# Closed-loop Pallet Engagement in an Unstructured Environment

Sertac Karaman*, Matthew R. Walter†, Emilio Frazzoli*, Seth Teller†

*Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA, USA
{sertac, frazzoli}@mit.edu

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

***Abstract*— This paper addresses the problem of autonomous manipulation of *a priori* unknown palletized cargo with a robotic lift truck. More specifically, we describe coupled perception and control algorithms that enable the vehicle to engage and drop off loaded pallets relative to locations on the ground or arbitrary truck beds. With little prior knowledge of the objects with which the vehicle is to interact, we present an estimation framework that utilizes a series of classifiers to infer the objects' structure and pose from individual LIDAR scans. The different classifiers share a low-level shape estimation algorithm that uses a linear program to robustly segment input data to generate a set of weak candidate features. We present and analyze the performance of the segmentation and subsequently describe its role in our estimation algorithm. We then evaluate the performance of the motion controller that, given an estimate for a pallet's pose, we employ to safely engage a pallet. We conclude with a validation of our algorithms for a set of real world pallet and truck interactions.**

## I. Introduction

We have developed a robotic forklift for autonomous materials handling in the outdoor, semi-structured environments typical of disaster relief and military storage warehouses [1]. The system performs typical warehouse tasks under the high-level direction of a human supervisor, notably picking up, transporting, and placing palletized cargo between truck beds and ground locations in the environment. Integral to the system is the robot's ability to accurately localize and safely manipulate unknown pallets despite their variable geometry, the uneven terrain, and the unknown truck geometry.

Successfully picking up a pallet from a truck with a 2700 kg forklift, given perfect information regarding the poses of the robot, pallet, and truck, is relatively easy. In real settings, the challenges lie in accurately controlling the nonholonomic lift truck so as to safely insert the tines within the pallet's slots. With little *a priori* information, however, the system must also detect the pallet and the truck bed, and subsequently maintain an accurate estimate for their structure and pose while approaching and engaging the pallet. These tasks are made difficult by variability in pallet and truck geometry together with the limited sensing available. For example, while certain features of cargo pallets are present across most pallets (i.e., roughly rectilinear, generally flat, usually two insertion points designed for forklift tines), the dense geometry of pallets is highly variable. The forklift must use onboard sensing to recover the pallet geometry in order to correctly insert the lifting tines; unlike many small-object manipulation strategies, it is not possible to use manipulator compliance or feedback control strategies to ease insertion. Even small forklifts designed for indoor warehouses can exert tons of force; the tines are extremely rigid and cannot be instrumented with the tactile sensing necessary for feedback control strategies for manipulation. As a result, attempting to insert tines incorrectly can damage or destroy the pallet (or its load) before the failure can be detected and corrected.

Fig. 1. The prototype forklift that is the host platform for the mobile manipulation algorithm presented in the paper. The vehicle autonomously detects and engages unknown pallets, picking them up from, and placing them onto the ground or the bed of a truck. The rendering on the right depicts the corresponding output of the pallet and truck estimation algorithms.

In addition, a pallet's appearance is also variable. The physical pallet structure is quite sparse, roughly 1 m square with a height of 15 cm and inserts that are each 30 cm wide. A view of a candidate pallet location is dominated largely by LIDAR returns from the pallet's load as well as the surface on which the pallet lies. Similarly, a truck's undercarriage comprises most of the view of a vertically-scanning LIDAR, with limited returns arising from the vertical and horizontal faces of the truck bed. Further complicating the problem of accurately detecting and estimating the pallet and truck poses is the fact that, while they are themselves rigid, the forklift's tines and carriage, to which the LIDARs are mounted, are not rigidly attached to the vehicle, which limits the accuracy of extrinsic calibration.

This paper presents a coupled perception and control strategy that addresses these challenges, enabling the forklift to manipulate unknown pallets within semi-structured, outdoor environments. We first introduce the overall robotic platform, briefly describing the aspects that are pertinent to our mobile manipulation work. We then describe a general strategy for pattern detection that identifies candidate linear structure within noisy 2D LIDAR scans. We next describe algorithms for pallet and truck estimation that utilize this pattern recognition tool to detect returns from the pallet structure and truck bed using weak to strong classifiers. The algorithms utilize positive detections as inputs to a set of filters that maintain estimates for the pallet and truck poses throughout engagement. We then describe the control strategy that we use to servo the poses of the vehicle and tines. Finally, we present the results of a series of validation tests that demonstrate the accuracy and limitations of our mobile manipulation strategy.

## II. Related Work

There has been considerable work in developing mobile manipulators to accomplish useful tasks in populated environments. This work has largely focused on the problems of planning and control [2], [3], which are not inconsiderable for a robot with many degrees of freedom and many actuators capable of exerting considerable force and torque. These approaches have generally taken one of two approaches: either assume a high-fidelity kinodynamic model and apply sophisticated search to solve for a feasible control plan [4]–[6], or use reactive policies with substantial sensing and feedback control (either visual [7] or tactile [8], [9]) to avoid the requirements of a model.

Meanwhile, there has been extensive work addressing the problem of object segmentation, classification, and estimation based upon range data. In particular, early work by Hebert *et al.* [10] describes algorithms for object detection and recognition with an outdoor robot using laser scan data. Hoffman and Jain [11] present a method, based on range data, to detect and classify the faces comprising 3D objects. Similarly, Newman *et al.* [12] propose a model-driven technique that leverages prior knowledge of object surface geometry to jointly classify and estimate surface structure. These techniques require range images of the scene, which, in the case of our platform, are subject to systematic error due to the pliancy of the forklift structure to which the LIDARs are mounted. Researchers have extended the robustness of range image segmentation [13] and object model parameter estimation [14], [15] using randomized sampling to accommodate range images with many outliers.

The specific problem of developing an autonomous lift truck that is able to pick up and transport loaded pallets is not new. The same is true of pallet detection and localization methods, which have been studied in the perception community due to pallets' sparse structure. Most of this work, however, differs significantly from our own, in that it assumes a clean, highly-structured environment, does not generalize across varying pallet geometry [16]–[18], and does not consider the challenging problem of placing pallets onto and picking pallets off of unknown truck beds.

## III. System Overview

Our platform is a 2700 kg Toyota forklift with drive-by-wire modifications enabling computer-based control of the vehicle and mast (i.e., tine height and forward/backward tilt) actuation. The platform is equipped with laser range finders for object detection as well as a forward-facing camera that provides images to a remote user's command interface. We estimate the vehicle's pose via dead-reckoning based upon wheel encoder velocity measurements together with orientation measurements from an integrated GPS/IMU.

Fig. 2. Forklift being commanded via the tablet PC to pick up a pallet that is stationed on a truck bed.

Pallet detection relies upon a single Hokuyo UTM laser range finder with a 30 m range and a 140 degree field-of-view. The unit is mounted at the elbow of one of the forklift's tines and scans in a horizontal plane situated slightly above the tine's top surface. Meanwhile, the truck bed estimation algorithms that follow utilize a pair of UTM laser range finders (30 m range, 270 degree FOV) mounted to the left and right sides of the carriage assembly with a vertical scan plane. All three LIDARs move in tilt and height with the carriage.

The forklift operates autonomously based upon high-level directives from a user who commands the system via a hand-held tablet computer [1], [19]. In the case of pallet engagement tasks, the user can direct the platform to pick up a pallet from the ground or a truck bed, or to place a pallet at a specified, unoccupied location on the ground or truck. The user indicates the desired pallet to engage by circling it within the image from the vehicle's forward-facing camera, which is displayed on the tablet (Figure 2). Similarly, the user identifies a desired pallet placement location by circling the region in the camera image.

In the subsequent sections, we explain how the robot autonomously manipulate pallets given directives of this form.

## IV. Fast Closest Edge Detection from Laser Range Finder Data

In this section, a novel efficient algorithm that identifies the closest edge in LIDAR data is proposed. Two closest edge detection problems are studied. In the relatively simple first case, the orientation of the edge is assumed to be known and the distance of the edge to the sensor is estimated. In the second variant, both the orientation and the distance of the edge are identified. Inspried by similar problems in learning with kernel methods [20], the first variant of the problem is formulated as a linear program, the dual of which is shown to be solvable in $O(n \min\{\nu, \log n\})$ time, where $n$ is the number of points and $\nu$ is a problem-specific parameter. Note

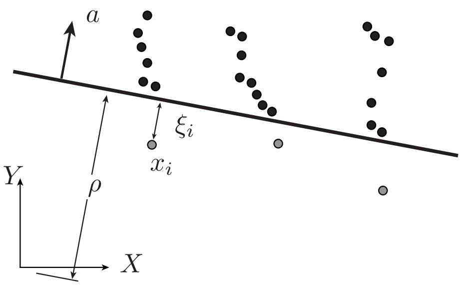



Fig. 3. A graphical representation of the closest edge detection problem for 2D laser returns from a pallet face. The three grey points are outliers with respect to the line $(a, \rho)$.

that solving the original linear program with, for instance, the interior point algorithm requires $O(n^{3.5})$ time in the worst case [21]; hence, exploiting the structure of the dual program results in significant computational savings, facilitating real-time implementation for robotics applications. For the second variant of the problem, a heuristic algorithm, which uses the algorithm for the first variant a constant number of times, is provided. Both algorithms are used as a basis to detect pallets and trucks in Sections V and VI, respectively.

### A. Closest Edge Detection with Known Orientation

Consider the first variant of the closest edge detection problem. To define the problem more formally, let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\} = \{x_i\}_{i \in \mathcal{I}}$, where $\mathcal{I} = \{1, 2, \ldots, n\}$, be the set of points in the two dimensional Euclidean space $\mathbb{R}^2$, representing the data sampled from a planar laser range finder. Figure 3 presents a simple example with laser returns that are representative of those from a pallet face. Without loss of generality, let the sensor lie in the origin of this Euclidean space and be oriented such that its normal vector is $[1,\ 0]^\top$. Let $a \in \mathbb{R}^2$ denote a normalized vector, i.e., $\| a \| = 1$. Informally, the problem is to find the distance $\rho$ of the line to the origin such that all data points in $\mathcal{X}$, except a few outliers, are separated from the origin by this line. 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, i.e., $\langle a, x \rangle$ denotes the distance of $x_i$ to the origin when projected along the vector $a$. Let $\xi_i$ denote the distance of point $x_i$ to the separating line if the distance from the origin to $x_i$ (projected along $a$) is less than $\rho$; otherwise let $\xi_i$ be zero. That is $\xi_i = \max(\rho - \langle a, x_i \rangle, 0)$ (see Figure 3).

Given a line described by a normal $a$ and distance $\rho$, a point $x_i$ with $\xi_i > 0$ is called an *outlier* with respect to the line $(a, \rho)$. We formulate the *closest edge detection problem* as maximization of the following function: $\rho - C \sum_{i \in \mathcal{I}} \xi_i$, where $C$ is a problem dependent constant parameter, that represents the trade-off between two objectives: maximizing the distance $\rho$ of the separating line to the origin and minimizing the total distance $\sum_{i \in \mathcal{I}} \xi_i$ of the outliers to line $(a, \rho)$. Notice that $C = 0$ will render $\rho = \infty$, in which case all data points will be outliers. $C \to \infty$, on the other hand, will allow no outliers in a feasible solution.

To further motivate, first let us consider the case with no outliers ($C \to \infty$) and the relatively easy problem of finding the distance $\rho$ of the line with normal $a$ to the origin such that $\rho$ is maximum and the line separates all points in $\mathcal{X}$ from the origin. Notice that a naïve algorithm that computes the distance of $x_i$ from the origin for all $i \in \mathcal{I}$ and returns the minimum distance solves this problem. Notice also that this algorithm runs in time $O(n)$. Indeed, it can be shown that any deterministic algorithm that solves this problem has to run in time $\Omega(n)$. However, due to the noise embedded in the laser range finder data, especially for LIDAR returns arising from the corners, this solution may provide noisy information. Precisely for this reason, the aforementioned formulation of the closest edge detection problem includes an extra term in the objective function so as to filter out such noise. The rest of this section details an algorithm that solves the closest edge detection problem while incurring small extra computational cost.

The closest edge detection problem can be formulated as a mathematical program as follows:

$$\begin{aligned} \text{maximize} \quad & \rho - \tfrac{1}{\nu} \textstyle\sum_{i \in \mathcal{I}} \xi_i, & & (1) \\ \text{subject to} \quad & d_i \geq \rho - \xi_i, \quad \forall i \in \mathcal{I}, & & (2) \\ & \xi_i \geq 0, \quad \forall i \in \mathcal{I}, & & (3) \end{aligned}$$

where $\rho \in \mathbb{R}$ and $\xi_i \in \mathbb{R}$ are the decision variables, and $\nu \in \mathbb{R}$ is a parameter such that $\nu = 1/C$. The parameter $d_i$ is the distance of point $x_i$ to the origin when projected along $a$, i.e., $d_i = \langle a, x_i \rangle$.

For computational purposes, it is useful to consider the dual of the linear program (1-3):

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i \in \mathcal{I}} d_i \lambda_i, & & (4) \\ \text{subject to} \quad & \textstyle\sum_{i \in \mathcal{I}} \lambda_i = 1, \quad \forall i \in \mathcal{I}, & & (5) \\ & 0 \leq \lambda_i \leq \tfrac{1}{\nu}, \quad \forall i \in \mathcal{I}, & & (6) \end{aligned}$$

where $\lambda_i$ are called the dual variables. Let $(\rho^*, \xi_1^*, \ldots, \xi_n^*)$ be an optimal solution to the linear program (1-3) and $(\lambda_1^*, \ldots, \lambda_n^*)$ be the optimal solution of the dual linear program (4-6). The optimal primal solution can be recovered from the dual solution as $\rho^* = \sum_{i \in \mathcal{I}} \lambda_i^* d_i$.

The dual linear program is particularly interesting for computational purposes. Strictly speaking,

**Proposition IV.1** *Algorithm 1 runs in $O(n \min\{\log n, \nu\})$ time and solves the dual linear program (4-6).*

Algorithm 1, `DUALSOLVE`, takes the parameter $\nu$, the normal vector $a$, and the set $\mathcal{X}$ as an input and returns an indexed set $\{\lambda_i\}_{i \in \mathcal{I}}$ of values for the dual variables. `DUALSOLVE` employs two primitive functions. `SORT` takes an indexed set $\{y_i\}_{i \in \mathcal{I}}$ as an input, where $y_i \in \mathbb{R}$, and returns a sorted sequence of indices $\mathcal{J}$ such that $y_{\mathcal{J}(j)} \leq y_{\mathcal{J}(j+1)}$ for all $j \in \{1, 2 \ldots, |\mathcal{I}|\}$. `MIN`, meanwhile, returns the index $j$ of the minimum element in a given index set, i.e., $y_j \leq y_{j'}$ for all $j' \in \mathcal{J}$.

Firstly, notice that the elementary operations in `DUALSOLVE` require only additions, multiplications, and evaluations of cross products, none of which require the computation of any trigonometric function. Apart from its theoretical computational guarantees ensured by Proposition IV.1, this particular property of Algorithm 1 makes it fast in practice as well. Secondly, notice also that with Algorithm 1 one can solve the mathematical program (1-3). Let us denote this procedure with $\texttt{DISTFIND}(\nu, a, \mathcal{X})$ (see Algorithm 2). Clearly, DISTFIND also runs in time $O(n \min\{\log n, \nu\})$.

**Algorithm 1**: $\texttt{DUALSOLVE}\,(\nu, a, \mathcal{X})$

```
for all i ∈ I do
    λ_i := 0;
for all i ∈ I do
    d_i :=< a, x_i >;
D := {d_i}_{i∈I};
if log|D| < ν then
    J := SORT(D);
    for j := 1 to ⌊ν⌋ do
        λ_{J(j)} := 1/ν;
    λ_{J(⌊ν⌋+1)} := 1 − ⌊ν⌋/ν;
else
    for i := 1 to ⌊ν⌋ do
        j := MIN(D);
        λ_j := 1/ν;
        D := D \ {d_j};
    j := MIN(D);
    λ_j := 1 − ⌊ν⌋/ν;
return {λ_i}_{i∈I}
```

**Algorithm 2**: $\texttt{DISTFIND}(\nu, a, \mathcal{X})$

```
for all i ∈ I do
    d_i :=< a, x_i >;
{λ_i}_{i∈I} := DUALSOLVE(ν, a, X);
ρ := Σ_{i∈I} λ_i d_i
```

The next sections present pallet and truck detection algorithms, which employ the DISTFIND algorithm heavily. The value $\nu$ influences the effectiveness of the detection algorithms. Although the choice of $\nu$ is generally problem-dependent, we present a couple of its interesting properties before moving on with detection algorithms.

**Proposition IV.2** $\min_{i \in \mathcal{I}} d_i \leq \rho^*$.

This proposition merely states that the distance returned by DISTFIND is never less than the distance of any of the points in $\mathcal{X}$ to the origin. That is, the line that separates the origin from the data points either passes through at least one of the data points, or there exists at least one data point that is an outlier with respect to the line. The following proposition indicates an important relation between the number of outliers and the parameter $\nu$.

**Proposition IV.3** *The parameter $\nu$ is an upper bound on the number of outliers with respect to the the line $(a, \rho^*)$.*

The proofs of these propositions are technical and are omitted for lack of space.

### *B. Closest Edge Detection with Unknown Orientation*

If the orientation is not known, then we invoke DUALSOLVE a constant number of times for a set $\{a_i\}_{i \in \{1,2,\ldots,N\}}$ of normal vectors, each oriented with angle $\theta_i$ relative to the X-axis, where $\theta_i$ are uniformly placed on the interval between $\theta_1 = \theta_{\min}$ and $\theta_2 = \theta_{\max}$ (see Algorithm 3). After each invocation to DUALSOLVE, a weighted average $z_i$ of the data points is computed where the dual variables returned from DUALSOLVE are used 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 3**: $\texttt{EDGEFIND}(\nu, \mathcal{X}, \theta_{\min}, \theta_{\max}, N)$

```
for j := 1 to N do
    θ := θ_min + (θ_max − θ_min) j/N;
    a := (cos(θ), sin(θ));
    {λ_i}_{i∈I} := DUALSOLVE(ν, a, X);
    z_j := Σ_{i∈I} λ_i x_i;
(z', a', w') := LINEFIT({z_j}_{j∈{1,2,...,N}});
return (z', a', w')
```

### *C. The Hierarchical Classification Framework*

Pallet and truck perception algorithms that we introduce in the next two sections run DISTFIND or EDGEFIND over sets $\{\mathcal{X}_k\}_{k \in \mathcal{K}}$ of data points to extract a set $\{f_k\}_{k \in \mathcal{K}}$ of *features* from the data. In most cases, these features correspond to real-world structure, such as the existence of slots in an edge returned by EDGEFIND, or the height of the truck bed detected using DISTFIND.

The data sets $\mathcal{X}_k$ can be LIDAR returns from different sensors, or returns from the same sensor but acquired at different time intervals. In some cases, $X_k$ are acquired from a single scan of the same sensor, but $\mathcal{X}_{k+1}$ is determined from the features $f_1, f_2, \ldots, f_k$ of the data sets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$. Yet, no matter how the data sets $\{\mathcal{X}_k\}_{k \in \mathcal{K}}$ are selected, the set $\{f_k\}_{k \in \mathcal{K}}$ of features are generated using intuitive algorithms that employ either DISTFIND or EDGEFIND. These features are then compared with a nominal set $\{\bar{f}_k\}_{k \in \mathcal{K}}$ of features, for instance by computing the distance $\|\{f_k\}_{k \in \mathcal{K}} - \{\bar{f}_k\}_{k \in \mathcal{K}}\|$ according to some norm; if the distance is within acceptable limits, the set $\{\mathcal{X}_k\}_{k \in \mathcal{K}}$ of data sets is marked as including the object that is to be perceived from the LIDAR data.

## V. Pallet Estimation

The algorithms described in Section IV are next used to design effective heuristic methods to detect pallets from a single LIDAR scan. The detection method is then used as the basis for batch detection and subsequent filtering.

The algorithms described in this section can be used for estimating both the pose and shape of pallets of various types and sizes. Most pallets used in industrial applications have distinctive features that are visible in LIDAR scans, namely two slots (each 20 cm to 40 cm wide) and an overall widths varying from 0.9 m to 1.5 m. Moreover, the two slots generally have the same width and are offset symmetrically with respect to the mid-point of the pallet face. Our pallet estimation algorithms first identify the closest edge in a single laser range finder scan, then look for these distinct features in the edge. The features are identified by invoking calls to `DISTFIND` and `EDGEFIND`.

As a step prior to online filtering, we would like to extract the aforementioned features and detect pallets that lie within the volume of interest. Since our main interest is online filtering, the detection is carried out using only a single scan (see Figure 4) instead of accumulated laser scans. However, assuming that the pallet roll angle is close to the that of the lift truck during active scanning, several detections obtained at different heights can be used for batch detection purposes as well, with essentially no modifications of the detection algorithm. Indeed, this strategy is employed in this work.

Given a single LIDAR scan, the pallet detection algorithm works as follows. Let $\mathcal{X}$ be the set of LIDAR points obtained from the laser range finder sensor mounted on the tine elbow. First, the algorithm culls the points within $\mathcal{X}$ that lie within the region of interest, forming a subset $\mathcal{X}_1$ (see Figure 4). Subsequently, $(z_{\text{pallet}}, a_{\text{pallet}}, w_{\text{pallet}}) := \texttt{EDGEFIND}$ is applied to $\mathcal{X}_1$ to detect the closest edge, which constitutes a candidate pallet face. The resulting width estimate constitutes the first classification feature, $f_1 = (w_{\text{pallet}})$. Second, the algorithm forms a subset $\mathcal{X}_1'$ of $\mathcal{X}_1$ that contains all those points in $\mathcal{X}_1$ that lie in a box centered at $z_1'$ of length $\epsilon$, width $w_1'$, and orientation $a_1'$ (see the blue box in 4). We use $\epsilon = 20$cm. Third, from $\mathcal{X}_1'$ four sets of points $\mathcal{X}_2$, $\mathcal{X}_3$, $\mathcal{X}_4$, and $\mathcal{X}_5$ are extracted. Intuitively, $\mathcal{X}_2$ is the set of all those points in $\mathcal{X}_1'$ that are to the left of the box and are at least 25 cm away from the center of the box. Similarly, $\mathcal{X}_4$ is the set of all those points in $\mathcal{X}_1'$ that are at least 25 cm right of center. The sets $\mathcal{X}_3$ and $\mathcal{X}_5$ are the complements of $\mathcal{X}_2$ and $\mathcal{X}_4$, respectively (Figure 4). The points in $\mathcal{X}_2$ and $\mathcal{X}_3$ are translated such that the origin is the point that is to the left of the box and is 25 cm away from the center. Similarly, the points in $\mathcal{X}_4$ and $\mathcal{X}_5$ are translated such that their origins are to the right of the box and 25 cm away from the center. Subsequently, the algorithm runs the `DISTFIND` function on $\mathcal{X}_i$ for all $i = 2, 3, 4, 5$ and notes the distance returned by the `DISTFIND` algorithm as the feature $f_i$ associated with data set $\mathcal{X}_i$. These features are denoted as $f_2 = (\delta^{\text{far}}_{\text{left}})$, $f_3 = (\delta^{\text{near}}_{\text{left}})$, $f_4 = (\delta^{\text{far}}_{\text{right}})$, and $f_5 = (\delta^{\text{near}}_{\text{right}})$. Note that, intuitively, $\delta^{\text{far}}_{\text{left}}$ is the distance from the *far* side of the *left* slot to the center of the pallet face and similar intuition applies to other features. Finally, the algorithm computes the width $w_{\text{left}}$ and $w_{\text{right}}$ of the left and right slots. Note that this threshold strategy can be implemented within the framework of Section IV-C. If the features are in acceptable bounds with respect to the prespecified nominal set of values, then the algorithm outputs the pallet detection $(z_{\text{pallet}}, a_{\text{pallet}}, w_{\text{pallet}}, w_{\text{left}}, w_{\text{right}}, x_{\text{left}}, x_{\text{right}})$, where $x_{\text{left}}$ and $x_{\text{right}}$ are the distance of the center of left and right slot locations computed directly from the features $f_2, \ldots, f_5$; otherwise it reports no pallet detection. The nominal values of the features as well as their acceptable bounds were hand-tuned in this work; however, they can, in principle, be learned from training data. We leave this for future work.

For batch detection, we actively scan the volume of interest by actuating the lift truck's mast and collecting pallet detections at various heights. A classification algorithm then first checks whether there is a set of detections that span a height consistent with that of typical pallets and that are mutually consistent in terms of Mahalanobis distance. If so the batch detection algorithm outputs the pallet detection averaged over this set of detections as well as the detection heights. Subsequently, we initialize a Kalman filter over the pallet detection states with this average detections and update the filter with any new detections. An active scanning operation is shown in Figure 5.

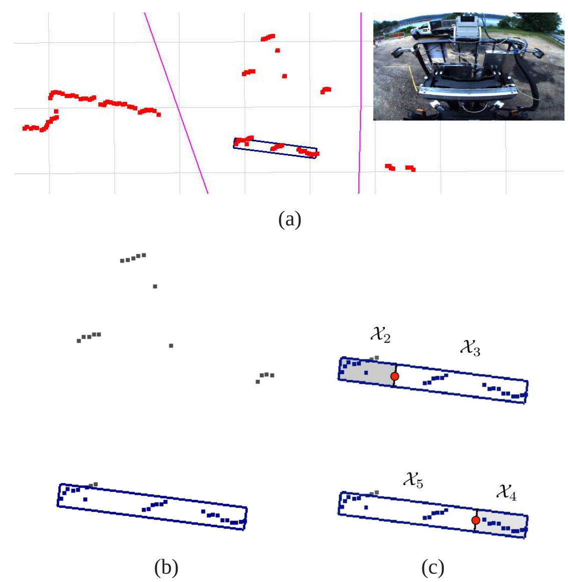
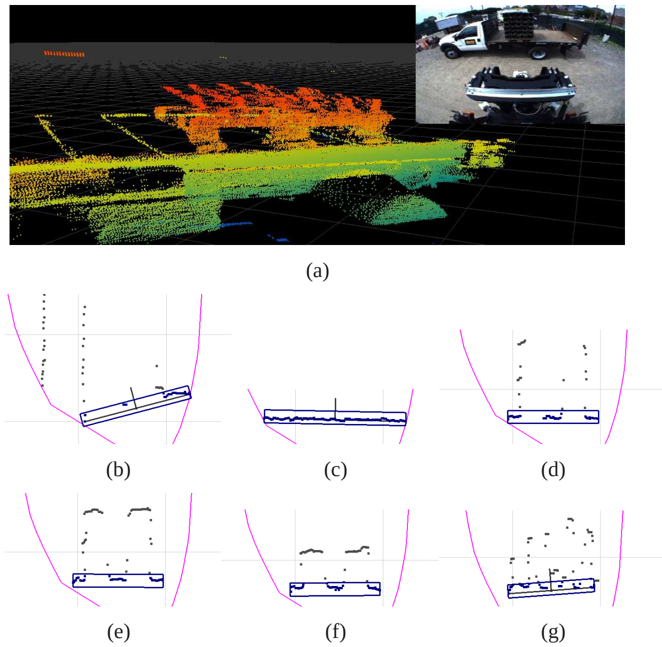



Fig. 4. (a) A single pallet scan and the user gesture projected on the world indicating boundaries of the region of interest (pink). (b) Points in the region of interest as well as the line detection and the associated box. (c) The data sets $\mathcal{X}_i$ with $i = 2, 3, 4, 5$ and their origins shown as red dots.

## VI. TRUCK BED ESTIMATION

This section describes our truck detection algorithms. Our approach to truck estimation employs a Kalman filter to estimate the location of the truck bed online. The user pen gesture projected into the world provides an initial condition for the Kalman filter. Data acquired from the two LIDARs mounted vertically on both sides of the mast are used for detection of the truck bed's height, distance, and orientation, which in turn are used to update the Kalman filter. The truck bed estimate is used in conjunction with the user pen gesture to estimate the drop off location, when placing a pallet on the truck bed.

The truck detection algorithm also operates within the framework described in Section IV-C. From the laser range



(a) (b) (c) (d) (e) (f) (g)

Fig. 5. Pallet detection algorithm as a pallet on a truck bed is being actively scanned. (b-c)LIDAR returns from the undercarriage and the truck bed are rejected as pallet candidates. (d-f) LIDAR returns from the pallet face are identified as the pallet. (g)The load on the pallet is correctly ruled out as a candidate pallet face.

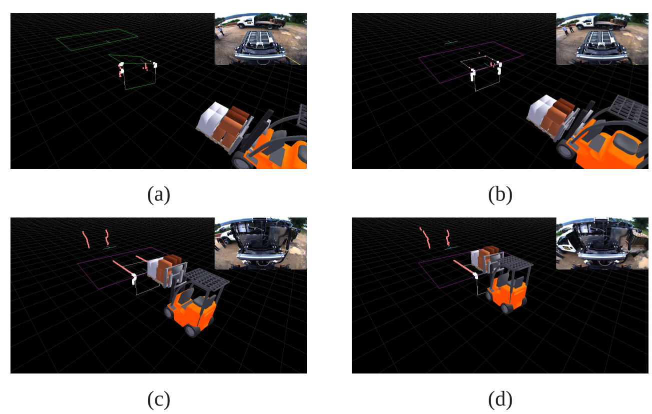
finder mounted to the left of the mast, two features, the distance of the sensor to the truck bed and the height of the truck bed are extracted. The same features are also extracted for the sensor that is mounted to the right of the mast. Subsequently, these features are compared with lower and upper bounds as well as each other to ensure consistency. If found to be within the limits, the detection algorithm outputs the truck bed detection as the height, position, and the orientation of the truck bed.

Strictly speaking, let $\mathcal{X}_{\text{left}}$ and $\mathcal{X}_{\text{right}}$ be the point sets acquired via the laser range finder sensor mounted to the left and right of the mast. Let $\mathcal{X}_1$ be the set of all those points in $\mathcal{X}_{\text{left}}$ that are at least 25 cm above the ground. The truck bed detection algorithm uses DISTFIND to detect the distance $d_{\text{left}}$ of these points to the sensor, which forms the element of the first feature $f_1 = (d_{\text{left}})$ for classification. Let $\mathcal{X}_2$ be the set of all those points in set $\mathcal{X}_1$ that are at least $d_{\text{left}}$ and at most $d_{\text{left}} + \epsilon$ away from the sensor. Moreover, let the points in $\mathcal{X}_2$ be translated such that their center is $d_{\text{left}}$ away from sensor and 5 m above the ground (see Figure 6). Next, the algorithm employs DISTFIND to determine the distance $h_{\text{left}}$ of these points from the ground, which is noted as the second feature $f_2 = (h_{\text{left}})$. Similarly, we employ the same procedure with $\mathcal{X}_{\text{right}}$ to obtain the sets $\mathcal{X}_3$ and $\mathcal{X}_4$ and two additional features, $f_3 = (d_{\text{right}})$ and $f_4 = (h_{\text{right}})$. Finally, the algorithm checks whether all the extracted features are within acceptable bounds, and for differences between the features observed via the left sensor and those observed with the right sensor, in which case it outputs the truck bed detection $(z_{\text{truck}}, a_{\text{truck}}, h_{\text{truck}})$; otherwise, the detector outputs


Fig. 6. Truck bed detection algorithm depicted with raw data (false-colored by height) acquired from the sensor mounted on the right of the mast.

no detection. The height $h_{\text{truck}}$ of the truck bed is computed as the average of $h_{\text{left}}$ and $h_{\text{right}}$. The location $z_{\text{truck}}$ on the other hand, is the intersection of the line that passes through $z_{\text{left}}$ and $z_{\text{right}}$ with the user pen-gesture ray projected on the plane of truck bed height and parallel to the ground (see Figure 6).

A Kalman filter is initialized with $(z^{\text{prior}}_{\text{truck}}, a^{\text{prior}}_{\text{truck}}, h^{\text{prior}}_{\text{truck}})$, where $h^{\text{prior}}_{\text{truck}}$ is a prior on the truck bed height set to 1 m in our experiments, and $z^{\text{prior}}_{\text{truck}}$ and $a^{\text{prior}}_{\text{truck}}$ are the prior position and orientation obtained from the user gesture as follows. First, a circle center is fitted to the user pen-gesture and projected as a ray in to the world. $z^{\text{prior}}_{\text{truck}}$ is the intersection of this ray with the plane of height $h^{\text{prior}}_{\text{truck}}$ parallel to the ground. $a^{\text{prior}}_{\text{truck}}$, on the other hand, is the unit vector oriented from the bot to $z^{\text{prior}}_{\text{truck}}$. The filter is updated with each positive detection. Figure 7 shows a pallet drop off operation, in which truck bed estimation is used to determine the drop-off location.



(a) (b) (c) (d)

Fig. 7. Truck bed estimation. (a) The initial estimate of the truck bed is determined from the user pen gesture; however, the truck bed detection algorithm initially outputs no detection. (b-d) As the robot drives towards the truck, the sensors get LIDAR returns from the truck bed, and the Kalman filter is updated accordingly. (d) The bot drops off the pallet to the part of the truck bed indicated by the user pen gesture.

## VII. Control Algorithms

This section presents a feedback control algorithm that can be used to steer 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. In the next section, we provide experimental results using this controller in closed-loop operation with the pallet and truck perception algorithms presented in the previous sections.

Let $z_{\text{initial}}$ and $a_{\text{initial}}$ be the robot's initial position and orientation, where $z_{\text{initial}}$ is a coordinate Euclidean plane and $a_{\text{initial}}$ is a normalized two-dimensional vector. Similarly, let $z_{\text{final}}$ and $a_{\text{final}}$ be the desired final position and orientation of the robot. (In our application, $z_{\text{final}}$ and $a_{\text{final}}$ represent the pallet position and orientation.) Without loss of generality, let $z_{\text{final}} = (0, 0)$ be the origin of the coordinate system and $a_{\text{final}} = (1, 0)$ be oriented toward the $X$-axis (see Figure 8). Similarly, let $e_y$ be the distance of $z_{\text{initial}}$ to $z_{\text{final}}$ along the direction orthogonal to $a_{\text{final}}$ and let $e_\theta$ be the angle between the vectors $a_{\text{initial}}$ and $a_{\text{final}}$, i.e., $e_\theta = \cos^{-1}(a_{\text{initial}} \cdot a_{\text{final}})$. Finally, let $\delta$ be the steering control input to the robot. In this work, we use the following steering control strategy for pallet engagement operations:

$$\delta = K_y \tan^{-1}(e_y) + K_\theta e_\theta, \tag{7}$$

where $K_y$ and $K_\theta$ are controller parameters. Assuming a Dubins vehicle model [22] of the robot as in

$$\dot{z} = (\cos\theta, \sin\theta), \tag{8}$$
$$\dot{\theta} = \tan^{-1}(\delta), \tag{9}$$

the nonlinear control law (7) 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 [23].

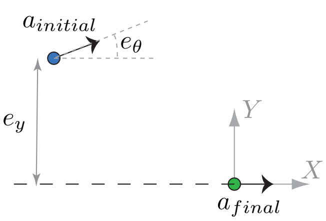
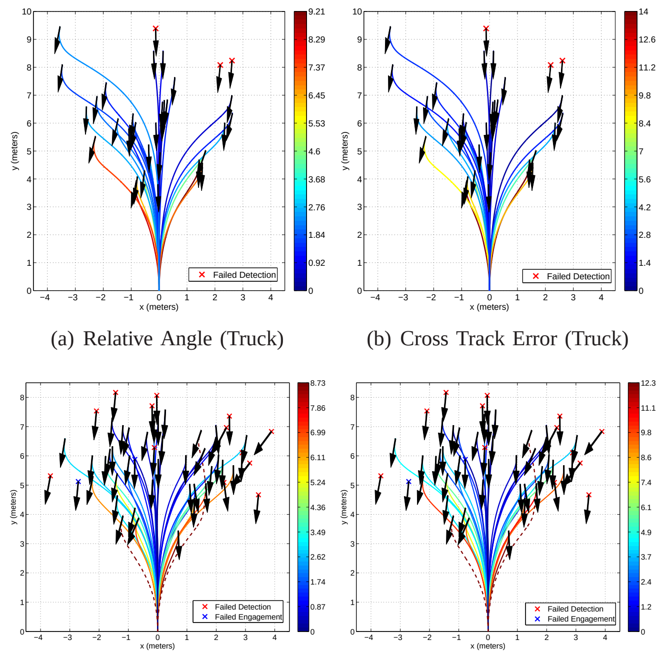



Fig. 8. Illustration of the controller algorithm

## VIII. Experimental Results

This section analyzes the pallet engagement system described above. The closed-loop pallet engagement software was tested extensively on the hardware described in Section III, at two testing sites, one on the MIT campus, in Cambridge, MA, and the second at Fort Belvoir, a U.S. Army base in Virginia. Both testing sites have packed gravel terrain with small rocks and mud. In these experiments, we commanded the bot to pick up pallets from different locations on the ground as well as from truck beds, and 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 and, within each type, the pallets varied in their 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.

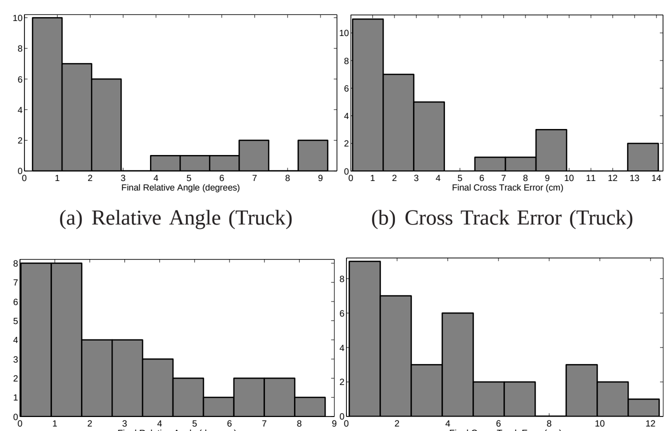
Figure 9 shows a plot of the success and failures of the pallet pickup tests, together with final relative angle and cross track error in each experiment (see Figure 10 for histograms). Note that most of the failures are due to pallet detection,




(a) Relative Angle (Truck) (b) Cross Track Error (Truck)

(c) Relative Angle (Ground) (d) Cross Track Error (Ground)

Fig. 9. Results of the validation tests for pallet engagements from (a), (b) a truck bed and (c), (d) the ground. Each path represents the robot's trajectory during a successful pickup. A red 'x' denotes the initial position of the robot for a failed engagement. 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 and (b), (d) the cross track error immediately prior to insertion.




(a) Relative Angle (Truck) (b) Cross Track Error (Truck)

(c) Relative Angle (Ground) (d) Cross Track Error (Ground)

Fig. 10. Histograms that depict the resulting error immediately prior to the forklift inserting the tines in the pallet slots, for a series of tests. Figures (a) and (c) correspond to the relative angle between the vehicle's forward direction and the pallet normal for engagements off of a truck and off of the ground, respectively. The histograms in (b) and (d) present the final lateral cross track error for the successful engagements.

and they occur when the bot starts longitudinally 7.5 meters and/or laterally 3 meters or more away from the pallet. In most of these cases, the resolution of the laser range finder seems insufficient for the data to include returns from the pallet surface. In some other cases, we have seen pallet engagements where the bot ended up pushing the pallet and turning up to 10 degrees; we classified these cases as failures. 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 where we define a successful engagement as one in which the forklift inserted 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.

## IX. Conclusions

We presented a novel coupled perception and control algorithm for an outdoor robotic forklift tasked with manipulation of unknown pallets. We have also shown an experimental demonstration of the algorithms on a full-sized forklift.

Our current research includes extending our perception algorithms to detect multiple pallets and to detect pallets without the help of a user gesture. We also plan to develop a path planning capability that identifies trajectories that minimize the resulting uncertainty in the pallet pose, thereby increasing the likelihood of successful engagement.

## Acknowledgments

We gratefully acknowledge the support of the U.S. Army Logistics Innovation Agency (LIA) and the U.S. Army Combined Arms Support Command (CASCOM).

This work was sponsored by the Department of the Air Force under Air Force Contract FA8721-05-C-0002. Any opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

## References

[1] S. Teller *et al.*, "A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, May 2010.

[2] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Coordination and decentralized cooperation of multiple mobile manipulators," *J. of Robotic Systems*, vol. 13, no. 11, pp. 755–764, 1996.

[3] R. A. Grupen and J. A. Coelho, "Acquiring state from control dynamics to learn grasping policies for robot hands." *Advanced Robotics*, vol. 16, no. 5, pp. 427–443, 2002.

[4] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int'l J. of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[5] J. Park and O. Khatib, "Robust haptic teleoperation of a mobile manipulation platform," in *Experimental Robotics IX*, ser. STAR Springer Tracts in Advanced Robotics, M. Ang and O. Khatib, Eds., 2006, vol. 21, pp. 543–554.

[6] D. Berenson, J. Kuffner, and H. Choset, "An optimization approach to planning for mobile manipulation," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2008.

[7] D. Kragic, L. Petersson, and H. I. Christensen, "Visually guided manipulation tasks," *Robotics and Autonomous Systems*, vol. 40, no. 2–3, pp. 193–203, Aug. 2002.

[8] R. Brooks, L. Aryananda, A. Edsinger, P. Fitzpatrick, C. Kemp, U. O'Reilly, E. Torres-Jara, P. Varshavskaya, and J. Weber, "Sensing and manipulating built-for-human environments," *Int'l J. of Humanoid Robotics*, vol. 1, no. 1, pp. 1–28, 2004.

[9] P. Deegan, R. Grupen, A. Hanson, E. Horrell, S. Ou, E. Riseman, S. Sen, B. Thibodeau, A. Williams, and D. Xie, "Mobile manipulators for assisted living in residential settings," *Autonomous Robots*, 2007.

[10] M. Hebert, "Outdoor scene analysis using range data," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 1986, pp. 1426–1432.

[11] R. Hoffman and A. Jain, "Segmentation and classification of range images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 608–620, Sept. 1987.

[12] T. Newman, P. Flynn, and A. Jain, "Model-based classification of quadric surfaces," *CVGIP: Image Understanding*, vol. 58, no. 2, pp. 235–249, 1993.

[13] P. Gotardo, O. Bellon, and L. Silva, "Range image segmentation by surface extraction using an improved robust estimator," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2003, pp. 33–38.

[14] X. Yu, T. Bui, and A. Krzyzak, "Robust estimation for range image segmentation and reconstruction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 530–538, May 1994.

[15] H. Wang and D. Suter, "MDPE: A very robust estimator for model fitting and range image segmentation," *Int'l J. of Computer Vision*, vol. 59, no. 2, pp. 139–166, Sept. 2004.

[16] R. Bostelman, T. Hong, and T. Chang, "Visualization of pallets," in *Proc. SPIE Optics East Conf.*, Oct. 2006.

[17] R. Cucchiara, M. Piccardi, and A. Prati, "Focus based feature extraction for pallets recognition," in *Proc. British Machine Vision Conf.*, 2000.

[18] D. Lecking, O. Wulf, and B. Wagner, "Variable pallet pick-up for automatic guided vehicles in industrial environments," in *Proc. IEEE Conf. on Emerging Technologies and Factory Automation*, May 2006, pp. 1169–1174.

[19] A. Correa, M. R. Walter, L. Fletcher, J. Glass, S. Teller, and R. Davis, "Multimodal interaction with an autonomous forklift," in *Proc. ACM/IEEE Int'l Conf. on Human-Robot Interaction (HRI)*, Osaka, Japan, March 2010.

[20] B. Schölkopf and A. Smola, *Learning with Kernels*. MIT Press, 2002.

[21] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.

[22] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American J. of Mathematics*, vol. 97, no. 3, pp. 497–516, 1957.

[23] G. Hoffmann, C. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and testing," in *American Control Conf.*, 2007.