

The forklift also uses its annunciators to inform bystanders that it is aware of their presence. Whenever a human is detected in the vicinity, the marquee lights, consisting of strings of individually

Figure 16: The robot utilizes a combination of LED signs and LED lights to indicate that it is aware of people in its vicinity. The lights mimic the robot's gaze towards approaching pedestrians.

addressable LEDs, display a bright region oriented in the direction of the detection (Fig. 16). If the estimated motion track is converging with the forklift, the LED signage and speakers announce "Human approaching."

5.9.3 Shout Detection

The operator's interface provides several mechanisms by which they can quickly place the robot in a safe stopped state. Other people operating within the robot's surround, however, must also have a means of stopping the robot when necessary. For that reason, we equipped the robot with four array microphones facing forward, right, left, and rearward. The forklift continuously listens on each channel for shouted warnings. The input audio is transmitted to a streaming speech recognizer (Hetherington, 2007) that is configured to detect a set of key utterances that include several different stop commands. Further, the system continuously listens for a small set of utterances that direct summoning and manipulation tasks, allowing users to command the robot without the tablet interface (Chuangsuwanich et al., 2010; Chuangsuwanich and Glass, 2011). The challenge that we address is to develop a recognizer that provides a low false negative rate (namely, for shouted stop commands) without significantly hindering the true positive rate, despite the noisy environments in which the robot operates.

5.9.4 Subservience and Autonomy Handoff

We place more trust on human judgment than we do on the forklift in all cases. Accordingly, the robot relinquishes complete control to any person in the driver's seat. When a human closely approaches the robot, it pauses for safety. When a human enters the cabin and sits down, the robot detects his/her presence in the cabin through the report of a seat-occupancy sensor, the user's

Figure 17: The robot transporting cargo at the Fort Lee SSA.

contact with the mast or transmission levers, or the turning of the steering wheel. In this event, the robot reverts to behaving exactly as a manned forklift, completely ceding control. The system remains fully in manual mode until the user engages the parking brake, steps out of the cabin and away from the vehicle, and places it in autonomous mode using the handheld tablet interface. At this point, the system uses visual and audible cues to indicate to the user and any bystanders that it is resuming autonomous operation. Additionally, when the robot can not perform a difficult task and requests help, anyone can then climb in and operate it.

6 Deployment and Results

Over the course of three and a half years developing the system, we have performed numerous experiments and field trials. We have operated the system successfully in a number of different model and actual warehouse facilities. These include extensive testing at a model warehouse configured as a military Supply Support Activity (SSA) situated on an outdoor, gravel shuttle parking lot on the MIT campus. We also operated the system for several weeks at an in situ SSA located at Fort Belvoir in the summer of 2009, where we performed extensive end-to-end testing. Additionally, we deployed the robot for a two week period in an active SSA at Fort Lee (Fig. 17) where the robot operated alongside U.S. Army soldiers. In the Fort Belvoir and Fort Lee field trials, as well as those at the MIT test site, the robot was frequently commanded by military personnel, including soldiers active and knowledgeable in military logistics. In each case, the soldier was given a brief training session before using either the handheld interface or speaking directly to the robot to command various tasks.

In the following sections, we present experimental results that evaluate each of the critical subsystems described previously. We then discuss the end-to-end performance of our system during the various deployments.

6.1 Obstacle Avoidance and Anytime Optimal Motion Planning

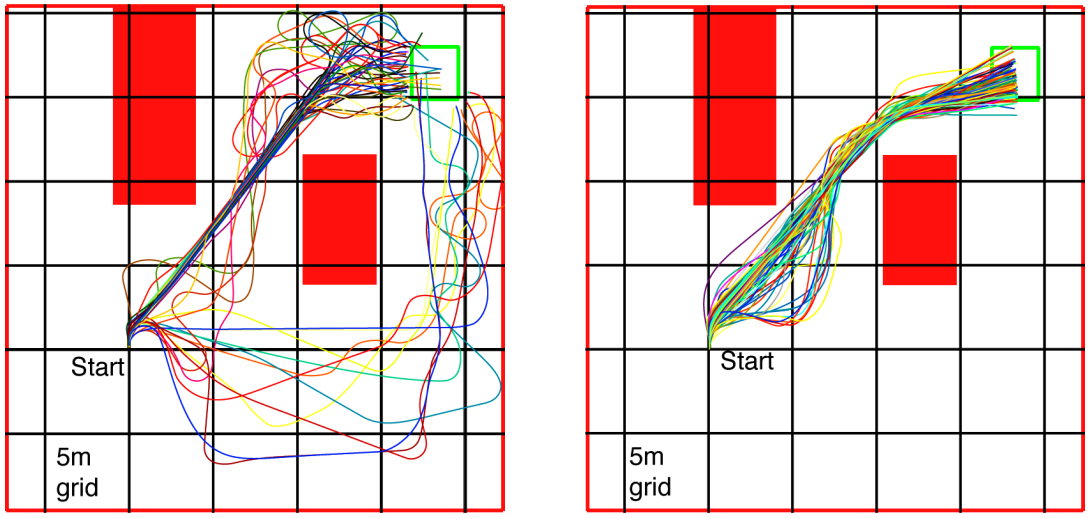
A key performance metric for the navigation subsystem is the ability to closely match the predicted trajectory with the actual path, as significant deviations may cause the actual path to become infeasible. Feasibility not only includes safe operation in the vicinity of bystanders and obstacles (i.e., avoiding collisions), but also requires that the vehicle does not reach the limits of its dynamic capabilities, namely risk roll-over when carrying significant load. During normal operation in several outdoor experiments, we recorded 97 different complex paths of varying lengths (6 m to 90 m) and curvatures. For each, we measure the average and maximum lateral error between the predicted and actual vehicle position over the length of the path. In all cases, the average prediction error does not exceed 0.12 m, while the maximum prediction error does not exceed 0.35 m.

We also test the robot's ability to achieve a variety of destination poses in the vicinity of obstacles of varying sizes. When a valid path exists, the forklift identifies and executes a collision-free route to the goal, navigating around pallets, vehicles, and pedestrians. In some case, the system rejects paths as being infeasible even though they were collision-free. We attribute this to the conservative 0.25 m safety buffer that the system places around each obstacle. We also test the robot's behavior when obstructed by a pedestrian (a mannequin), in which case the robot stops and waits for the pedestrian to move out of the way.

We evaluate the performance of our anytime optimal motion planning algorithm using both a high-fidelity simulator as well as with the forklift operating in our MIT warehouse. In both cases, we compare our algorithm with one based on an RRT. The simulation experiments consider the environment shown in Figure 18, in which the robot must find a feasible path from an initial pose in the lower left to the goal region indicated by the green box. We performed a total of 166 Monte Carlo simulation runs with our motion planner and 191 independent runs with the standard RRT. Both planners use branch-and-bound for tree expansion and maintain a committed trajectory. Both the RRT and RRT* were allowed to explore the state space throughout the execution period.

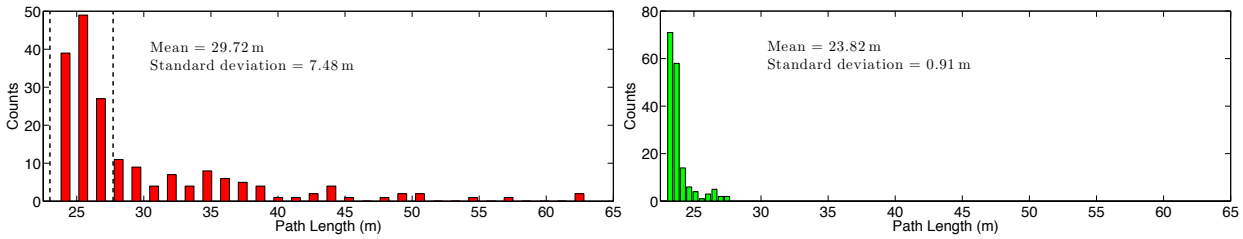
Figure 18 compares the paths that result from the RRT-based motion planner with those that the vehicle traversed using our anytime optimal motion planning algorithm. In some situations, the algorithm initially identifies a high-cost path that routes the vehicle to the right of the obstacles. In each case, however, the opportunistic rewiring reveals a shorter, lower-cost route between the obstacles. Subsequently, the branch-and-bound and online refinement further improve the plan into a more direct path as the vehicle navigates to the goal. While the RRT algorithm refines the initial plan using branch-and-bound, these improvements tend to be local in nature and do not provide significant improvements to the tree structure. Consequently, the RRT-based planner often gets "stuck" with a tree that favors longer paths that steer the forklift to the right of the obstacles.

Our algorithm exploits the execution period to modify the tree structure as it converges to the optimal path. This convergence is evident in the distribution over the length of the executed simulation trajectories (Fig. 19(b)) that exhibits a mean length (cost) of 23.82 m with a standard deviation of 0.91 m for the set of 166 simulations. For comparison, Figure 19(a) presents the distribution for the RRT planner. The mean path length for the 191 RRT simulations is 29.72 m with a standard deviation of 7.48 m. The significantly larger variance results from the RRT getting "stuck" refining



(a) RRT (b) RRT*

Figure 18: Vehicle paths traversed for (a) 65 simulations of the RRT and (a) 140 simulations with our RRT* planner.



(a) RRT (b) RRT*

Figure 19: Histogram plots of the executed path length for (a) RRT and (b) RRT* simulations. The vertical dashed lines in (a) depict the range of path lengths that result from the RRT* planner.

a tree with sub-optimal structure. The anytime RRT*, on the other hand, opportunistically takes advantage of the available execution time to converge to near-optimal paths.

These simulation experiments are useful in evaluating the convergence properties of the planner. We also performed a series of experiments with the forklift at our MIT model warehouse to validate the performance of the algorithm under real-world conditions. We operated the vehicle in the gravel lot among stationary and moving vehicles. We ran a series of tests in which the robot was tasked to navigate from a starting position in one corner to a 1.6 m goal region in the opposite corner while avoiding the obstacles. In each experiment, the robot began to plan motions immediately prior to tracking the committed trajectory with the controller.

Figure 20 presents the result of two different tests with the anytime motion planner as well as with the RRT. The plots depict the best trajectory as maintained by the planner at different points during the plan execution (false-colored by time). In the scenario represented in the upper left, our algorithm initially identifies a sub-optimal path that goes around an obstacle but, as the vehicle

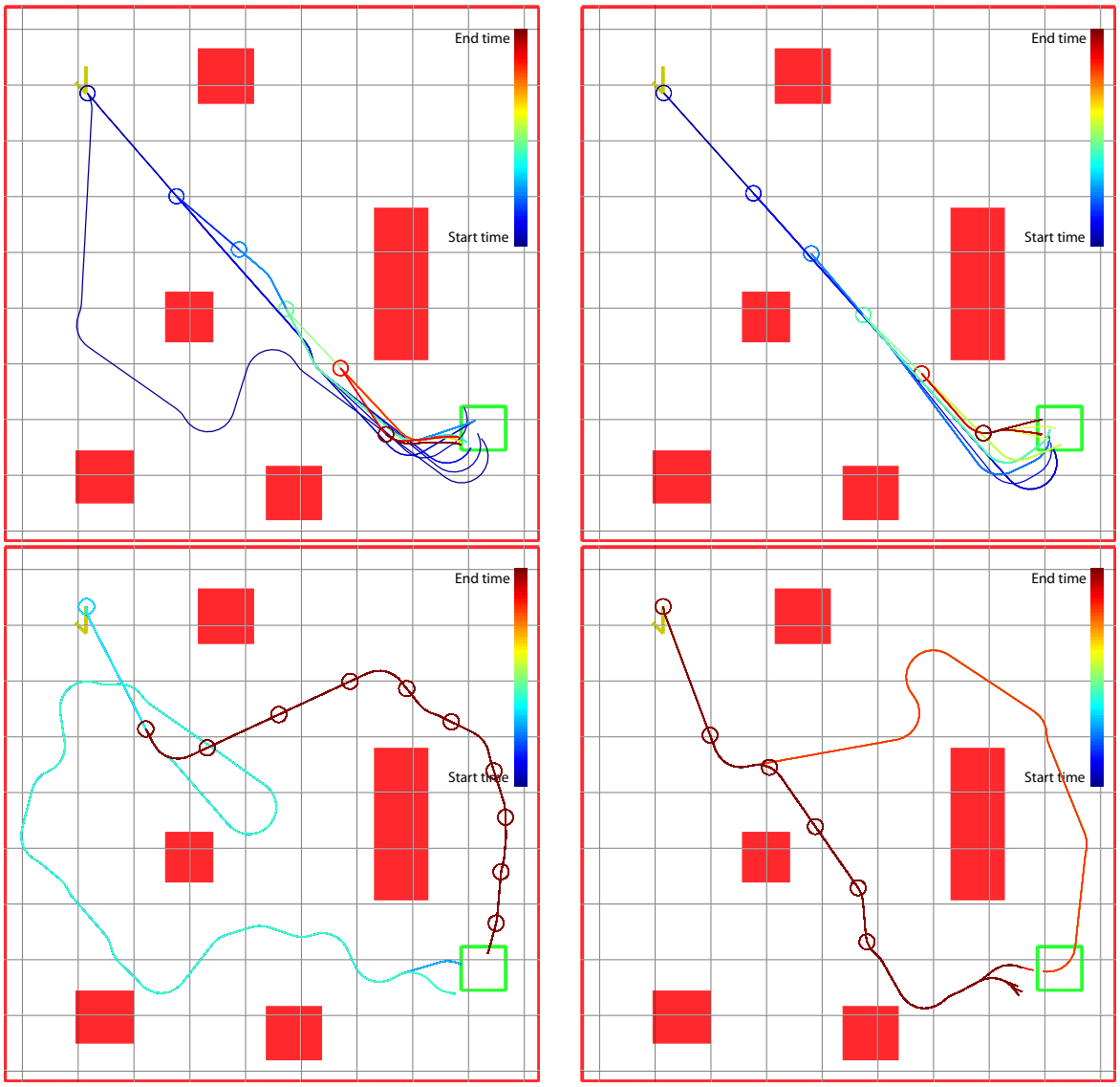


Figure 20: Two runs of our anytime optimal planner (top) compared with those of the RRT (bottom). The forklift started in the upper left and was tasked with driving to the goal region while avoiding obstacles. The trajectories indicate the planned paths at different points in time during the execution and are false-colored by time. Circles denote the initial position for each path.

begins to execute the path, the planner correctly refines the solution to a shorter trajectory. As the vehicle proceeds along the committed trajectory, the planner continues to rewire the tree, as evident in the improvements near the end of the execution, when the paths more directly approach the goal. Meanwhile, the RRT initially identifies paths that take unnecessarily high-cost routes around obstacles. After moving a few meters, the planner discovers a shorter path but the structure of the tree biases the RRT towards refinements that are only local in nature.

6.2 Pallet Engagement: Estimation and Manipulation

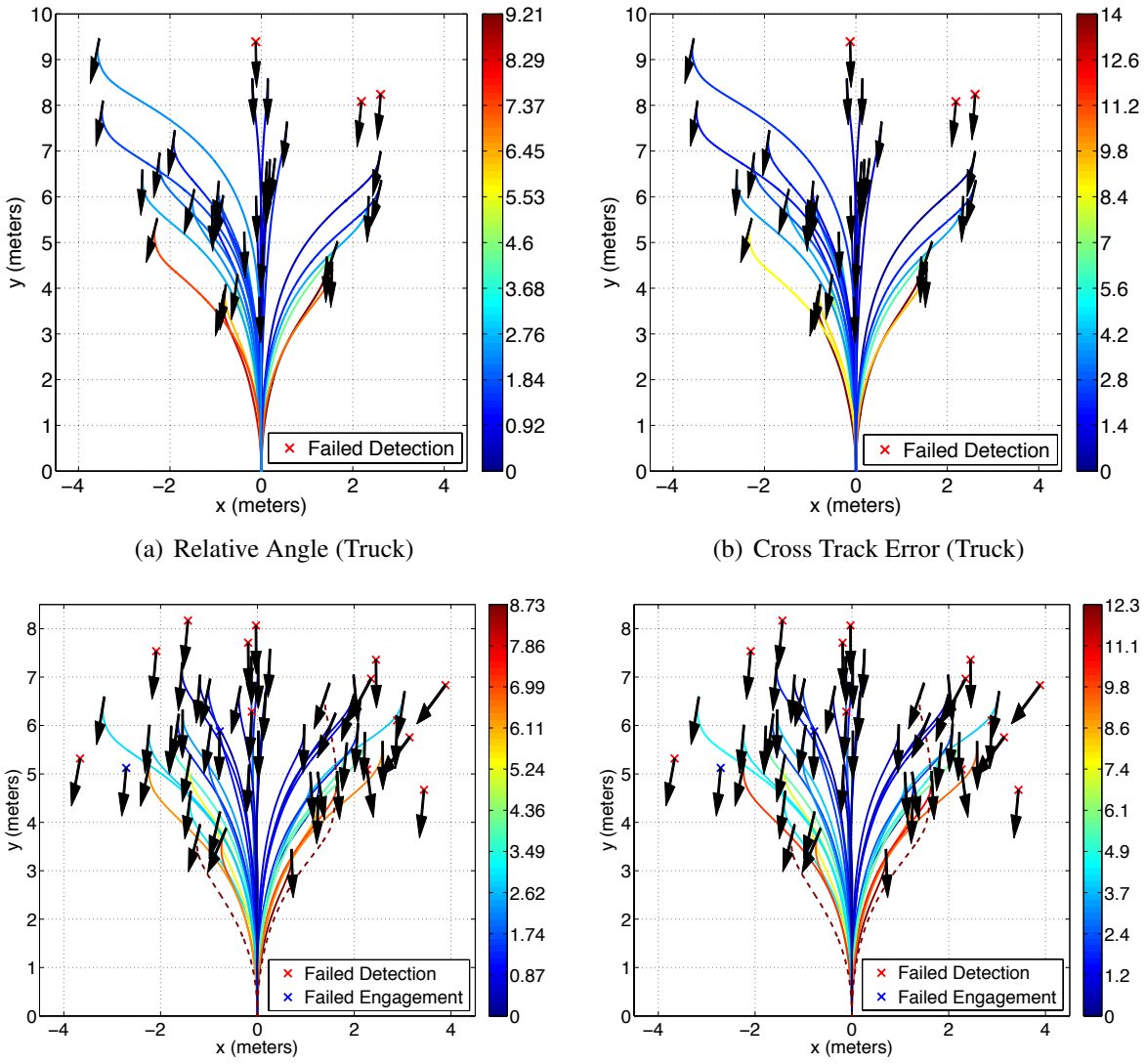
Next, we present an evaluation of the pallet estimation and manipulation algorithms. This is based on extensive testing within two of the outdoor warehouse environments. We commanded the robot to pick up pallets from different locations on the ground as well as from truck beds. We recorded the lateral position and orientation of the robot with respect to the pallet in each test as reported by the robot's dead-reckoning module. Note that the experiments were conducted with different types of pallets that, within each type, exhibited varying geometry (i.e., width, slot location, and slot width). The pose of the pallet relative to the truck and the truck's pose relative to the forklift also varied.

Figure 21 shows a plot of the successful and failed pallet pickup tests, together with the final relative angle and cross track error in each experiment (see Figure 22 for histograms). Note that most of the failures are due to unsuccessful pallet detection, and they occur when the robot starts longitudinally 7.5 m and/or laterally 3 m or more away from the pallet. In the majority of these cases, the robot's vantage point together with the sparseness of the pallet structure were such that the laser range finder yielded few returns from the pallet face. In the cases in which the pallet was visible during the initial scanning of the volume of interest, 35 of the 38 ground engagements were successful. We define a successful engagement as one in which the forklift inserts the tines without moving the pallet. In one of the three failures, the vehicle inserted the tines but moved the pallet slightly in the process. In tests of truck-based engagements, the manipulation was successful in all 30 tests in which the pallet was visible during the initial scanning process.

6.3 Vision-based Object Reacquisition

We performed an extensive set of experiments over several days at the Fort Lee SSA to validate the performance of our object reacquisition algorithm. The experiments are meant to assess the method's robustness to typical conditions that include variations in illumination, scene clutter, object ambiguity, and changes in context. The environment consisted of more than one hundred closely-spaced objects, including washing machines, generators, tires, engines, and trucks, among others. In most cases, objects of the same type with nearly identical appearance were placed less than a meter apart (i.e., well below the accuracy of the robot's absolute position estimate). In addition to clutter, the data sets were chosen for the presence of lighting variation that include global brightness changes, specular illumination, and shadow effects, along with viewpoint changes and motion blur. The conditions are representative of many of the challenges typical to operations in unprepared, outdoor settings.

We structured the experiments to emulate a typical operation in which the user first provides the robot with a single example of each object as the robot drives alongside them. We follow this *tour* (*training*) phase with a *reacquisition* (*testing*) phase in which the user directs the robot to retrieve one or more of the objects by name. A number of conditions can change between the time that the object is first indicated and the time it is reacquired, including the physical sensor (right-facing vs. front-facing camera), illumination, object positions within the environment, aspect angle, and scale.



(a) Relative Angle (Truck)

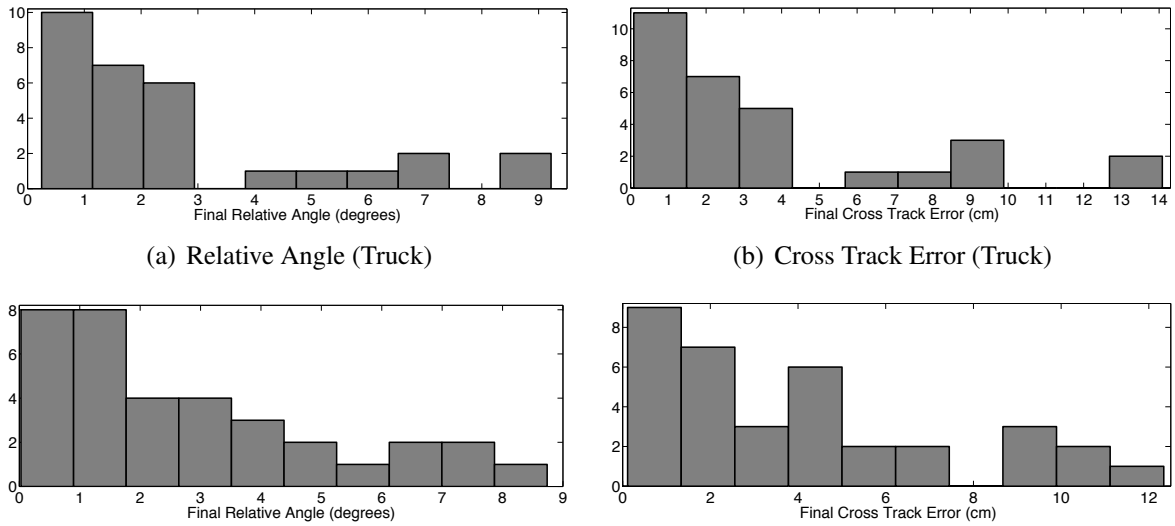
(b) Cross Track Error (Truck)

(c) Relative Angle (Ground)

(d) Cross Track Error (Ground)

Figure 21: Results of the pallet engagement tests from (a-b) a truck bed and (c-d) the ground. Each path represents the robot's trajectory during a successful pickup. Each 'x' denotes the robot's initial position for failed detections (red) and engagements (blue). Arrows indicate the robot's forward direction. All poses are shown relative to that of the pallet, centered at the origin with the front face along the x-axis. The trajectories are colored according to (a), (c) the relative angle between the pallet and the robot (in degrees) and (b), (d) the cross track error (in cm) immediately prior to insertion.

Several video clips collected at 2 Hz were paired with one another in five combinations. Each pair consists of a short tour clip acquired from the right-facing camera and a longer reacquisition clip acquired from the front-facing camera. Ground truth annotations were manually generated for each image in the reacquisition clips and were used to evaluate performance in terms of precision and recall. We consider a detection to be valid if the area of the intersection of the detection and ground truth regions exceed a fraction of the area of their union.



(a) Relative Angle (Truck)

(b) Cross Track Error (Truck)

(c) Relative Angle (Ground)

(d) Cross Track Error (Ground)

Figure 22: Histograms of the error in relative angle (left) and lateral offset (right) for pallets engaged from truck beds and the ground.

Table 2 lists the scenarios, their characteristics, and the performance achieved by our algorithm. Possible condition changes between tour and reacquisition clips include *sensor* (right vs. front camera), *lighting* (illumination and shadows), *3D pose* (scale, standoff, and aspect angle), *context* (unobserved object relocation with respect to the environment), *confusers* (objects of similar appearance nearby), and Δt (intervening hours:minutes). True and false positives are denoted as TP and FP, respectively; *truth* indicates the total number of ground truth instances; *frames* is the total number of images; and *objects* refers to the number of unique object instances that were toured in the scenario. Performance is reported in terms of aggregate precision TP/(TP+FP) and recall TP/truth.

In all five experiments, the method produces few if any false positives as indicated by the high precision rates. This demonstrates that our approach to modeling an object's appearance variation online does not result in drift, as often occurs with adaptive learners. We attribute this behavior to the geometric constraints that help to prevent occlusions and clutter from corrupting the appearance models.

While the algorithm performs well in this respect, it yields a reduced number of overall detections in the first scenario. This experiment involves significant viewpoint changes between the initial training phase and the subsequent test session. Training images were collected as the robot moved in a direction parallel to the front face of each object, and the user provided the initial training example for each object, when it was fronto-parallel to the image. As the robot proceeded forward, only views of the object's front face and one side were available for and added to its appearance model. During the testing phase of the experiment, the robot approached the objects from a range of different angles, many of which result in views of unmodeled sides of the object. In these cases, the algorithm is unable to identify a match to the learned model. This, together with saturated images of highly reflective objects results in an increased false negative rate. While utilizing homography

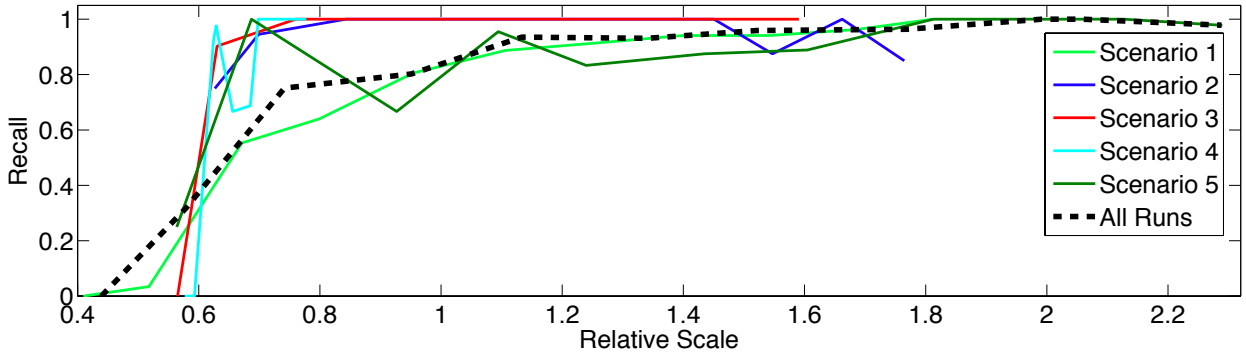
Table 2: Conditions and reacquisition statistics for the different experiment scenarios.

Scenario	1	2	3	4	5
Train	Afternoon	Evening	Morning	Morning	Noon
Test	Afternoon	Evening	Evening	Evening	Evening
Sensor	✓	✓	✓	✓	✓
Lighting		✓	✓	✓	✓
3D pose	✓	✓	✓	✓	✓
Context					✓
Confusers	✓	✓	✓		
Δt	00:05	00:05	14:00	10:00	07:00
Frames	378	167	165	260	377
Objects	6	1	1	1	1
Truth	1781	167	165	256	257
TP	964	158	154	242	243
FP	59	0	0	0	0
Precision	94.23%	100%	100%	100%	100%
Recall	54.13%	94.61%	93.33%	94.53%	94.55%

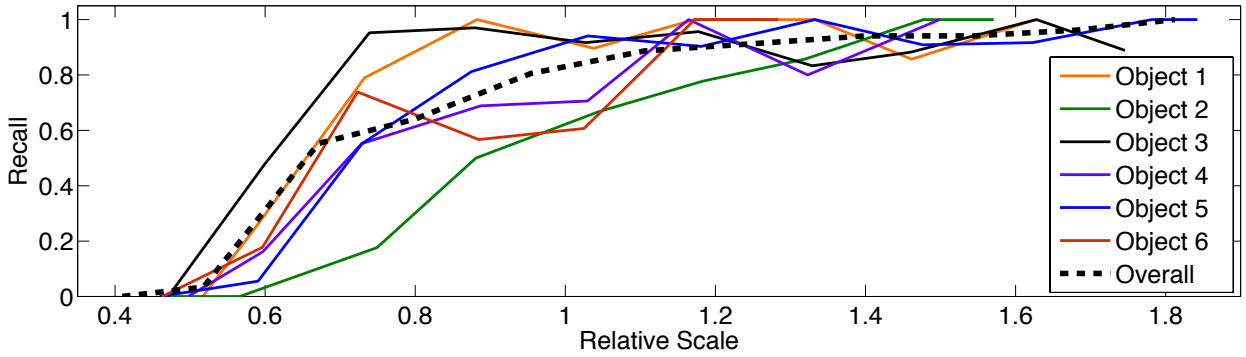
validation as part of the multiple-view matching significantly reduces false matches, it also results in false negatives due to unmodeled 3D effects such as parallax.

We evaluate the relationship between recall rate and the change in scale between an object's initial view (scale=1) and subsequent observations. Figure 23(a) plots aggregate performance of all objects for each of the five test scenarios, while Figure 23(b) shows individual performance of each object in Scenario 1. Figure 24 plots the performance of a single object from Scenario 5 in which the context has changed: the object was transported to a different location while nearby objects were moved. Finally Figure 25 reports recall rates for this object, which is visible in each of the scenarios.

For the above experiments, we manually injected a gesture for each object during each tour clip—while the robot was stationary—to initiate model learning. We employ a single set of parameters for all scenarios: for single-view matching, the SIFT feature match threshold (dot product) is 0.9 with a maximum of 600 RANSAC iterations and an outlier threshold of 10 pixels; single-view matches with confidence values below 0.1 are discarded. The reasoning module adds new views whenever a scale change of at least 1.2 is observed and the robot moves at least 0.5 m. We find that the false positive rate is insensitive to these settings for the reasoning parameters, which we chose to effectively trade off object view diversity and model complexity (data size).



(a) By Scenario



(b) By Object

Figure 23: Recall rates as a function of scale change (a) for all objects by scenario and (b) for each object in Scenario 1.

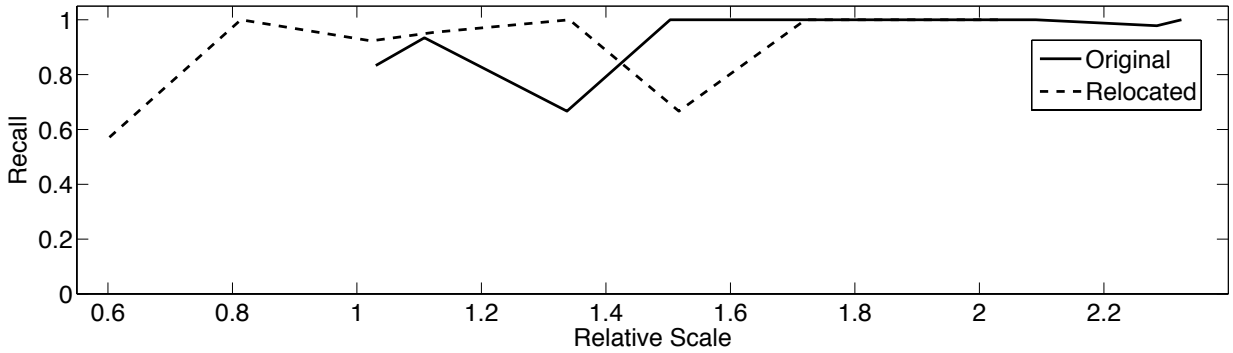


Figure 24: Recall rates as a function of scale change for an object in different positions and at different times. The pallet was on the ground during the tour and reacquired 7 hours later both on the ground and on a truck bed.

6.4 Shouted Warning Detection

We assess the behavior of the shouted warning system through a study involving five male subjects in the MIT model warehouse on a fairly windy day (6 m/s average wind speed), with wind gusts

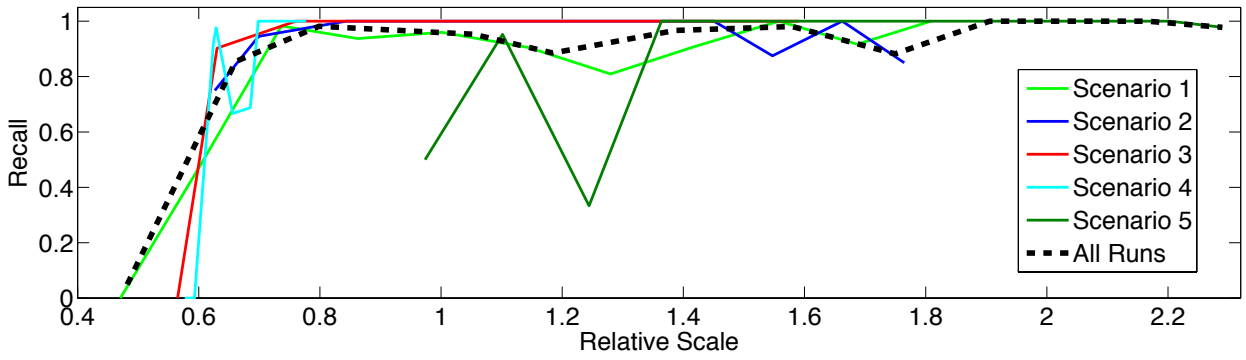


Figure 25: Recall rates as a function of scale change for a single object across all scenarios.

clearly audible in the array microphones. Subjects were instructed to shout either "Forklift stop moving" or "Forklift stop" under six different operating conditions: idling (reverberant noise), beeping, revving engine, moving forward, backing up (and beeping), and moving with another truck nearby backing up (and beeping). Each subject shouted commands under each condition (typically at increasing volume) until successful detection occurred. All subjects were ultimately successful under each condition; the worst case required four attempts from one subject during the initial idling condition. Including repetitions, a total of 36 shouted commands were made, of which 26 were detected successfully on the first try. The most difficult operating condition occurred when the engine was being revved (low SNR), resulting in five missed detections and the only two false positives. The other two missed detections occurred when the secondary truck was active. We refer the reader to our earlier work (Chuangsuwanich et al., 2010; Chuangsuwanich and Glass, 2011), which presents the results of additional shouted warning experiments.

6.5 End-to-End Operation

The robot has successfully performed a number of end-to-end deployments at numerous model and actual military SSAs. These include extensive testing over the course of several years at the MIT warehouse test site, in which the robot was commanded by both civilians as well as military personnel. The experiments include scenarios in which the operator commanded the robot from within its immediate vicinity, using both the tablet interface as well as direct speech. Additionally, the robot has successfully performed end-to-end missions directed by operators who were more than 1000 km away over a standard internet connection. This type of control is made possible by the task-level nature of the command-and-control architecture together with our system design, which favors greater autonomy and situational awareness on the part of the robot.

Among our many trials with the U.S. Army, the robot operated successfully over the course of two weeks on a packed earth facility at Fort Belvoir (Virginia) in June 2009. Under voice and gesture command of a U.S. Army Staff Sergeant, the forklift unloaded pallets from a flatbed truck in the receiving area, drove to a bulk yard location specified verbally by the supervisor, and placed the pallets on the ground. The robot, commanded by the supervisor's stylus gesture and verbally-specified destination, retrieved another indicated pallet from the ground and placed it on a flatbed

truck in the issuing area. During operation, the robot was interrupted by shouted "Stop" commands, pedestrians (mannequins) were placed in its path, and observers stood and walked nearby.

We also directed the robot to perform impossible tasks, such as engaging a pallet whose slots were physically and visually obscured by fallen cargo. In this case, the forklift paused and requested supervisor assistance. In general, such assistance can come in three forms: the supervisor can command the robot to abandon the task; a human can modify the world to make the robot's task feasible; or a human can climb into the forklift cabin and operate it through the challenging task. In this case, we manually moved the obstruction and resumed autonomous operation.

In June 2010, the robot was deployed for two weeks at an active U.S. Army SSA at Fort Lee (Virginia). Over the course of the year since the June 2009 operation, we had developed the robot's ability to perform vision-based object reacquisition and to correctly interpret and execute more extensive spoken commands, including those that reference objects in the environment. Additionally, we endowed the robot with additional annunciation mechanisms and safety behaviors, including the ability to regulate its speed based upon its perception of local hazards in its surround. During the experiments at Fort Lee, personnel commanded the vehicle to perform a number of pallet manipulation and transport tasks using a combination of the tablet interface and by directly speaking to the forklift. The vehicle safely carried out these tasks in the presence of military personnel both on foot and operating other forklifts and trucks.

7 Lessons Learned and Future Work

While military observers judged the demonstrations to be successful, the prototype capability remains crude when compared to the capabilities of skilled operators. In operational settings, the requirement that the supervisor break down complex task that require advanced levels of reasoning (e.g., "Unload the truck") into individual subtasks, and explicitly issue a command for each would likely become burdensome. A direction for current work is to enable robots to reason over higher-level actions to achieve greater autonomy. Moreover, our robot is not yet capable manipulating objects with nearly the same level of dexterity as expert human operators (e.g., lifting the edge of a pallet with one tine to rotate or reposition it, gently bouncing a load to settle it on the tines, shoving one load with another, etc.). This aptitude requires advanced capabilities to estimate and reason over load dynamics, and plan actions drawn from a set of behaviors far richer than the set that our system considers.

We learned a number of valuable lessons from testing with real military users. First, pallet indication gestures vary widely in shape and size. The resulting conical region sometimes includes extraneous objects, causing the pallet detector to fail to lock on to the correct pallet. Second, people are willing to accommodate the robot's limitations. For example, if a speech command or gesture is misunderstood, the supervisor will often cancel execution and repeat the command; if a shout is not heard, the shouter will repeat it more loudly. This behavior is consistent with the way a human worker might interact with a relatively inexperienced newcomer.

We developed an interaction mechanism that uses the microphones affixed to the vehicle to allow

people to command the robot simply by talking to it, much as they would with a manned vehicle. This capability, however, raises challenges that must be addressed if the system is to be deployed in noisy, hostile environments. While human operators are able to identify which commands are directed towards them, our system can not isolate relevant speech from audible clutter. Indeed, this is an open research problem. Additionally, it is difficult to recognize who is issuing the command with this form of interaction, thereby limiting the security of the command interface, which military personnel have expressed as a priority of any deployed system.

More generally, recognition of shouted speech in noisy environments has received little attention in the speech community, and presents a significant challenge to current speech recognition technology. From a usability perspective, it is likely that a user may not be able to remember specific "stop" commands, and that the shouter will be stressed, especially if the forklift does not respond to an initial shout. From a safety perspective, it may be appropriate for the forklift to pause if it hears anyone shout in its general vicinity. Thus, we are collecting a much larger corpus of general shouted speech, and aim to develop a capability to identify general shouted speech, as a precursor to identifying any particular command. In addition, we are also exploring methods that allow the detection module to adapt to new audio environments through feedback from users.

Support for additional interaction mechanisms would improve usability and acceptance by current personnel. Speech is only one way in which supervisors command manned forklifts. During each of our visits to active storage and distribution centers, we found that people make extensive use of hand and body gestures to convey information to the driver. People often use gestures by themselves or accompany them with spoken commands, for example, saying "Put the pallet of tires over there" and pointing at or gazing towards the referred location. In other instances, particularly when conditions are noisy, people may use only gestures to communicate with the operator. For example, ground guides frequently use hand signals to help the driver negotiate a difficult maneuver. Our system does not provide a mechanism for people to communicate with the robot through hand gestures or eye gaze. Several gesture-based input mechanisms exist (Perzanowski et al., 2001), however, and it would be possible to take advantage of the structured nature of gestures used by the military (Song et al., 2011) to incorporate this form of interaction. We are currently investigating algorithms that incorporate deictic gestures into our natural language grounding framework.

As the robot is directed to perform increasingly complex tasks, there is a greater likelihood that it will encounter situations in which it can not make progress. Our system includes a centralized process that monitors the robot's status in conjunction with status monitoring capabilities at the level of individual processes. This allows the robot to detect certain instances when it is not making progress, but requires overly detailed input from the system designer. They can identify each potential failure condition and the associated symptoms, but that does not scale with task complexity. Alternatively, the designer may establish abstract symptoms that suggest failure (e.g., the time duration of the current task), but that makes it difficult to identify the specific cause and, in turn, to ask for help. We adopted both approaches. A fielded system requires a capacity to detect when the robot is "stuck" that generalizes to the large space of possible failure conditions and that can discern the different causes, with minimal domain knowledge required of the designer. Further, the system should notify the user in a comprehensible manner and request assistance, by asking for additional information to resolve ambiguity (Tellex et al., 2012) or for the user to take over manual

control. It would be desirable for the robot to use these instances as positive examples and learn from the user-provided demonstrations (Argall et al., 2009) to improve its proficiency at the task.

Every one of our tests at MIT and military bases was performed with a dedicated safety operator monitoring the robot. Without exception, the robot operated safely but in the event that anything went wrong, we could immediately disable the robot using a remote kill switch. In order to deploy the system commercially or with the military, which we can not require to have a safety officer, other mechanisms are necessary to guarantee that the robot's behavior is safe. Most critical, the robot must be able to recognize each instance when a person, either on foot or driving another vehicle, enters or is likely to enter its path, and behave accordingly. Similarly, the robot mustn't engage or place cargo in close proximity to people. The slow speeds at which the vehicle operates improves reaction time. The greatest challenge is to develop algorithms for person detection suitable to unstructured environments that can guarantee near-zero false negative rates with false positive rates sufficiently low to be usable in practice. To our knowledge, algorithms do not exist that offer performance comparable to that of human operators.

Our system fuses data from several sensors mounted to the robot, which requires that each be accurately calibrated. While there are well-defined procedures to determine the intrinsic and extrinsic calibration parameters for LIDARs and cameras, they can be time consuming and require external infrastructure. It is not uncommon for these parameters to change over the course of operations, for example, as a result of sensors being replaced due to failure or being removed and remounted. Each time this happens, the sensors must be recalibrated, which can be a challenge when the vehicle is deployed. One way to simplify the extrinsic calibration would be to manufacture vehicles with fixed sensor mount points with known pose relative to a body-fixed reference frame. Alternatively, the robotics community would benefit from calibration procedures that can be performed opportunistically in situ, such as by exploiting the terrain. An on-the-fly capability would not only simplify calibration, but it could also be used to automatically detect instances of inaccurate calibration estimates. There has been some work in this direction, including the use of ego-motion for calibration (Brookshire and Teller, 2012), but it remains an open problem.

Over the past several years, the automotive industry has been moving towards manufacturing cars that ship with drive-by-wire actuation. To our knowledge, the same is not true of most commercial ground vehicles, including lift trucks. We chose the Toyota platform since it offered the easiest path to drive-by-wire actuation, but we still spent significant effort modifying our baseline lift truck to control its degrees of freedom. Particularly difficult were the steering, brake, and parking brake inputs, which required that we mount and control three motors and interface a clutch with the steering column. It would greatly simplify development and deployment if vehicles were manufactured to be drive-by-wire with a common interface such as CAN bus for control. Manufacturers are not incentivized by research customers to make these changes, but the transition can be accelerated by customers with greater purchasing power like the U.S. military.

The storage and distribution centers operated by the military or disaster relief groups are typically located in harsh environments. The sensors onboard the vehicle must be resistant to intrusions from inclement weather, mud, sand storms, and dust. Most of the externally-mounted hardware on our platform has an IP64 rating or better, however greater levels of protection are necessary

for the environments that we are targeting. Meanwhile, we encountered several instances when one or more of our sensors (e.g., a laser range finder) became occluded by dust and mud. The system needs a means of detecting these instances and, ideally, mechanisms to clear the sensor's field-of-view, much like windshield wipers. Perhaps most challenging is to develop sensors and perception algorithms that can function during extreme events like sand storms, which would blind the cameras and LIDARs onboard our robot.

Throughout the duration of the project, representatives from the military and industry stressed the importance of having easy-to-use tools to diagnose hardware and software failures. Our group has developed a few different introspective tools as part of LCM (Huang et al., 2010) and Libbot (Huang et al., 2014), which we used extensively during development and testing. These tools, however, are intended primarily as diagnostic tools for developers and are deliberately verbose in the amount of information that they expose to the user. If the system were to be put in the hands of users without the benefit of having developers nearby, we would need to provide simple diagnostic tools, analogous to the scan tools used in the automotive industry.

8 Conclusion

This paper described the design and implementation of an autonomous forklift that manipulates and transports cargo under the direction of and alongside people within minimally-prepared outdoor environments. Our approach involved early consultation with members of the U.S. military to develop a solution that would be both usable and culturally acceptable by the intended users. Our contributions include: novel approaches to shared situational awareness between humans and robots; efficient command and control methods; and mechanisms for improved predictability and safety of robot behavior. Our solution includes a one-shot visual memory algorithm that opportunistically learns the appearance of objects in the environment from single user-provided examples. This method allows the robot to reliably detect and locate objects by name. Our multimodal interface enables people to efficiently issue task-level directives to the robot through a combination of simple stylus gestures and spoken commands. The system includes an anytime motion planner, annunciation and visualization mechanisms, and pedestrian and shout detectors.

We described the principal components of our system and their integration into a prototype mobile manipulator. We evaluated the effectiveness of these modules and described tests of the overall system through experiments at model and operating military warehouses. We discussed the lessons learned from these experiences and our interaction with the U.S. military, and offered our conclusions on where future work is needed.

References

Alami, R., Clodic, A., Montreuil, V., Sisbot, E. A., and Chatila, R. (2006). Toward human-aware robot task planning. In *Proceedings of the AAAI Spring Symposium*, pages 39–46, Stanford, CA.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from

demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

Arras, K. O., Mozos, O. M., and Burgard, W. (2007). Using boosted features for the detection of people in 2D range data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3402–3407, Rome, Italy.

Babenko, B., Yang, M.-H., and Belongie, S. (2009). Visual tracking with online multiple instance learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 983–990, Miami, FL.

Bostelman, R., Hong, T., and Chang, T. (2006). Visualization of pallets. In *Proceedings of SPIE, Intelligent Robots and Computer Vision XXIV: Algorithns, Techniques, and Active Vision*, Boston, MA.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brookshire, J. and Teller, S. (2012). Extrinsic calibration from per-sensor egomotion. In *Proceedings of Robotics: Science and Systems (RSS)*, Sydney, Australia.

Chuangsuwanich, E., Cyphers, S., Glass, J., and Teller, S. (2010). Spoken command of large mobile robots in outdoor environments. In *Proceedings of the IEEE Spoken Language Technologies Workshop (STL)*, pages 306–311, Berkeley, CA.

Chuangsuwanich, E. and Glass, J. (2011). Robust voice activity detector for real world applications using harmonicity and modulation. In *Proceedings of Inerspeech*, pages 2645–2648, Florence, Italy.

Collet, A., Berenson, D., Srinivasa, S. S., and Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 48–55, Kobe, Japan.

Collins, R. T., Liu, Y., and Leordeanu, M. (2005). Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577.

Correa, A., Walter, M. R., Fletcher, L., Glass, J., Teller, S., and Davis, R. (2010). Multimodal interaction with an autonomous forklift. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 243–250, Osaka, Japan.

Cucchiara, R., Piccardi, M., and Prati, A. (2000). Focus-based feature extraction for pallets recognition. In *Proceedings of the British Machine Vision Conference*, Bristol, U.K.

Cui, J., Zha, H., Zhao, H., and Shibasaki, R. (2007). Laser-based detection and tracking of multiple people in crowds. *Computer Vision and Image Understanding*, 106(2-3):300–312.

Davis, R. (2002). Sketch understanding in design: Overview of work at the MIT AI Lab. In *Proceedings of the AAAI Spring Symposium on Sketch Understanding*, pages 24–31, Stanford, CA.

Dragan, A. and Srinivasa, S. (2013). Generating legible motion. In *Proceedings of Robotics: Science and Systems (RSS)*, Berlin, Germany.

Dubins, L. E. (1957). On the curves of minimal length on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516.

Durrant-Whyte, H., Pagac, D., Rogers, B., Stevens, M., and Nelmes, G. (2007). An autonomus straddle carrier for moving of shipping containers. *IEEE Robotics & Automation Magazine*, 14(3):14–23.

Dzifcak, J., Scheutz, M., Baral, C., and Schermerhorn, P. (2009). What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4163–4168, Kobe, Japan.

Ferland, F., Pomerleau, F., Le Dinh, C., and Michaud, F. (2009). Egocentric and exocentric teleoperation interface using real-time, 3D video projection. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 37–44.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fong, T., Thorpe, C., and Glass, B. (2003). PdaDriver: A handheld system for remote driving. In *Proceedings of the IEEE International Conference on Advanced Robotics*, Coimbra, Portugal.

Glass, J. R. (2003). A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17(2–3):137–152.

Gordon, I. and Lowe, D. G. (2006). What and where: 3D object recognition with accurate pose. In *Toward Category-Level Object Recognition*, pages 67–82. Springer-Verlag.

Grabner, H., Leistner, C., and Bischof, H. (2008). Semi-supervised on-line boosting for robust tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–247, Marseille, France.

Hähnel, D., Schulz, D., and Burgard, W. (2003). Mobile robot mapping in populated environments. *Advanced Robotics*, 17(7):579–597.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

Hetherington, I. L. (2007). PocketSUMMIT: Small-footprint continuous speech recognition. In *Proceedings of Interspeech*, pages 1465–1468, Antwerp, Belgium.

Hilton, J. (2013). Robots improving materials handling efficiencies while reducing costs. *Automotive Industries*, 192(2):48–49.

Hoffmann, G. M., Tomlin, C. J., Montemerlo, M., and Thrun, S. (2007). Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and testing. In *Proceedings of the American Control Conference (ACC)*, pages 2296–2301, New York, NY.

Hoiem, D., Rother, C., and Winn, J. (2007). 3D LayoutCRF for multi-view object class recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN.

Holzapfel, H., Nickel, K., and Stiefelhagen, R. (2004). Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3D pointing gestures. In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, pages 175–182, State College, PA.

Huang, A. S., Bachrach, A., and Walter, M. R. (2014). Libbot robotics library.

Huang, A. S., Olson, E., and Moore, D. (2010). LCM: Lightweight communications and marshalling. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4057–4062, Taipei, Taiwan.

Jeon, J. h., Karaman, S., and Frazzoli, E. (2011). Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 3276–3282, Orlando, FL.

Kalal, Z., Matas, J., and Mikolajczyk, K. (2009). Online learning of robust object detectors during unstable tracking. In *On-line Learning for Computer Vision Workshop*, pages 1417–1424, Kobe, Japan.

Kalal, Z., Matas, J., and Mikolajczyk, K. (2010). P-N learning: Bootstrapping binary classifiers by structural constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–56, San Francisco, CA.

Karaman, S. and Frazzoli, E. (2010a). Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems (RSS)*, Zaragoza, Spain.

Karaman, S. and Frazzoli, E. (2010b). Optimal kinodynamic motion planning using incremental sampling-based methods. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 7681–7687, Atlanta, GA.

Karaman, S. and Frazzoli, E. (2011). Sample-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the RRT*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1478–1483, Shanghai, China.

Kelly, A., Nagy, B., Stager, D., and Unnikrishnan, R. (2007). An infrastructure-free automated guided vehicle based on computer vision. *IEEE Robotics & Automation Magazine*, 14(3):24–34.

Keskinpala, H. K. and Adams, J. A. (2004). Objective data analysis for PDA-based human-robot interaction. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2809–2814, The Hague, The Netherlands.

Keskinpala, H. K., Adams, J. A., and Kawamura, K. (2003). PDA-based human-robotic interface. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics (SMC)*, pages 3931–3936, Washington, DC.

Klein, G., Woods, D. D., Bradshaw, J. M., Hoffman, R. R., and Feltovich, P. J. (2004). Ten challenges for making automation a "team player" in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010). Toward understanding natural language directions. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266, Osaka, Japan.

Lecking, D., Wulf, O., and Wagner, B. (2006). Variable pallet pick-up for automatic guided vehicles in industrial environments. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (EFTA)*, pages 1169–1174, Toulouse, France.

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774.

Liebelt, J., Schmid, C., and Schertler, K. (2008). Viewpoint-independent object class detection using 3D feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK.

Lim, J., Ross, D., Lin, R.-S., and Yang, M.-H. (2004). Incremental learning for visual tracking. In *Advances in Neural Information Processing Systems (NIPS)*, pages 793–800, Vancouver, B.C., Canada.

Lowe, D. G. (2001). Local feature view clustering for 3D object recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 682–688, Kauai, HI.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110.

MacMahon, M., Stankiewicz, B., and Kuipers, B. (2006). Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1475–1482, Boston, MA.

Marble, J. D. and Bekris, K. E. (2011). Asymptotically near-optimal is good enough for motion planning. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, Flagstaff, AZ.

Matsumaru, T., Iwase, K., Akiyama, K., Kusada, T., and Ito, T. (2005). Mobile robot with eyeball expression as the preliminary-announcement and display of the robot's following motion. *Autonomous Robots*, 18(2):231–246.

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., and Fox, D. (2012). A joint model of language and perception for grounded attribute learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, Edinburgh, Scotland.

Matuszek, C., Fox, D., and Koscher, K. (2010). Following directions using statistical machine translation. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258, Osaka, Japan.

Moore, D. A., Huang, A. S., Walter, M., Olson, E., Fletcher, L., Leonard, J., and Teller, S. (2009). Simultaneous local and global state estimation for robotic navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3794–3799, Kobe, Japan.

Mutlu, B., Yamaoka, F., Kanda, T., Ishiguro, H., and Hagita, N. (2009). Nonverbal leakage in robots: communication of intentions through seemingly unintentional behavior. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 69–76, San Diego, CA.

Nebot, E. M. (2005). Surface mining: Main research issues for autonomous operations. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, pages 268–280, San Francisco, CA.

Perzanowski, D., Schultz, A. C., Adams, W., Marsh, E., and Bugajska, M. (2001). Building a multimodal human-robot interface. *IEEE Intelligent Systems*, 16(1):16–21.

Pradalier, C., Tews, A., and Roberts, J. (2010). Vision-based operations of a large industrial vehicle: Autonomous hot metal carrier. *Journal of Field Robotics*, 25(4-5):243–267.

Savarese, S. and Fei-Fei, L. (2007). 3D generic object categorization, localization and pose estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Seelinger, M. and Yoder, J. D. (2006). Automatic visual guidance of a forklift engaging a pallet. *Robotics and Autonomous Systems*, 54(12):1026–1038.

Skubic, M., Anderson, D., Blisard, S., Perzanowski, D., and Schultz, A. (2007). Using a hand-drawn sketch to control a team of robots. *Autonomous Robots*, 22(4):399–410.

Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., and Brock, D. (2004). Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(2):154–167.

Song, Y., Demirdjian, D., and Davis, R. (2011). Tracking body and hands for gesture recognition: NATOPS aircraft handling sygnals database. In *Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition Workshops*, pages 500–506, Santa Barbara, CA.

Takayama, L., Dooley, D., and Ju, W. (2011). Expressing thought: Improving robot readability with animation principles. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 69–76, Lausanne, Switzerland.

Teller, S., Walter, M. R., Antone, M., Correa, A., Davis, R., Fletcher, L., Frazzoli, E., Glass, J., How, J. P., Huang, A. S., Jeon, J. h., Karaman, S., Luders, B., Roy, N., and Sainath, T. (2010). A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pages 526–533, Anchorage, AK.

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI)*, pages 1507–1514, San Francisco, CA.

Tellex, S., Thaker, P., Deits, R., Kollar, T., and Roy, N. (2012). Toward information theoretic human-robot dialog. In *Proceedings of Robotics: Science and Systems (RSS)*, Sydney, Australia.

Tews, A., Pradalier, C., and Roberts, J. (2007). Autonomous hot metal carrier. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1176–1182, Rome, Italy.

United States Department of Labor Occupational Safety & Health Administration (1969). Powered industrial trucks – occupational safety and health standards – 1910.178.

Walter, M. R., Friedman, Y., Antone, M., and Teller, S. (2012). One-shot visual appearance learning for mobile manipulation. *International Journal of Robotics Research*, 31(4):554–567.

Walter, M. R., Karaman, S., Frazzoli, E., and Teller, S. (2010). Closed-loop pallet engagement in unstructured environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5119–5126, Taipei, Taiwan.

Wurman, P. R., D'Andrea, R., and Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–19.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, 38(4).

Zhang, Q. and Pless, R. (2004). Extrinsic calibration of a camera and a laser range finder. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2301–2306, Sendai, Japan.