2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

3D Perception for Accurate Row Following: Methodology and Results

Ji Zhang, Andrew Chambers, Silvio Maeta, Marcel Bergerman, and Sanjiv Singh

***Abstract*— Rows of trees such as in orchards, planted in straight parallel lines can provide navigation cues for autonomous machines that operate in between them. When the tree canopies are well managed, tree rows appear similar to corridor walls and a simple 2D sensing scheme suffices. However, when the tree canopies are three dimensional, or ground vegetation occludes tree trunks, it is necessary to use a three dimensional sensing mode. An additional complication in prolific canopies is that GPS is not reliable and hence is not suitable to register data from sensors onboard a traversing vehicle. Here, we present a method to register 3D data from a lidar sensor onboard a vehicle that must accurately determine its pose relative to the rows. We first register point cloud into a common reference frame and then determine the position of tree rows and trunks in the vicinity to determine the vehicle pose. Our method is tested online and with data from commercial orchards. Experimental results show that the accuracy is sufficient to enable accurate traversal between tree rows even when tree canopies do not approximate planar walls.**

I. Introduction

Here, we address the problem of guidance for autonomous row following in environments such as orchards. The trees in modern orchards are planted in straight and parallel pattern that form tree rows. Our method automatically detects and allows a vehicle to drive along the tree rows, enabling the vehicle to perform tasks such as spraying, mowing, and transportation. Further, we want the vehicle to be capable of keeping a constant distance to one side of the trees during the row following, such that the vehicle can function as a platform, carrying workers to work on the trees while driving.

When the tree rows are well-structured to form planar walls, a horizontally placed planar laser scanner can provide enough information for the row following [1]. However, when the tree canopies are irregular and high weeds are present, it is difficult to use the returns from a planar laser to guide the vehicle while maintaining a fixed offset between the tree rows. Here, we use a lidar sensor, which consists of a planar laser driven by a motor for roll rotation, and an encoder for measuring the rotation angle. The lidar scans around two orthogonal axes and produces a 3D point cloud.

The lidar point cloud is registered by the vehicle motion measured by wheel encoders and refined by the Iterative Closest Point (ICP) method [2]. Then, the point cloud is fitted into two parallel straight lines based on least square regression using a RANdom SAmple Consensus (RANSAC) algorithm [3], which represent the tree rows on both sides of the vehicle. The straight lines are integrated into an Extended Kalman Filter (EKF) [4] to reduce the line fitting noise. The EKF takes the odometry measurements into the prediction step and the detected tree rows into the update step. To prevent faulty tree row detection from being integrated in the EKF, a checking mechanism is implemented that removes the EKF update step if the straight lines are fitted incorrectly. Finally, the lateral offsets of the straight lines are refined using the density information in the lidar point cloud.

J. Zhang, A. Chambers, S. Maeta, M. Bergerman, and S. Singh are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213. Email: {zhangji, achamber, smaeta}@andrew.cmu.edu and {marcel, ssingh}@cmu.edu.

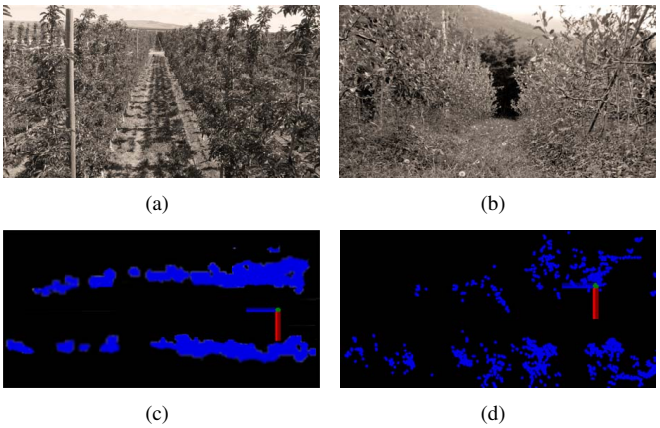


Fig. 1. An example of regular tree rows (a) where the tree canopies form planar "fruit walls", and irregular tree rows (b) where the tree canopies are three dimensional and high weeds are present. (c) and (d) show the point cloud perceived by a horizontally placed planar laser scanner corresponding to (a) and (b), in top-down view. In a three dimensional orchard, a planar laser scanner cannot provide sufficient information to detect the tree rows, while a 3D lidar becomes necessary.

To follow the tree rows with a constant offset to one side of the trees, we also detect the closest tree trunk to the vehicle on each side. A particle filter [4] based method is proposed, which uses a number of 500 particles on each side. The motion model is provided by the odometry measurements, while the observation model is given by the lidar point cloud. The particles are resampled by low variance resampling [4]. In the resampling process, the method selects the closest tree trunk to the vehicle, and draws new particles from those that are associated with the selected tree trunk.

The rest of this paper is organized as follows. In section II, we present related work. In section III, we state our problem. The lidar hardware and the software system are overviewed in Section IV. The proposed method is presented with details in Section V. Experimental results are shown in Section VI and conclusions are made in Section VII.

II. Related Work

Autonomous row following has become a popular research area [5]. The task involves detecting a pathway for a vehicle to follow, using environmental sensors such as cameras or

laser scanners. Most of the work focuses on road lane detection and following [6]. On a structured road, the road surface is often homogeneously colored and landmarks such as traffic lines are detectable to cameras [5]. The 3D structure on road boarders can also provide information for laser scanners to detect the road [7]. For example, Tsogas et al.'s method detect the road boundaries by fusing data from a camera and a laser scanner with a prior map [8]. On an unstructured road, dominant orientations and vehicle wheel tracks are often used to recognize the road [6]. The color difference between road surface and boarder areas, if available, can also be used [9]. Compared to [6], [9], our problem is different in that robust color pattern invariant to lighting, tree type, and season changes is unavailable for identifying the trees. However, the 3D structure formed by the tree rows provides a corridor environment, making lidar suitable for the task.

Autonomous row following is also available for agricultural applications [10], [11]. For example, Astrand and Baerveldt adopt a monocular camera and detect the crops planted in straight rows using Hough transform [12]. Also using Hough transform, Satow et al.'s method detects the crop rows from 3D point cloud perceived by a laser scanner through a rotating mirror [13]. Biber et al.'s navigation system performs row following using data from a lidar, an RTK GPS, an inertial sensor, and odometry [14]. In literature [10]–[14], the methods only deal with low crops. Comparing these methods to our problem, the 3D structure formed by the tall trees makes the task more complicated and challenging.

Few methods focus on the row following problem between tall trees in an orchard. In [15], the vehicle follows the tree rows by driving along a planed path using reading from an RTK GPS. A 3D laser scanner provides a point cloud from the tree rows for obstacle detection and avoidance. In [1], [16], a fixed planar laser scanner is used in front of the vehicle for tree row detection. Consider that the point returns from a fixed laser scanner cannot represent the 3D environment, the methods can fail if the tree rows are ill-structured. Our proposed method detects the tree rows from 3D point cloud perceived by a lidar sensor. Further, our method also detects the tree trunk that is the closest to the vehicle on each side, which provides more information to assist the vehicle navigation between the tree rows.

III. Problem Statement

From a robotics perspective, the problem addressed in this paper is to detect the tree rows and trunks in an orchard using a 3D lidar; and to use that information to guide an autonomous vehicle to follow the rows. Toward this end we make the following assumptions:

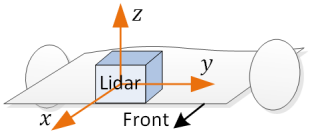


Fig. 2. Vehicle coordinate system $\{V\}$.

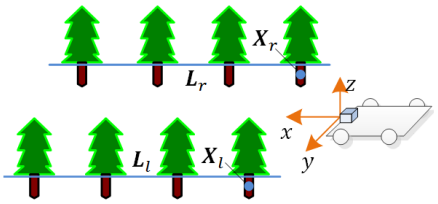


Fig. 3. Problem statement. $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$ are two parallel straight lines representing the tree rows, and $\boldsymbol{X}_l$ and $\boldsymbol{X}_r$ are two points representing the centers of the closest tree trunks on both sides of the vehicle. Our task consists in detecting $\boldsymbol{L}_l$, $\boldsymbol{L}_r$, $\boldsymbol{X}_l$, and $\boldsymbol{X}_r$ from the lidar point cloud.

1) The ground is relatively flat within a short range. This allows us to register the lidar point cloud using the vehicle odometry, which measures planar motion.
2) The extrinsic parameters of the lidar are known from a calibration process.

As a convention in this paper, we use a right subscript k, $k \in Z^+$ to indicate each lidar frame. We define $\{V\}$, the vehicle coordinate system, as follows: the origin of $\{V\}$ is at the center of the lidar; the y-axis is parallel to the vehicle axles and points toward the left side; the x-axis points forward; and the z-axis points upward. A point in $\{V\}$ at frame k is denoted as $\boldsymbol{X}_k = [x_k,\ y_k,\ z_k]^T$.

Let $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$ be two horizontal, parallel lines in the vehicle coordinate system $\{V\}$ representing the tree rows on the left and right sides of the vehicle. Also, let $\boldsymbol{X}_l$ and $\boldsymbol{X}_r$ be the two points in $\{V\}$ representing the centers of the tree trunks closest to the vehicle on each side. Let $\mathcal{P}_k$ be the point cloud perceived by the lidar at frame k. Since the extrinsic parameters of the lidar are known from calibration, $\mathcal{P}_k$ can be projected into the vehicle coordinate system $\{V\}$. Further, let Δx, Δy, and $\Delta\theta$ be the vehicle translation and heading rotation between frames $k-1$ and k. The quantities Δx, Δy, and $\Delta\theta$ are measured by the odometry and used to register the lidar point cloud. With the terms now defined, the problem statement can be more formally stated as:

Problem 1: Given the point cloud, $\mathcal{P}_k$, $k \in Z^+$, and the vehicle motion, Δx, Δy, and $\Delta\theta$, compute the tree rows, $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$, and tree trunks, $\boldsymbol{X}_l$ and $\boldsymbol{X}_r$, for each lidar frame k.

IV. System Overview

A. Lidar Hardware

The lidar used in this study is a custom-built unit based on the Hokuyo laser scanner connected to a motor. The laser

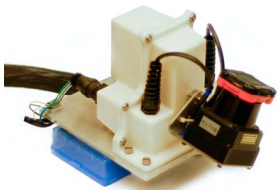


Fig. 4. The 3D lidar used in this study consists of a Hokuyo laser scanner driven by a motor for rotational movement, and an encoder that measures the rotation angle. The laser scanner has a field of view of 180° with a resolution of 0.25°. The frame rate is 40Hz. The motor is controlled to rotate from -55° to 55° with the horizontal position as zero. A microcontroller controls the motor and synchronizes the laser scanner and encoder data.

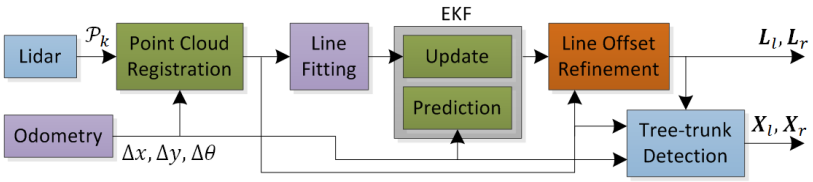


Fig. 5. Block diagram of the lidar software system.

scanner has a field of view of $180°$ with $0.25°$ resolution and 40Hz frame rate. An embedded microcontroller controls the rotation of the motor at a rotation speed of $100°/s$ and measures the rotation angle through an optical quadrature encoder with a resolution of $0.25°$. The microcontroller also manages the time synchronization between the laser scanner and the encoder. The laser scanner provides a synchronization pulse that indicates the start of each scanning cycle. The synchronization pulse triggers the microcontroller to report the encoder reading. Through a serial cable, the data is sent to a computer, where the laser points are projected into 3D space. When projecting the laser points, the rotation angle of the laser scanner is found by linearly interpolating the encoder reading within the laser scanning cycles.

B. Software System Overview

Fig. 5 shows a diagram of the software system. The point cloud is registered via the odometry measurements, Δx, Δy, and $\Delta\theta$, and refined by the ICP method [2]. A RANSAC algorithm [3] is then used to fit the registered point cloud into two straight lines, representing the tree rows on both sides of the vehicle. The detected tree rows are filtered by an EKF [4] using the odometry measurements, and the lateral offsets of the tree rows are refined to compute the outputs, $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$. To detect the tree trunks, $\boldsymbol{X}_l$ and $\boldsymbol{X}_r$, we use a particle filter-based [4] method that takes as inputs the registered point cloud, the detected tree rows, $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$, and the odometry measurements, Δx, Δy, and $\Delta\theta$. Section V presents each block in the software diagram in detail.

V. Tree Row and Trunk Detection

This section describes the main contribution of this paper, namely, a methodology to extract tree row and trunk information from a 3D lidar point cloud that can be used by an autonomous vehicle to follow the row between the trees–either keeping a constant distance from the centerline, or a constant distance to one side of the trees.

A. Point Cloud Registration

The point cloud perceived in a single lidar frame, $\mathcal{P}_k$, contains points located on a 2D slice of the environment. The first step in creating a 3D representation is to register each point cloud with the odometry measurements and combine it with previous ones. Let $\mathcal{Q}_k$ be the registered point cloud at frame k. Using the vehicle motion measured by the odometry, Δx, Δy, and $\Delta\theta$, a point $\boldsymbol{X}_{k-1}$, $\boldsymbol{X}_{k-1} \in \mathcal{Q}_{k-1}$, is projected onto frame k as:

$$\boldsymbol{X}'_k = \mathbf{R}_z(\Delta\theta)\boldsymbol{X}_{k-1} - [\Delta x,\ \Delta y,\ 0]^T, \tag{1}$$

where $\mathbf{R}_z(\cdot)$ is the rotation matrix around the z-axis. To reduce the effect of wheel slip, which causes inaccuracy in odometry measurements, the ICP method is used to refine Δx, Δy, and $\Delta\theta$. This method finds the closest point in $\{\boldsymbol{X}'_k\}$ to match each point in the current lidar data, $\mathcal{P}_k$, and uses the matched points to recompute Δx, Δy, and $\Delta\theta$, iteratively. Then, $\mathcal{Q}_k$ is calculated as the combination of the current lidar data, $\mathcal{P}_k$, and the points projected from $\mathcal{Q}_{k-1}$,

$$\mathcal{Q}_k = \mathcal{P}_k \bigcup \{\boldsymbol{X}'_k\}. \tag{2}$$

We use a decay time to "forget" points that are too distant in the past–in our case, each point is kept for five seconds. The next step is to use $\mathcal{Q}_k$ to detect the tree rows.

B. Line Fitting for Tree Row Detection

The information about the 3D point cloud at frame k, $\mathcal{Q}_k$, is processed by a RANSAC algorithm to yield the supporting lines of the tree rows. Recall that $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$ are two parallel lines representing the tree rows on the left and right sides of the vehicle. Let they be represented as:

$$\boldsymbol{L}_l:\ y = ax + b_l, \tag{3}$$

$$\boldsymbol{L}_r:\ y = ax + b_r, \tag{4}$$

where a, b_l, and b_r are the line coefficients.

To determine these coefficients, we use a procedure similar to least square line regression, except that in this case we need to fit two parallel lines simultaneously. Let $\mathcal{L}_k$ and $\mathcal{R}_k$ be two subsets of $\mathcal{Q}_k$ on the left and right sides of the vehicle, respectively. We use the points in $\mathcal{L}_k$ and $\mathcal{R}_k$ to fit the lines. Define a regret as the sum of the squared offset form each point in $\mathcal{L}_k$ and $\mathcal{R}_k$ to its corresponding lines,

$$R = \sum_{i\in\mathcal{L}_k} (y_k^i - ax_k^i - b_l)^2 + \sum_{i\in\mathcal{R}_k} (y_k^i - ax_k^i - b_r)^2. \tag{5}$$

To solve the line fitting problem, we minimize the regret, R, with respect to the coefficients, a, b_l, and b_r,

$$\{a,\ b_l,\ b_r\} = \arg\min_{a,b_l,b_r} R. \tag{6}$$

In (6), the condition for R to be the minimum is that its partial derivatives with respect to a, b_l, and b_r are equal zero, $\partial R/\partial a = 0$, $\partial R/\partial b_l = 0$, and $\partial R/\partial b_r = 0$. Consequently, we can derive a linear equation for a, b_l, and b_r as follows,

$$\mathbf{A}[a,\ b_l,\ b_r]^T = \boldsymbol{b}, \tag{7}$$

where

$$\mathbf{A} = \begin{bmatrix} \sum_{i\in\mathcal{L}_k} x_k^i, & |\mathcal{L}_k|, & 0 \\ \sum_{i\in\mathcal{R}_k} x_k^i, & 0, & |\mathcal{R}_k| \\ \sum_{i\in\mathcal{L}_k\bigcup\mathcal{R}_k} (x_k^i)^2, & \sum_{i\in\mathcal{L}_k} x_k^i, & \sum_{i\in\mathcal{R}_k} x_k^i \end{bmatrix}, \tag{8}$$

$$\boldsymbol{b} = [\sum_{i\in\mathcal{L}_k} y_k^i, \sum_{i\in\mathcal{R}_k} y_k^i, \sum_{i\in\mathcal{L}_k\bigcup\mathcal{R}_k} x_k^i y_k^i]^T. \tag{9}$$

Solving (7), we can compute $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$. To guarantee that (7) is mathematically solvable, at least one point has to be selected on each side of the vehicle, $|\mathcal{L}_k|, |\mathcal{R}_k| \geq 1$, and at least three points have to be selected on both sides of the vehicle, $|\mathcal{L}_k| + |\mathcal{R}_k| \geq 3$.

The line fitting method discussed above is implemented in a RANSAC algorithm, as shown in Algorithm 1. In each iteration, we randomly select two sets of points, $\mathcal{L}_k$ and $\mathcal{R}_k$, one on each side of the vehicle, and use $\mathcal{L}_k$ and $\mathcal{R}_k$ to compute the tree lines' coefficients (line 5 in Algorithm 1). Then, the points in $\mathcal{Q}_k$ are evaluated in terms of the distance from each point to the lines, and a set of inliers is selected from $\mathcal{Q}_k$ on each side (line 6). If the inlier numbers on both sides are sufficient, the lines are computed again using the selected inliers (line 8). The algorithm terminates if the mean squared distance (MSD) from the points to their corresponding lines is smaller than a threshold, or the maximum number of iterations is reached. The algorithm returns $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$ with the minimum MSD found .

Algorithm 1: Line Fitting

```
1  input : Q_k
2  output : L_l and L_r
3  begin
4      for a certain number of iterations do
5          From Q_k, randomly select a set of points on the left
           side, L_k, and a set of points on the right side, R_k,
           where |L_k|, |R_k| ≥ 1 and |L_k| + |R_k| ≥ 3, compute
           L_l and L_r based on (7);
6          Compute the distance from each point in Q_k to L_l
           and L_r, select a set of inliers from Q_k on the left
           and right sides, respectively, replace L_k and R_k with
           the selected inlier sets;
7          if |L_k| and |R_k| are both larger than a threshold then
8              Compute a_k, b_l, and b_r again using L_k and R_k
               based on (7), then, compute the squared distance
               (SD) from each point in L_k to L_l, and from each
               point in R_k to L_r;
9              if the mean SD is smaller than a threshold then
10                 Break;
11             end
12         end
13     end
14     Return L_l and L_r with the minimum mean SD found.
15 end
```

C. EKF Filtering for Fitted Lines

The fitted lines from the lidar point cloud contain a considerable amount of noise. To deal with the noise, an EKF is used. The filter takes in odometry measurements in the prediction step, and the fitted lines in the update step. The covariance matrices for both steps are pre-defined. Since the two lines, $\boldsymbol{L}_l$ and $\boldsymbol{L}_r$, are integrated in the same way, in the following, we only use the left line as an example.

Let $\boldsymbol{L}_{k|k}$ and $\boldsymbol{L}_{k|k-1}$ be two lines representing the EKF predicted state estimate and updated state estimate corresponding to $\boldsymbol{L}_l$. Recall that Δx, Δy, and $\Delta\theta$ represent the vehicle motion measured by the odometry. The EKF prediction step calculates $\boldsymbol{L}_{k|k-1}$ from $\boldsymbol{L}_{k-1|k-1}$ using the odometry measurements,

$$a_{k|k-1} = \tan(\tan^{-1} a_{k-1|k-1} - \Delta\theta), \tag{10}$$

$$b_{k|k-1} = b_{k-1|k-1} + \Delta x \sin\tan^{-1} a_{k-1|k-1} - \Delta y \cos\tan^{-1} a_{k-1|k-1}, \tag{11}$$

where $a_{k-1|k-1}$, $b_{k-1|k-1}$ and $a_{k|k-1}$, $b_{k|k-1}$ are the line coefficients of $\boldsymbol{L}_{k|k-1}$ and $\boldsymbol{L}_{k-1|k-1}$. The EKF update step calculates $\boldsymbol{L}_{k|k}$ from $\boldsymbol{L}_{k|k-1}$ using the fitted line $\boldsymbol{L}_l$,

$$[a_{k|k},\ b_{k|k}]^T = [a_{k|k-1},\ b_{k|k-1}]^T + \mathbf{K}[a - a_{k|k-1},\ b_l - b_{k|k-1}]^T, \tag{12}$$

where $\mathbf{K}$ is the Kalman gain generated by the EKF.

To prevent faulty tree row detection, we monitor the difference between the predicted state estimate, $\boldsymbol{L}_{k|k-1}$, and the fitted line, $\boldsymbol{L}_l$. Let $\Sigma_{k|k-1}$ be the covariance matrix of $\boldsymbol{L}_{k|k-1}$, where $\Sigma_{k|k-1}$ is generated by the EKF. A squared Mahalanobis distance is defined as an evaluation metric,

$$d = [a - a_{k|k-1},\ b_l - b_{k|k-1}]\Sigma_{k|k-1}^{-1}[a - a_{k|k-1},\ b_l - b_{k|k-1}]^T. \tag{13}$$

If d is larger than a threshold, $\boldsymbol{L}_l$ is considered as faulty, and the corresponding EKF update step is ignored.

D. Line Lateral Offset Refinement

The next step is to refine the lateral offsets of the fitted lines using the point density information in the lidar point cloud, $\mathcal{Q}_k$. This step is necessary because, in our experience, tree trunks and large branches generate denser and stabler lidar returns than leaves and grass. Using the points from trunks and large branches of the trees usually produces a more reliable estimate of the line offset. Similarly to the previous section, the offset refinement is equally computed for the left and right sides, therefore here we use the left line as an example.

With the fitted lines from the previous section, we rotate the point cloud around the z-axis such that the lines are parallel to the x-axis in the vehicle coordinate frame $\{V\}$, as shown in Fig. 6(b). Then, we separate the point cloud into a few horizontal layers, as shown in Fig. 6(a). For the types of crops in which we operate, we found that five layers spaced 0.5m apart are sufficient to characterize the trees' profile. Let j indicate the layers, $j \in \{1,\ 2, ...,\ 5\}$. Recall that b_l is the offset for the left line, $\boldsymbol{L}_l$. To refine its value, we change b_l within an 1m range above and below its original value. Then, for each b_l, we count the number of points in a rectangular region around $\boldsymbol{L}_l$, as illustrated by the gray band in Fig. 6(b). The number of points on layer j is denoted as $n(b_l, j)$.

Fig. 6(a) gives an intuitive illustration of the distribution of $n(b_l, j)$ with respect to b_l on each layer j, as the red

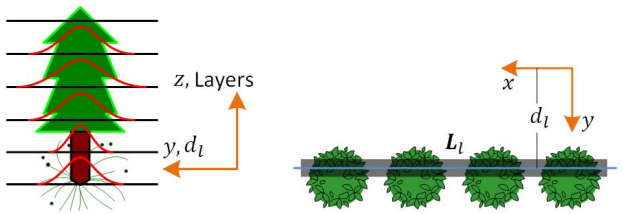


Fig. 6. Line offset refinement. For each height represented by the black horizontal lines in (a), we count the number of points in a rectangular band (represented by the gray colored band in (b)) around the fitted line, $\boldsymbol{L}_l$. At each height, the number of points with respect to the line offset, d_l, follows a normal distribution as represented by the red curves in (a). This information is used to refine the line fitting.

colored curves. On the layers containing tree trunks and large branches, the distribution is more concentrated than in the other layers. Let $p_j(b_l)$ be the distribution of $n(b_l, j)$ on layer j; $p_j(b_l)$ is a function of the line offset, b_l. We want to use the variance of $p_j(b_l)$, denoted as σ_j^2, to evaluate the point density on a certain layer. To refine the line offset, we first select the layer where σ_j^2 is the minimum,

$$j^* = \arg\min_j \sigma_j^2. \tag{14}$$

Then, b_l is set at the point where $p_j(b_l)$ is the maximum on the selected layer,

$$b_l^* = \arg\max_{b_l} p_j(b_l),\ \ j = j^*. \tag{15}$$

After refining b_l, the height of $\boldsymbol{L}_l$ is also determined, at the middle of the selected layer.

E. Tree Trunk Detection

With the tree rows detected, we now find the tree trunks closest to the vehicle on each side using a particle filter. Since the tree trunks follow a multimodal distribution, a particle filter is suitable for solving the problem. The motion model of the particle filter is provided by the odometry measurements, and the observation model is given by the lidar point cloud, $\mathcal{Q}_k$. Again, the procedure is identical on both sides, we explain it using the left side as an example.

Recall that $\boldsymbol{X}_l$ is a point in the vehicle coordinate system $\{V\}$ representing the center of the closest tree trunk on the left side. Let $\boldsymbol{X}_k^p$ be a particle at frame k corresponding to $\boldsymbol{X}_l$. The motion model of the particle filter can be expressed in the same way as (1):

$$\boldsymbol{X}_k^p = \mathbf{R}_z(\Delta\theta)\boldsymbol{X}_{k-1}^p - [\Delta x,\ \Delta y,\ 0]^T, \tag{16}$$

where $\mathbf{R}_z(\cdot)$ is the rotation matrix around the z-axis.

For the observation model, let us assume that the lidar points returned from a tree trunk follow a 3D Gaussian distribution. In the vehicle coordinate frame $\{V\}$, the distribution can be illustrated by an ellipsoid shown as the blue colored region in Fig. 7(a). The long axis of the ellipsoid points in the z-direction, and the two short axes point in the x- and y- directions with equal length. Let Σ_T be a 3×3 covariance matrix corresponding to the Gaussian distribution. The probability that a point $\boldsymbol{X}_k$, $\boldsymbol{X}_k \in \mathcal{Q}_k$, belongs to the tree trunk centered at $\boldsymbol{X}_k^p$ is computed as:

$$p(\boldsymbol{X}_k|\boldsymbol{X}_k^p) = \frac{1}{(2\pi)^{\frac{3}{2}}|\Sigma_T|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{X}_k - \boldsymbol{X}_k^p)^T \Sigma_T^{-1}(\boldsymbol{X}_k - \boldsymbol{X}_k^p)}. \tag{17}$$

The lidar points, $\mathcal{Q}_k$, are stored in a 3D KD-tree [17] for fast indexing. Then, for each particle, $\boldsymbol{X}_k^p$, a set of its surrounding points are selected from $\mathcal{Q}_k$, denoted as $\mathcal{S}_k$, such that $p(\boldsymbol{X}_k|\boldsymbol{X}_k^p)$ is larger than a threshold (here we use 1%). Let z_k^p be the observation of particle $\boldsymbol{X}_k^p$; then the conditional probability of z_k^p given $\boldsymbol{X}_k^p$, $P(z_k^p|\boldsymbol{X}_k^p)$, can be computed as $\prod_{\boldsymbol{X}_k \in \mathcal{S}_k} p(\boldsymbol{X}_k|\boldsymbol{X}_k^p)$. Here, since $\mathcal{S}_k$ contains different number of points for each particle, an additional term, $(p_0)^m$, is multiplied by the conditional probability, $p(\boldsymbol{X}_k|\boldsymbol{X}_k^p)$, for normalization. Here, m is the difference between the number of points in $\mathcal{S}_k$ and the maximum number of surrounding points found for each particle, and p_0 is a scale factor ($p_0 = 1\%$). Intuitively, by multiplying by $(p_0)^m$, we equally add in m points to $\mathcal{S}_k$ with p_0 probability of being on the tree trunk. This is also the minimum probability for a point to be selected into $\mathcal{S}_k$. Correspondingly, the importance weight for $\boldsymbol{X}_k^p$ is proportional to its conditional probability, $P(z_k^p|\boldsymbol{X}_k^p)$,

$$w \propto p(z_k^p|\boldsymbol{X}_k^p) \propto (p_0)^m \prod_{\boldsymbol{X}_k \in \mathcal{S}_k} e^{-\frac{1}{2}(\boldsymbol{X}_k - \boldsymbol{X}_k^p)^T \Sigma_T^{-1}(\boldsymbol{X}_k - \boldsymbol{X}_k^p)}. \tag{18}$$

A total number of 500 particles are used on each side. The particles are resampled at each frame. While most of the particles are concentrated on the closest tree trunk, we also want to keep an eye on the surrounding region such that as the vehicle drives, the closest tree trunk switches from one tree to another. Here, we generate 5% of the particles based on a 3D Gaussian distribution as illustrated by the gray colored region in Fig. 7(b). The center of the distribution is at the intersection of the fitted line, $\boldsymbol{L}_l$, and the $y-z$ plane of the vehicle coordinate system $\{V\}$. The long axis is parallel to $\boldsymbol{L}_l$, and the two short axes point in the z-direction and the direction perpendicular to $\boldsymbol{L}_l$, respectively. The remaining particles are resampled based on the importance weights computed in (18), using low variance resampling [4]. To select the closest tree trunk, a checking mechanism monitors potential tree trunks that are closer to the vehicle. The low variance resampling draws new particles from those associated with the closest tree trunk. Then, the detected tree trunk, $\boldsymbol{X}_l$, is the weighted average of the same set of particles used for the resampling.

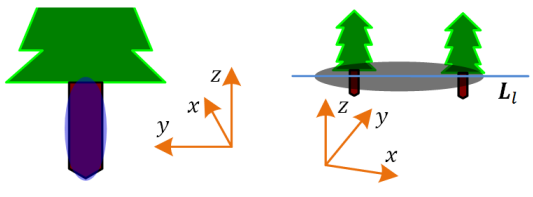
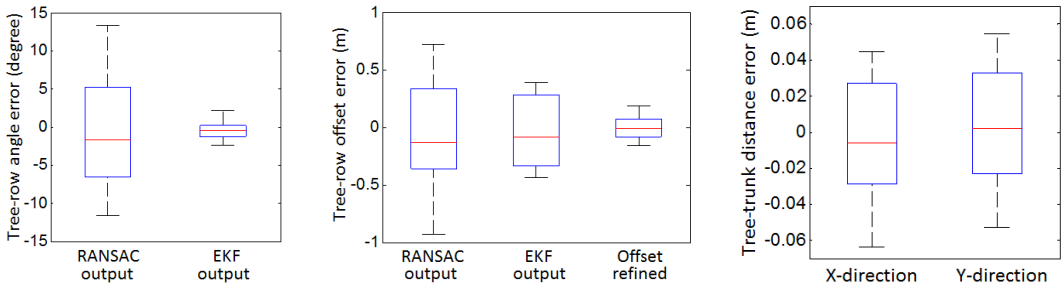
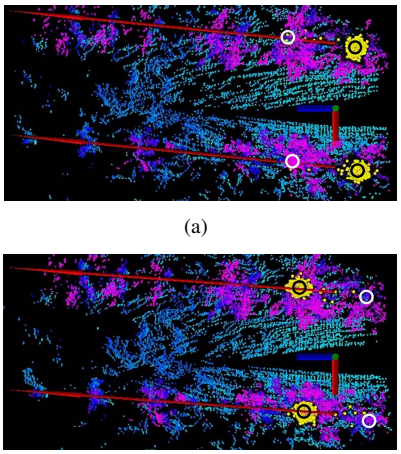


Fig. 7. (a) Distribution of the points on a tree trunk. The points are assumed to follow a Gaussian distribution as illustrated by the blue colored region. (b) Resampling region. In each resampling, 5% of the particles are drawn from a Gaussian distribution as illustrated by the gray colored region.



(a) (b) (c)

Fig. 9. Tree row and tree trunk detection errors using logged data. (a-b) show the angle and lateral offset errors for the detected tree rows. The errors are compared for the output of the RANSAC line-fitting algorithm in Section V-B, the lines filtered by the EKF in Section V-C, and the ones with the lateral offsets refined in Section V-D. (c) shows the distance errors for the detected tree trunks, in the forward (x-) and lateral ($y-$) directions. The red, blue, and black lines represent the median, the 75% range, and the 100% range of the errors, respectively.



(a)

(b)

Fig. 8. An example of detected tree rows and tree trunks using logged data. The red colored lines represent the tree rows and the yellow colored dots illustrate the particles surrounding the closest tree trunk on each side. The black and white colored circles are tree trunks labeled by hand, the black colored circles indicate the closest ones to the vehicle. As the vehicle drives, the particles switch from one tree to the next. The lidar points are is color coded by elevation, in the top-down order of red, blue, and green.

VI. Experiments

To validate the 3D perception algorithm proposed in this paper, we collect range data with the sensor in Fig. 4 in various commercial orchards. These are environments characterized by canopies with large volumes, branches sticking out into the driving row, pipes, cinder blocks, and other small objects that a 2D planar laser is not able to or has great difficulty to perceive. Additionally, we conducted online row following experiments based on the tree row and tree trunk information resulted from processing of the lidar data in real time. These are described in this section.

A. Off-line Tests with Orchard Data

Off-line tests were conducted with data collected over four hours in three commercial orchards in South Korea. The lidar sensor was mounted on a manually-driven vehicle, moving at about 0.4m/s. An example of the results obtained is shown in Fig. 8. The red lines represent the tree rows; the yellow dots are the particles surrounding the tree trunk closest to the vehicle on each side. The black and white circles are manually-labeled tree trunks for ground truth purposes; the black ones represent the trunks closest to the vehicle. From Fig. 8(a) to Fig. 8(b), as the vehicle drives, the particles switch from one tree to the next on the same row. Correspondingly, the detected tree trunks switch as indicated by the black circles.

Fig. 9 presents the accuracy for the tree row and trunk detection. Fig. 9(a-b) show the angle and lateral offset errors for the detected tree rows, respectively. The errors are compared with the output of the RANSAC line-fitting algorithm in Section V-B, the lines filtered by the EKF in Section V-C, and the ones with the lateral offsets refined in Section V-D. Since the offset refinement step does not change the line orientation, only the first two terms are compared in Fig. 9(a). One can see that the errors are reduced significantly after each step of processing. Fig. 9(c) shows the distance errors for the detected tree trunks in the forward and lateral directions. Most of the errors are within a 3cm range around the true value.

B. Closed Control-loop Row Following

To show that the proposed 3D perception system provides sufficient accuracy for row following, we implemented it onboard an autonomous orchard vehicle (Fig. 10(a)). The vehicle is based on a Toro Workman MDE chassis retrofitted as a drive-by-wire platform. It is equipped with steering and driving wheel encoders that provide odometry measurements. A proportional-integral controller is used in the low-level steering and speed control loops. Trajectory control is im-

(a) (b)

Fig. 10. Autonomous vehicle and orchard used to validate the 3D perception algorithm. (a) Drive-by-wire orchard vehicle based on a Toro Workman MDE chassis, equipped with steering and driving wheel encoders for odometry measurements. The 3D lidar sensor is mounted on the front. The vehicle is also equipped with an Applanix Pos-LV positioning system for ground truth acquisition. (b) Tree rows used in the row following tests.

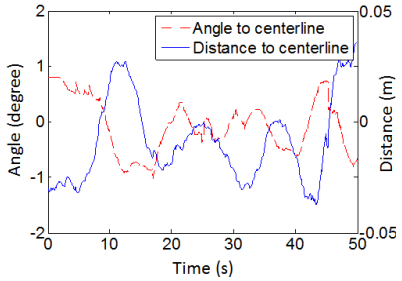


Fig. 11. An example of row following while the vehicle drives along the row centerline. The red and blue curves represent the angle and lateral distance between the vehicle and the row centerline. The position error is within 4cm in a 350cm wide row, and the angular error is within 1 degree.

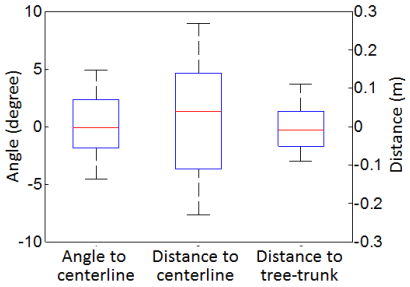


Fig. 12. Row following errors. The left and middle columns show the angular and lateral distance errors while the vehicle follows the row centerline, using the detected tree rows. The right column shows the lateral distance error from the vehicle to one side of the trees, while the vehicle keeps a constant distance to the trees on that side using the detected tree trunks. The red, blue, and black colors have the same meaning as in Fig. 8.

plemented by the vehicle tracking a look-ahead point 1.5m ahead of it, a technique known as pure pursuit [18]. The 3D lidar sensor is mounted on the front, and a laptop computer on the vehicle runs the perception and control algorithms. The vehicle is also equipped with a high-accuracy Applanix Pos-LV positioning system for ground truth purposes.

The experiment site is in a commercial orchard in Pennsylvania (Fig. 10(b)) composed of four rows, each one over 200m long. We first tested tree row detection by driving the vehicle along the row centerline, which is the mid-line between the two detected tree rows. The vehicle follows each row three times at a speed of 0.8m/s. Fig. 11 shows a typical result. The red and blue curves represent the angle and lateral distance between the vehicle and the row centerline. The position error is within 4cm in a 350cm wide row, and the angular error is within 1 degree. Fig. 12 (left and center plots) show the angular and distance errors over a total of twelve traversals. The maximum angular error is 5 degrees, with 50% of the errors within 2.5 degrees; likewise, the maximum distance error is about 27cm in a 350cm wide row, with 50% of the errors within 14cm.

Next, we tested the tree trunk detection by commanding the vehicle to drive with a constant distance to one side of the trees. The vehicle follows each row twice, first following one side and then the other, at a speed of 0.4m/s. Fig.13 shows an example of the detected tree trunks in one test. The red curve shows the distance of the closest tree trunks to the vehicle in the forward direction, and the blue curve

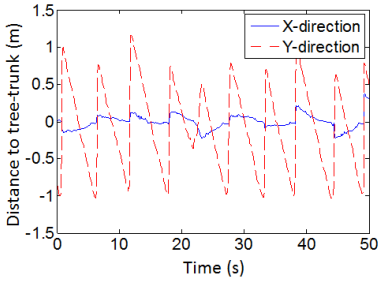


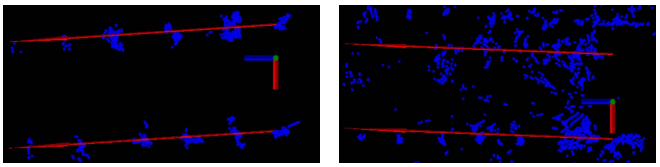
Fig. 13. An example of row following while the vehicle keeps a constant distance to one side of the trees. The red curve is the distance from the tree trunks to the vehicle in the forward direction, and the blue curve shows the distance in the lateral direction. As the vehicle drives, the closest tree trunk switches from one tree to the next one, causing the instantaneous jumps on both curves. The blue curve shows that, at every transition from one trunk to the next, the distance error decreases asymptotically to zero.

is the distance in the lateral direction. As the vehicle drives, the closest tree trunk switches from one tree to the next one, causing the instantaneous jumps on both curves. The blue curve shows that, at every transition from one trunk to the next, the distance error decreases asymptotically to zero. The lateral distance error from the vehicle to the tree trunks is measured with a tape ruler as the vehicle passes by each tree. The overall error over eight traversals is shown in Fig. 12 (right). The maximum error is 11cm, with 50% of the errors within 5cm.

C. Comparison with Planar Lidar

When the tree rows are well structured to form the equivalent of straight walls, a planar lidar is sufficient for row following. To illustrate this case, we use data logged in a high-density orchard in Pennsylvania. The vehicle is driven at 0.5m/s for two hours. We compare the results with the data logged in three orchards in South Korea, where the trees are more irregular. Since the returns from a planar lidar are on a horizontal plane, the offset refinement step (Section V-D) is unnecessary. Fig. 14 shows an example of the detected tree rows when a planar lidar is used in regular and irregular rows. The data in Fig. 14(b) is perceived by the 3D lidar sensor, and extracted from a horizontal plane to simulate the case where a planar lidar is used.

Fig. 15 shows the maximum absolute errors in each row for tree row detection. The errors are compared in three scenarios: a planar lidar in regular rows, a planar lidar in irregular rows, and the 3D lidar in irregular rows (the proposed method). Since we require continuous estimation of the tree rows during the row following, these are the maximum errors



(a) (b)

Fig. 14. An example of tree row detection using a planar lidar in regular (a) and irregular (b) rows. The red lines are the detected tree rows. The blue points are the lidar returns on a horizontal plane.

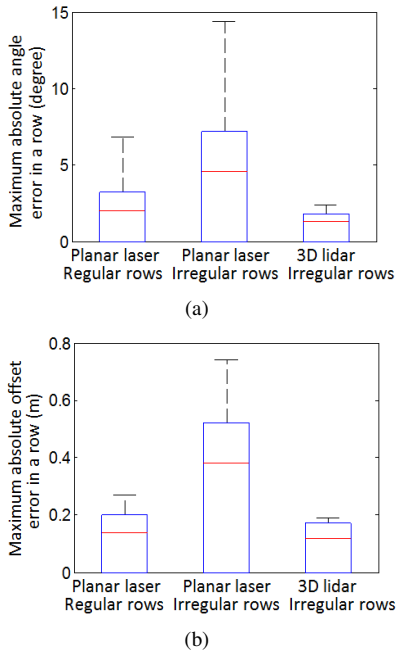


Fig. 15. Comparison of the maximum absolute errors in each row during the tree row detection. These are the maximum errors that the vehicle has to deal with to successfully follow a row. (a) and (b) show angular and offset errors for three cases: planar lidar in regular rows, planar lidar in irregular rows, and 3D lidar in irregular rows. The red, blue, and black colors have the same meaning as Fig. 8. The planar lidar's accuracy is insufficient for row following in irregular environments, a shortcoming that is properly solved by the 3D lidar and the perception algorithm described here.

that the vehicle has to deal with to successfully follow a row. In the middle column, the lidar returns are extracted from multiple horizontal planes with different heights above the ground. Then, the horizontal plane whose associated errors are the smallest is selected. The results in Fig. 15 indicate that in irregular rows, a planar lidar is insufficient while a 3D lidar sensor becomes necessary.

VII. Conclusion

Two-dimensional lidar sensing has been proven effective for autonomous row following in modern orchards, where the trees form an almost perfect "fruit wall" that facilitates perception. Most fruit production environments, however, have volumous canopies with branches sticking into the row and objects such as pipes and rocks that make 2D sensing infeasible for row following. Here we propose a novel 3D perception method based on a 3D lidar to detect the tree rows and trunks in uneven, irregular environments. The method is validated with real orchard data and proven to provide much better row information than a planar, 2D lidar in the same environment. We also preliminarily demonstrate that the row information can be used to close the vehicle guidance and control loop at an orchard experimental site.

Future work will focus on further validating the perception method in commercial orchards with uneven canopies for the purposes of refining the underlying algorithms. The ultimate goal is to build a low-cost perception and navigation package that can be incorporated into many different drive-by-wire farm vehicles to automate fruit production operations.

Acknowledgement

This work is supported partly by the R&D program of the Korea Ministry of Knowledge and Economy (MKE) and the Korea Institute for Advancement of Technology (KIAT). (Project: 3D Perception and Robot Navigation Technology for Unstructured Environments, M0000451).

References

[1] M. Bergerman, S. Singh, and B. Hamner, "Results with autonomous vehicles operating in specialty crops," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, May 2012.

[2] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 12, pp. 119–152, 1994.

[3] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, The MIT Press, 2005.

[5] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transporation Systems*, vol. 9, no. 1, 2008.

[6] H. Kong, J. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Transactions on Image Processing*, vol. 19, no. 8, 2010.

[7] A. Wimmer, T. Weiss, F. Flogel, and K. Dietmayer, "Automatic detection and classification of safety barriers in road construction sites using a laser scanner," in *IEEE Intelligent Vehicles Symposium*, Xian, China, July 2009.

[8] M. Tsogas, N. Floudas, P. Lytrivis, and et al., "Combined lane and road attributes extraction by fusing data from digital map, laser scanner and camera," *Information Fusing, Special Issue on Intelligent Transportation Systems*, vol. 12, pp. 28–36, 2011.

[9] S. Zhou and K. Iagnemma, "Self-supervised learning method for unstructured road detection using fuzzy support vector machines," in *IEEE International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct 2008.

[10] S. Hiremath, F. Evert, G. Heijden, and et al., "Image-based particle filter for robot navigation in a maize field," in *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal, Oct. 2012.

[11] T. Bakker, H. Wouters, K. Asselt, J. Bontsemab, L. Tangc, J. Mullerd, and G. Stratena, "A vision-based row detection system for sugar beet," *Computers and Electronics in Agriculture*, vol. 60, pp. 87–95, 2008.

[12] B. Astrand and A. Baerveldt, "A vision-based row-following system for agricultural field machinery," *Mechatronics*, vol. 15, no. 2, pp. 251–269, 2005.

[13] T. Satow, K. Matsuda, S. Ming, and et al., "Development of laser crop row sensor for automatic guidance system of implements," in *Conference on Automation Technology for Off-road Equipment*, Kyoto, Japan, Oct. 2004.

[14] P. Biber, U. Weiss, M. Dorna, and A. Albert, "Navigation system of the autonomous agricultural robot Bonirob," in *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal, Oct. 2012.

[15] S. Morrehead, B. Gilmore, C. Dima, and C. Wellington, "Automating orchards: A system of autonomous tractors for orchard maintenance," in *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal, Oct. 2012.

[16] T. Ryo, N. Noboru, and M. Akira, "Automatic guidance with a laser scanner for a robot tractor in an orchard," in *Conference on Automation Technology for Off-road Equipment*, Kyoto, Japan, Oct. 2004.

[17] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computation Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008.

[18] S. Roth and P. Batavia, "Evaluating path tracker performance for outdoor mobile robots," in *Automation Technology for Off-Road Equipment*, July 2002.