2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Monocular Visual Navigation of an Autonomous Vehicle in Natural Scene Corridor-like Environments

Ji Zhang, George Kantor, Marcel Bergerman, and Sanjiv Singh

***Abstract*— We present a monocular visual navigation methodology for autonomous orchard vehicles. Modern orchards are usually planted with straight and parallel tree rows that form a corridor-like environment. Our task consists of driving a vehicle autonomously along the tree rows. The original contributions of this paper are: 1) a method to recover vehicle rotation independently of translation by modeling the vehicle as a car-like robot driving on a 3D ground surface–the rotation is estimated from monocular images while the translation is measured by a wheel encoder; and 2) a method to fit the 3D points corresponding to the trees into straight lines via an optimization algorithm that minimizes the error variance on the robot lookahead point. Additionally, we use a simple vanishing point detection approach to find the ends of the tree rows. The vanishing point detection is integrated into the system via an extended Kalman filter. The methodology's robustness to environmental changes is validated in more than fifty experiments in research and commercial orchards, six of which are presented and discussed in detail.**

I. Introduction

This paper focuses on autonomous navigation for orchard applications. As shown in Fig. 1, modern orchards are planted with straight and parallel tree rows. The tree rows are often straight over hundreds of meters, and the space between two tree rows forms a corridor-like environment. The task is to drive a robot autonomously along the tree rows, which requires an approach to continuously estimate the distance and orientation of the tree rows with respect to the robot. Due to different tree types and lighting conditions, the approach has to handle considerable environment variations in both shape and appearance. To keep the system simple and relatively low-cost, a monocular camera is used.

We reconstruct the local structure of the tree rows using image frames and wheel encoder readings. The Ackermann steering model [1] is adopted as the problem constraint. While existing work in visual navigation that uses the robot steering model assumes a flat ground plane, we model the robot as a car-like vehicle driving on a 3D ground surface. By using the Ackermann steering model, we are able to recover the robot orientation separately from the translation. The orientation is estimated with the Singular Value Decomposition (SVD) method [2]. To deal with the scale ambiguity associated with a monocular camera, the translation is measured by wheel encoder reading. The 3D points on the tree rows are reconstructed from the robot motion, and then fitted into straight lines. The line fitting employs an optimization applied to a RANdom SAmple Consensus (RANSAC) [3] algorithm, which minimizes the error variance on the robot lookahead point.

This work is supported by the USDA SCRI program under award number 2008-51180-04876.

J. Zhang, G. Kantor, M. Bergerman, and S. Singh are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213. Email: zhangji@andrew.cmu.edu, kantor@cmu.edu, marcel@cmu.edu, and ssingh@cmu.edu.

(a) (b)

Fig. 1. (a) The robot. A monocular camera is attached in front of the robot as highlighted by the yellow box, a closed view of the camera is shown in the small thumbnail at the right-top corner. (b) The robot drives autonomously along the tree rows. In this example, a liftable platform is attached on the back of the robot, the platform carries two workers.

The proposed tree row reconstruction gives a noisy estimation on the tree row orientations. To obtain an accurate estimation on the orientation, a simple vanishing point detection approach is constructed that finds the tree rows' ends. The vanishing point detection uses the vertical edges on the tree trunks and branches, and finds the point in the image frame where the tree rows vanish. To make the approach robust to lighting condition changes, the edge points are passed through a color filter in $c_1c_2c_3$ space [4]. Then, the two parts of our approach are integrated in an Extended Kalman Filter (EKF) [5].

We conduct experiments with six types of trees, at different orientations and times of day. For each test, the tree rows are over 100 m long. We present field results with the robot autonomously driving over 1.7 km. The results show that our approach is robust to environmental changes, with the crosstrack error with respect to the road centerline at the 10 cm level.

II. Related Work

Research on visual navigation [6] has progressed significantly over the recent decade. Successful applications have been implemented on mobile robots [7], [8], aerial robots [9], [10], underwater robots [11], and even mobile devices [12]. For agriculture applications, vision-based guidance has received large attention [13]–[16]. Most of this work focuses on short crops. Camera images from top-down view are used for detecting the vegetation from ground. Considering that the trees in the orchards are tall and form a corridor-like environment, top-view images are unavailable.

Visual navigation in corridor-like environments is common for indoor robots. Landmarks or objects with distinct appearance such as doors and unique colored floor planes are often employed [17]. Other work utilizes artificial structures such as straight and parallel line segments on door and wall boundaries [18]. Since our robot navigates in natural scene environments, artificial features are unavailable. Instead, optical flow [19], [20] and Structure From Motion (SFM) [21]–[23] are two popular options. Considering the high noise nature of optical flow methods, this paper adopts SFM.

Our approach adopts the Ackermann steering model as the problem constraint. Nourani-Vatani et al. have used the Ackermann steering model for their practical visual odometry [24]. Scaramuzza et al. have used the steering model along with steering encoder readings as constraints for their SFM and visual odometry [25], [26]. The approach has a low computational cost and shows significantly improved accuracy compared to unconstrained cases. However, the above work assumes that the robot motion is limited to a planar ground surface. The assumption does not hold in our problem since the robot drives on off-road irregular terrains. In this paper, the motion model is revised such that the robot drives on a 3D ground surface.

Vanishing point detection is another closely related area to this paper. Existing work can be found in the road detection literature [27]. While much work focuses on structured roads [28], [29], where the road surface is homogeneously colored and landmarks such as traffic lines are often available, studies on unstructured roads are also explored [30]–[32]. Based on the fact that an orchard does not have explicit and constant structures on the ground surface, this paper simply utilizes the vertical structure of tree trunks and branches.

This paper is based on our previous work where an autonomous navigation system is developed using a fixed planar laser scanner as the environmental sensor [33]. Since the measurements of a fixed laser scanner can not represent 3D environment, this paper uses a monocular camera to fulfil the task. The tree row reconstruction relies on robot motion, while the vanishing point detection uses a single image. Combination of the two parts allows the robot to operate both in driving mode and starting from stationary.

III. Problem Definition

The problem is to detect the tree rows in orchards using a monocular vision system, and drive the robot autonomously along the tree rows. We start with the assumptions that guide our work.

A. Assumptions

1) We assume that the robot follows the Ackermann steering model [1]. As shown in Fig. 1(a), the robot is built with straight rear wheels and steering front wheels, falling in the category of nonholonomic car-like robots. The robot motion is limited to the direction perpendicular to the axles, translation parallel to the axles is not allowed.
2) We assume that the driving speed is known and measured by a wheel encoder. This assumption helps us deal with the scale ambiguity associated with the monocular camera. We also assume that the yaw, pitch, and roll angles of the robot are unknown, and will be estimated from the image frames. Since the robot drives on a 3D ground surface, the orientation cannot simply be measured from encoder readings.
3) We assume that the camera follows the pinhole camera model [2]. The intrinsic and extrinsic parameters are known from pre-calibration, the lens distortion is also removed.

B. Notation and Coordinate Systems

As a convention in this paper, we use right uppercase superscripts to indicate the coordinate system, and right subscript $k \in Z^+$ to indicate the image frames.

- The camera coordinate system $\{C\}$ is a 3D coordinate system. As shown in Fig. 2, the origin is at the camera optical center with the z-axis coinciding with the camera principal axis. The $x-y$ plane is parallel to the camera image sensor with the x-axis parallel to the horizontal direction of the image pointing to the left. A point in $\{C_k\}$ is denoted as $\boldsymbol{X}_k^C = [x_k^C,\ y_k^C,\ z_k^C]^T$.
- The vehicle coordinate system $\{V\}$ is a 3D coordinate system. As shown in Fig. 2, the origin coincides with the origin of $\{C\}$, the x-axis is parallel to the robot axles pointing to the left-hand side, the y-axis points upward, and the z-axis points forward. Although the Ackermann steering model requires that the y-axis passes through the mid-point of the robot rear axle, the requirement is relaxed due to small steering angles. The same procedure can be found in [26]. A point in $\{V_k\}$ is denoted as $\boldsymbol{X}_k^V = [x_k^V,\ y_k^V,\ z_k^V]^T$.
- The image coordinate system $\{I\}$ is a 2D coordinate system with its origin at the right bottom corner of the image. The u- and v-axes in $\{I\}$ point to the same direction as the x- and y-axes in $\{C\}$. A point in $\{I_k\}$ is denoted as $\boldsymbol{X}_k^I = [u_k,\ v_k,\ 1]^T$.

C. Problem Description

Recall that the problem addressed in this paper is to drive the robot between two parallel tree rows. Let l_k^l and l_k^r be two parallel straight lines on the $x-z$ plane in $\{V_k\}$, where l_k^l and l_k^r represent the tree rows on the left and right sides of

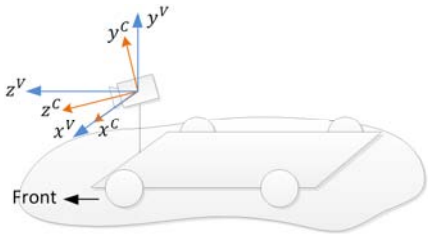


Fig. 2. Illustration of the vehicle coordinate system $\{V\}$ and the camera coordinate system $\{C\}$.

the robot. Let l_k^c be the centerline of l_k^l and l_k^r, l_k^c represents the road center that we want the robot to follow. Define $\boldsymbol{L}_k^V$ as a 2×1 vector containing the coefficients of l_k^c; the line function of l_k^c can be expressed as

$$x + L_k^V(1)y + L_k^V(2) = 0, \tag{1}$$

where $L_k^V(h)$, $h = 1, 2$, is the h-th entry of $\boldsymbol{L}_k^V$. In this paper, we want to compute $\boldsymbol{L}_k^V$ using the image frames. The task can be described as follows:

Problem 1: Given a set of image frames $k \in Z^+$, compute $\boldsymbol{L}_k^V$ for each frame k, and use $\boldsymbol{L}_k^V$ to drive the robot along the tree rows.

IV. Tree Row Reconstruction

A. Estimate Robot Orientation

We start by estimating the robot orientation. It shows that by adopting the Ackermann steering model, the orientation can be recovered independently of the translation. Because of the steering model assumed, the translation is limited to the z-axis direction in $\{V\}$. Let Δz_k be the translation between frames $k-1$ and k in $\{V_{k-1}\}$ coordinate system. Furthermore, let Δp_k, Δt_k, and Δr_k be the robot yaw, pitch, and roll rotational angles between frames $k-1$ and k, respectively, around the y-, x-, and z- axes of $\{V_{k-1}\}$ following the right hand rule. In this section, we will estimate Δp_k, Δt_k, and Δr_k using two consecutive frames.

From the pin-hole camera model, we have the following relationship between $\{I_k\}$ and $\{C_k\}$ [2]:

$$\varsigma_k \boldsymbol{X}_k^I = \mathbf{K}\boldsymbol{X}_k^C, \tag{2}$$

where ς_k is a scale factor, and $\mathbf{K}$ is the camera intrinsic parameter matrix, known from the pre-calibration.

The relationship between $\{C_k\}$ and $\{V_k\}$ coordinate systems can be expressed as [2]

$$\boldsymbol{X}_k^C = \mathbf{R}_z(r)\mathbf{R}_x(t)\mathbf{R}_y(p)\boldsymbol{X}_k^V, \tag{3}$$

where $\mathbf{R}_x(\cdot)$, $\mathbf{R}_y(\cdot)$, and $\mathbf{R}_z(\cdot)$ are rotational matrices around x-, y-, and z- axes, respectively, and p, t, and r are the corresponding rotational angles from $\{V_k\}$ to $\{C_k\}$. Note that p, t, and r are the camera extrinsic parameters, known from the pre-calibration.

Substituting (3) into (2) for frames $k-1$ and k, we have

$$\varsigma_{k-1}\boldsymbol{X}_{k-1}^I = \mathbf{T}\boldsymbol{X}_{k-1}^V, \ \varsigma_k \boldsymbol{X}_k^I = \mathbf{T}\boldsymbol{X}_k^V, \tag{4}$$

where

$$\mathbf{T} = \mathbf{K}\mathbf{R}_z(r)\mathbf{R}_x(t)\mathbf{R}_y(p). \tag{5}$$

Using the robot motion between frames $k-1$ and k, a relationship can be established between $\{V_{k-1}\}$ and $\{V_k\}$ [2]:

$$\boldsymbol{X}_k^V = \mathbf{R}_z(\Delta r_k)\mathbf{R}_x(\Delta t_k)\mathbf{R}_y(\Delta p_k)\boldsymbol{X}_{k-1}^V + [0,\ 0,\ \Delta z_k]^T. \tag{6}$$

Substituting (4) into (6), we can eliminate Δz_k. Then, since in this paper a relatively high camera frame rate (10 Hz) and low driving speed (1 m/s) is used, Δp_k, Δt_k, and Δr_k are small angles in practice. We propose linearization to obtain the following equation:

$$a\Delta p_k + b\Delta t_k + c\Delta r_k + d = 0, \tag{7}$$

where

$$a = N_k(2),\ b = -N_k(1), \tag{8}$$

$$c = N_{k-1}(1)N_k(1) + N_{k-1}(2)N_k(2), \tag{9}$$

$$d = N_{k-1}(1)N_k(2) + N_{k-1}(2)N_k(1). \tag{10}$$

Here, $N_{k-1}(h)$ and $N_k(h)$, $h = 1, 2$ are the h-th entries of $\boldsymbol{N}_{k-1}$ and $\boldsymbol{N}_k$, respectively, where

$$\boldsymbol{N}_{k-1} = \mathbf{T}^{-1}\boldsymbol{X}_{k-1}^I,\ \boldsymbol{N}_k = \mathbf{T}^{-1}\boldsymbol{X}_k^I, \tag{11}$$

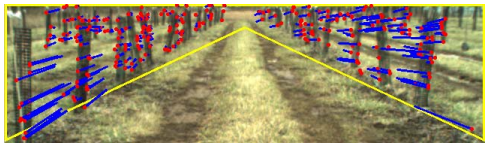
and $\mathbf{T}$ is a invertible matrix based on (5).

Eq. (7) describes a linear relationship of Δp_k, Δt_k, and Δr_k that is independent of Δz_k, which indicates that Δp_k, Δt_k, and Δr_k can be recovered regardless of Δz_k using three or more feature points from the image frames. In this paper, the SVD method is used to solve the function.

B. Feature Tracking and 3D Point Reconstruction

To track features over image frames, we define the region of interest. As shown in Fig. 3(a), the pixels in the yellow colored polygon are used for the tree row reconstruction. A 5×5 pixel Sobel filter is applied to the region of interest to highlight the vertical edges. The edge points with values larger than a constant threshold are selected, as shown in Fig. 3(b). After that, a set of feature points are randomly selected from the edge points and tracked over frames with the Lucas Kanade (LK) method [34].

At each frame, a certain number of feature points (20 points) are generated, and tracked over a fixed number of frames (15 frames). Therefore, at each frame we have a total of $20 \times 15 = 300$ features, while 20 of the features are tracked for m, $m = 1, 2, ..., 15$, frames each. The SVD method is applied to a RANSAC algorithm, which selects a subset of inliers from the overall tracked features to compute



(a)

(b)

Fig. 3. (a) An image frame with tracked feature points. The yellow colored polygon illustrates the region of interest. The red colored dots represent the features, and the blue colored line segments illustrate the feature displacements over tracked frames. (b) Vertical edges extracted in the region of interest.

Δp_k, Δt_k, and Δr_k. Note that the features are tracked over different number of frames, we rewrite (7) as follows:

$$a\sum_{i=0}^{m-1}\Delta p_{k-i}+b\sum_{i=0}^{m-1}\Delta t_{k-i}+c\sum_{i=0}^{m-1}\Delta r_{k-i}+d=0, \quad (12)$$

where Δp_{k-i}, Δt_{k-i}, and Δr_{k-i} are the robot rotational angles between frames $k-i-1$ and $k-i$, respectively. Eq. (12) can be used for features tracked for m frames. Here, note that by frame k, Δp_{k-i}, Δt_{k-i}, and Δr_{k-i}, $i=1,2,...,m-1$ are already computed. The unknowns are Δp_k, Δt_k, and Δr_k. Correspondingly, $\boldsymbol{N}_{k-1}$ in (9), (10), and (11) is replaced by $\boldsymbol{N}_{k-m}$, and $\boldsymbol{X}^I_{k-1}$ in (11) is replaced by $\boldsymbol{X}^I_{k-m}$, which represents the feature at frame $k-m$.

With the rotational angles computed, the 3D points can be reconstructed in $\{V_k\}$. Using a simple geometrical relationship, $\boldsymbol{X}^V_k$ can be computed from the following equation:

$$\boldsymbol{X}^V_k = z^V_k\left[N_k(1)/N_k(3),\ N_k(2)/N_k(3),\ 1\right]^T, \quad (13)$$

where $N_k(3)$ is a non-zero term based on (11), and

$$z^V_k=\frac{\Delta z^k_{k-m}N'_{k-m}(1)N_k(3)}{N'_{k-m}(3)N_k(1)-N'_{k-m}(1)N_k(3)}. \quad (14)$$

Here, Δz^k_{k-m} is the driving distance of the robot between frames $k-m$ and k, which is known from the encoder measurements, $N'_{k-m}(h)$ and $N_k(h)$, $h=1,2,3$ are the h-th entries of $\boldsymbol{N}'_{k-m}$ and $\boldsymbol{N}_k$, respectively, where

$$\begin{aligned}\boldsymbol{N}'_{k-m}=&\mathbf{R}_z(\sum_{i=0}^{m-1}\Delta r_{k-i})\mathbf{R}_x(\sum_{i=0}^{m-1}\Delta t_{k-i})\\ &\mathbf{R}_y(\sum_{i=0}^{m-1}\Delta p_{k-i})\mathbf{T}^{-1}\boldsymbol{X}^I_{k-m}. \quad (15)\end{aligned}$$

Finally, we compute the covariance of $\boldsymbol{X}^V_k$, which will be useful in the next section. We start with the image reprojection error. Let $\hat{u}_k$ and $\hat{v}_k$ be the reprojected u- and v-coordinates in $\{I_k\}$; $\hat{u}_k$ and $\hat{v}_k$ can be computed by treating u_k and v_k as the unknowns in (12), respectively, and solving the linear equation. Let $e^u_k=|u_k-\hat{u}_k|$ and $e^v_k=|v_k-\hat{v}_k|$ represent the image reprojection error at frame k. Define Σ^I_k as a 2×2 matrix representing the covariance of $\boldsymbol{X}^I_k$; Σ^I_k can be expressed as

$$\Sigma^I_k=\mathrm{diag}\left[(e^u_k)^2,(e^v_k)^2\right]. \quad (16)$$

Similarly, let Σ^I_{k-m} be the matrix containing the covariance of $\boldsymbol{X}^I_{k-m}$, Σ^I_{k-m} can be computed in the similar fashion.

Eq. (13) indicates that $\boldsymbol{X}^V_k$ is a function of $\boldsymbol{X}^I_{k-m}$ and $\boldsymbol{X}^I_k$. Let $\mathbf{J}^{I-V}_{k-m}$ and $\mathbf{J}^{I-V}_k$ be the Jacobian matrices of the function with respect to $\boldsymbol{X}^I_{k-m}$ and $\boldsymbol{X}^I_k$, respectively, $\mathbf{J}^{I-V}_{k-m}=\partial\boldsymbol{X}^V_k/\partial\boldsymbol{X}^I_{k-m}$ and $\mathbf{J}^{I-V}_k=\partial\boldsymbol{X}^V_k/\partial\boldsymbol{X}^I_k$. Define Σ^V_k as the covariance matrix of $\boldsymbol{X}^V_k$, Σ^V_k can be computed as [2]

$$\Sigma^V_k=\mathbf{J}^{I-V}_{k-m}\Sigma^I_{k-m}(\mathbf{J}^{I-V}_{k-m})^T+\mathbf{J}^{I-V}_k\Sigma^I_k(\mathbf{J}^{I-V}_k)^T. \quad (17)$$

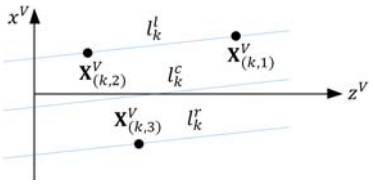


Fig. 4. Illustration of computing straight lines l^l_k, l^r_k, and l^c_k from three reconstructed points $\boldsymbol{X}^V_{(k,1)}$, $\boldsymbol{X}^V_{(k,2)}$, and $\boldsymbol{X}^V_{(k,3)}$.

C. Fit Reconstructed Points into Straight Lines

With the 3D points reconstructed, we fit the points into straight lines. The line fitting employs an optimization that minimizes the error variance of the robot lookahead point. We start with the simplest case. Suppose there are only three reconstructed points, $\boldsymbol{X}^V_{(k,1)}$, $\boldsymbol{X}^V_{(k,2)}$, and $\boldsymbol{X}^V_{(k,3)}$, with the corresponding covariance matrices denoted by $\Sigma^V_{(k,1)}$, $\Sigma^V_{(k,2)}$, and $\Sigma^V_{(k,1)}$, respectively. Without loss of generality, we assume $\boldsymbol{X}^V_{(k,1)}$ and $\boldsymbol{X}^V_{(k,2)}$ are on the robot left side, and $\boldsymbol{X}^V_{(k,3)}$ is on the robot right side, as shown in Fig. 4. Recall that l^l_k, l^r_k, and l^c_k are straight lines representing the left tree row, right tree row, and the road centerline. Using geometry, l^l_k can be computed as

$$l^l_k:\ x+a_kz+b^l_k=0, \quad (18)$$

where

$$a_k=-\frac{x^V_{(k,2)}-x^V_{(k,1)}}{z^V_{(k,2)}-z^V_{(k,1)}},\ b^l_k=\frac{x^V_{(k,2)}z^V_{(k,1)}-x^V_{(k,1)}z^V_{(k,2)}}{z^V_{(k,2)}-z^V_{(k,1)}}. \quad (19)$$

l^r_k is parallel to l^l_k and passes through $\boldsymbol{X}^V_{(k,3)}$, thus we can compute l^r_k as

$$l^r_k:\ x+a_kz+b^r_k=0, \quad (20)$$

where

$$b^r_k=\frac{x^V_{(k,3)}z^V_{(k,1)}-x^V_{(k,1)}z^V_{(k,3)}-x^V_{(k,3)}z^V_{(k,2)}+x^V_{(k,2)}z^V_{(k,3)}}{z^V_{(k,2)}-z^V_{(k,1)}}. \quad (21)$$

The center line of l^l_k and l^r_k, l^c_k, is then obtained as

$$l^c_k:\ x+a_kz+(b^l_k+b^r_k)/2=0. \quad (22)$$

With l^l_k, l^r_k, and l^c_k computed, let us derive the expression of the robot lookahead point. Let z_0 be the lookahead distance (z_0 = 4 m), and let x^c_k be the x-coordinate of the point on l^c_k where $z=z_0$. $[x^c_k,\ 0,\ z_0]^T$ represents a point on the $x-z$ plane in $\{V_k\}$, which is known as the robot lookahead point. From (22), we can derive x^c_k as

$$x^c_k=(b^l_k+b^r_k)/2-a_kz_0. \quad (23)$$

Eq. (23) indicates that x^c_k is a function of $\boldsymbol{X}^V_{(k,h)}$, $h=1,2,3$. Let $\mathbf{J}^V_{(k,h)}$ be the Jacobian matrix of this function with respect to $\boldsymbol{X}^V_{(k,h)}$, $\mathbf{J}^V_{(k,h)}=\partial x^c_k/\partial\boldsymbol{X}^V_{(k,h)}$. Let σ^c_k be the standard deviation of x^c_k. σ^c_k can be obtained as [2]

$$(\sigma^c_k)^2=\sum_{h=0}^{3}\mathbf{J}^V_{(k,h)}\Sigma^V_{(k,h)}(\mathbf{J}^V_{(k,h)})^T. \quad (24)$$

Algorithm 1: Line Fitting

input : A set of $\boldsymbol{X}_k^V$ and the corresponding Σ_k^V
output : $\boldsymbol{L}_k^V$
begin
  for a certain number of iterations **do**
    Randomly select three points from the set of $\boldsymbol{X}_k^V$ where at least one point is on the robot left and right side, compute a_k, b_k^l, b_k^r using (18), (20), (22);
    Compute the Mahalanobis Distance (MD) from each $\boldsymbol{X}_k^V$ to l_k^l or l_k^r, depending on which one is smaller, find a subset of $\boldsymbol{X}_k^V$ as the inliers whose MD is smaller than a threshold;
    for a certain number of iterations **do**
      Randomly and repeatedly select three points without replacement from the inliers of $\boldsymbol{X}_k^V$, where at least one point is on the robot left and right side, compute $a_{(k,j)}$, $b_{(k,j)}^l$, $b_{(k,j)}^r$, $\sigma_{(k,j)}^c$ using (18), (20), (22), (24);
      Compute w_j using (29), compute a_k, b_k^l, b_k^r using (25), and compute $\boldsymbol{L}_k^V$ using (30);
      Compute the mean MD from each $\boldsymbol{X}_k^V$ to l_k^l or l_k^r, depending on which one is smaller;
      if the mean MD is smaller than a threshold **then**
        Return $\boldsymbol{L}_k^V$;
      end
    end
  end
  Return $\boldsymbol{L}_k^V$ with the minimum mean MD found.
end

In this simplest case, l_k^l, l_k^r, and l_k^c compose a line set. In the general case where more than three reconstructed points are available, we repeatedly select three points at a time without replacement to compute a line set. Suppose there are a total of $n \in Z^+$ successful selections; let $l_{(k,j)}^l$, $l_{(k,j)}^r$, and $l_{(k,j)}^c$, $j = 1, 2, ..., n$ be the line set generated in the j-th selection, namely line set j. Accordingly, let $a_{(k,j)}$, $b_{(k,j)}^l$, and $b_{(k,j)}^r$ be the coefficients of line set j. Furthermore, starting from this point, we abuse the meanings of a_k, b_k^l, and b_k^r to let them refer to the terms computed using the overall line sets. In this paper, a_k, b_k^l, and b_k^r are computed as the weighted sums of $a_{(k,j)}$, $b_{(k,j)}^l$, and $b_{(k,j)}^r$, $j = 1, 2, ..., n$:

$$a_k = \sum_{j=0}^{n} w_j a_{(k,j)},\ b_k^l = \sum_{j=0}^{n} w_j b_{(k,j)}^l,\ b_k^r = \sum_{j=0}^{n} w_j b_{(k,j)}^r, \tag{25}$$

where w_j is the weight for line set j, such that

$$\sum_{j=0}^{n} w_j = 1,\ \text{and}\ w_j \geq 0,\ j = 1, 2, ..., n. \tag{26}$$

Similarly, let $x_{(k,j)}^c$ and $\sigma_{(k,j)}^c$ indicate the x_k^c and σ_k^c computed from line set j. We use x_k^c and σ_k^c to refer to the terms computed from the overall line sets. Substituting (25) into (23), we can derive x_k^c and σ_k^c as

$$x_k^c = \sum_{j=0}^{n} w_j x_{(k,j)}^c,\ (\sigma_k^c)^2 = \sum_{j=0}^{n} w_j^2 (\sigma_{(k,j)}^c)^2. \tag{27}$$

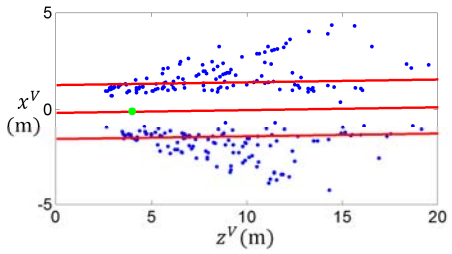


Fig. 5. Reconstructed points and fitted straight lines in the vehicle coordinate system. The robot is at the origin. The blue colored dots are the reconstructed points, the red colored lines in top-to-bottom order are l_l, l_c, and l_r, and the green colored dot represent the lookahead point.

Now, we assign values to the weights in (25) and (27). In this paper, we want to compute w_j such that the error variance of the robot lookahead point, $(\sigma_k^c)^2$, is minimized. Mathematically, the optimization problem is expressed as

Problem 2: Given $\sigma_{(k,j)}^c$, $j = 1, 2, ..., n$, compute

$$\{w_j,\ j = 1, 2, ..., n\} = \arg\min_{w_j} (\sigma_k^c)^2, \tag{28}$$

based on the expression of σ_k^c in (27) and subject to the constraints in (26).

Problem 2 is analytically solvable using the Lagrange multiplier method [35]. Here, we directly give the solution:

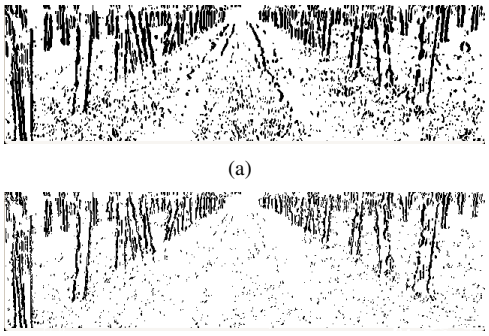
$$w_j = \frac{1/(\sigma_{(k,j)}^c)^2}{\sum_{q=0}^{n} 1/(\sigma_{(k,q)}^c)^2},\ j = 1, 2, ..., n. \tag{29}$$

Note that selecting the reconstructed points without replacement guarantees that the optimization problem is analytically solvable. This way, no two line sets use the same reconstructed point. However, if selection with replacement is used, the problem of minimizing $(\sigma_k^c)^2$ becomes a convex optimization problem that has to be solved numerically [36]. With w_j computed, the RANSAC line fitting algorithm is shown in Algorithm 1. The algorithm evaluates the reconstructed points using the Mahalanobis distance, and selects a subset of the points to compute l_k^l and l_k^r. After that, the coefficients of l_k^c, $\boldsymbol{L}_k^V$, can be obtained as

$$\boldsymbol{L}_k^V = \left[a_k,\ (b_k^l + b_k^r)/2\right]^T. \tag{30}$$

V. Vanishing Point Detection

We extract the vanishing point using a single image frame. The vanishing point represents the tree row end. Intuitively, the approach measures the height of the trees by counting the vertical edges on the tree trunks and branches, and looks for the point where the tree rows vanish. We use the pixels in the entire image area as shown in Fig. 3(a). Similar to Section IV-B, a 5×5 pixel Sobel filter is applied to highlight the vertical edges. The edge points with values larger than a constant threshold are selected, as shown in Fig. 6(a). Then, the selected points are passed through a color filter, whose function is to keep the points on the tree trunks and branches, and cancel those on grass or tree leafs. We first convert the image from RGB space to $c_1c_2c_3$ space to make it robust to shadowing and illumination changes [37], then the color



(a)

(b)

Fig. 6. (a) Vertical edges, and (b) Edge points passed through a color filter. Pixels on tree trunks or branches can pass the color filter, while pixels on grass or tree leafs are canceled.

filter with thresholds on c_1, c_2, and c_3 channels is applied. An example after the color filtering is shown in Fig. 6(b).

We count the number of edge points in the image frame for each column of one pixel width, represented as the blue colored dots in Fig. 7. Then, a low-pass filter is applied to obtain the blue colored curve. Let $f(u_k)$ be the function of that curve. We find the critical points on $f(u_k)$ as illustrated by the red colored dots. Suppose we have a total number of $l \in Z^+$ critical points, then, we find two critical points i and j, $i, j = 1, 2, ..., l$, $j > i$, using an evaluation function:

$$\begin{aligned}&\{i,j\} = \\ &\arg\min_{i,j} f(u_{(k,i)}) - f(u_{(k,i+1)}) + f(u_{(k,j)}) - f(u_{(k,j-1)}) \\ &+ (\sum_{q=1}^{i} f(u_{(k,q)}) + \sum_{q=j}^{l} f(u_{(k,q)}))/(l-j+i+1) \\ &- \sum_{q=i+1}^{j-1} f(u_{(k,q)})/(j-i-1), \end{aligned} \tag{31}$$

where $u_{(k,i)}$ and $u_{(k,j)}$ are the u-coordinates of the two critical points, respectively. In this paper, $[u_{(k,i)},\ u_{(k,j)}]$ is treated as the trustable region where the vanishing point exists, as illustrated by the green colored dots in Fig. 7. In the next step, we find the minimum point on $f(u_k)$ in the trustable region. Let it be u_k^*, where

$$\{u_k^*,\ u_k \in [u_{(k,i)},\ u_{(k,j)}]\} = \arg\min_{u_k^*} f(u_k). \tag{32}$$

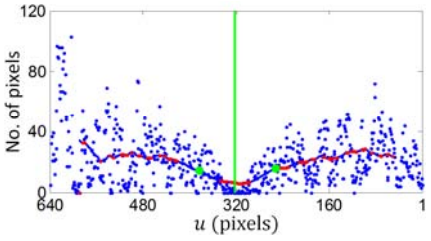


Fig. 7. Vanishing point detection. The blue colored dots represent the number of vertical edges in each column, the blue colored curve is the low-pass $f(u_k)$, the red colored dots are the critical points on $f(u_k)$, the two green colored dots illustrate the trustable region, and the green colored vertical line illustrates the detected vanishing point.

After that, we find the largest neighborhood of u_k^*, denoted as $[u_{(k,l)}^*,\ u_{(k,r)}^*]$, such that

$$\forall u_k \in [u_{(k,l)}^*,\ u_{(k,r)}^*],\ |f(u_k) - f(u_k^*)| \leq \delta, \tag{33}$$

where δ is a threshold. Finally, define u_k^v as the detected vanishing point in an image frame, u_k^v is obtained as the mid-point of the neighborhood region:

$$u_k^v = (u_{(k,l)}^* + u_{(k,r)}^*)/2. \tag{34}$$

VI. Data Integration

We use an EKF to integrate the tree row reconstruction and the vanishing point detection. The EKF takes the tree row reconstruction output for the prediction step and the vanishing point detection output for the update step. The covariance matrices for both steps are measured a priori. Recall that $\boldsymbol{L}_k^V$ containes the coefficients of l_k^c. Let $\boldsymbol{L}_{k|k}^V$ and $\boldsymbol{L}_{k|k-1}^V$ be two 2×1 vectors representing the EKF predicted state estimate and updated state estimate at frame k, defined in the same fashion as $\boldsymbol{L}_k^V$. The EKF prediction step can be expressed as

$$\boldsymbol{L}_{k|k-1}^V = \boldsymbol{L}_k^V. \tag{35}$$

For the EKF update step, recall that u_k^v is the vanishing point detected from the image frames. Let $\hat{u}_k^v$ be the estimated value of u_k^v. Using geometry, the following relationship holds for computing $\hat{u}_k^v$:

$$\hat{u}_k^v = (K(1,1) * L_{k|k-1}^V(1) + K(1,3))/K(3,3), \tag{36}$$

where $K(g,h)$, $g,h = 1,3$, is the (g,h)-th entry of the camera intrinsic parameter matrix $\mathbf{K}$, and $L_{k|k-1}^V(1)$ is the first entry of $\boldsymbol{L}_{k|k-1}^V$. Using the above relationship, the EKF update step can be represented as

$$\boldsymbol{L}_{k|k}^V = \boldsymbol{L}_{k|k-1}^V + \boldsymbol{Z}_k(u_k^v - \hat{u}_k^v), \tag{37}$$

where $\boldsymbol{Z}_k$ is the state update matrix generated by EKF.

VII. Experiments

We conducted experiments on a robot as shown in Fig. 1(a). The robot is based on a Toro MDE eWorkman electric vehicle and is retrofitted to drive autonomously. The robot uses PI controllers for the low level steering and speed control. The speed is set to 1 m/s. An Imagingsource DFK 21BUC03 camera with a frame rate of 10 Hz and image resolution of 640×480 pixels is used for image acquisition. The camera focal length is set at 4 mm. A Panasonic CF-F8 Toughbook computer running Ubuntu 10.04 is used to run the algorithms. There is an algorithm that controls the robot to drive toward the lookahead point. The robot is also equipped with a high accuracy positioning system (Applanix Pos-LV), accurate to better than 10 cm for ground truth purposes.

We have conducted more than 50 tests with different experimental configurations. In each test the robot successfully drives along the tree row from the beginning to the end. In this paper, we select six representative tests from the overall tests for presentation; the other ones are implemented in redundant ways and led to similar results. As shown in

TABLE I

CONFIGURATIONS OF THE SIX REPRESENTATIVE TESTS.

Test No.	Configuration			
	Tree type	Length	Ori.	Time
1	Maple	181 m	92°	12:32 PM
2	Golden Deli.	155 m	112°	1:27 PM
3	Fuji	112 m	20°	2:35 PM
4	Mac	137 m	35°	3:24 PM
5	Gala	105 m	60°	4:37 PM
6	Honey Crisp	162 m	40°	5:35 PM

Table I, the representative tests are selected at different times of day, with different tree types and different orientations of the tree rows. Here, the tree row orientation is represented by the Azimuth angle. For each test, we let the robot drive toward both directions of the tree rows to obtain the maximum lighting condition change. The overall driving distance combining the six tests is over 1.7 km.

We compare three different approaches in the experiment,

1) Tree row Reconstruction with Equal Weights (**TREW**): Another version of the tree row reconstruction, where we use equal weights in (25) instead of the optimized weights. We choose this approach to show the effect of the optimization in the tree row reconstruction.
2) Tree row Reconstruction withOut Vanishing point detection (**TROV**): The tree row reconstruction without integrating the vanishing point detection. We adopt this approach to show the effect of the vanishing point detection in the EKF.
3) Tree row Reconstruction With Vanishing point detection (**TRWV**): The full version of the tree row reconstruction integrated with the vanishing point detection. This is the proposed approach in this paper.

To evaluate the experimental results, we define σ^a and σ^b as the standard deviation of the 1th and 2th entries of the estimated $\boldsymbol{L}_k^V$. Intuitively, σ^a and σ^b represent the orientation errors and the lateral errors for the detected tree rows. Furthermore, let σ^c be the standard deviation of the lookahead point x_k^c. σ^a, σ^b, and σ^c are measured by the Applanix Pos-LV system using combination of the

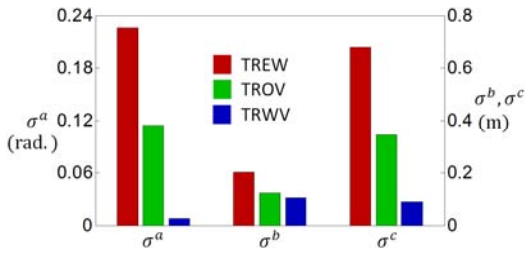


Fig. 8. Comparison of the tree row detection results using combination of the data from the six representative tests. σ^a, σ^b, and σ^c are respectively the standard deviations of the orientation errors, lateral errors, and the lookahead point errors for the detected tree rows. TREW is another version of the tree row reconstruction using equal weights in (25) instead of the optimized weights, TROV is the tree row reconstruction without integrating the vanishing point detection, and TRWV is our proposed method. The ground truth is measured by the Applanix Pos-LV system.

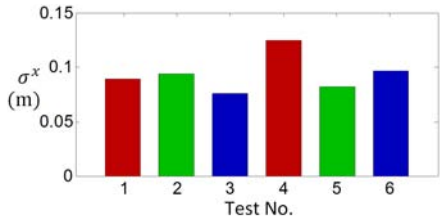


Fig. 9. Comparison of tree row following results for the six representative tests. σ^x is the standard deviation of the robot crosstrack errors with respect to the road centerline.

data from the six representative tests. Fig. 8 shows the tree row detection results. Comparing TREW with TROV, it is obvious that TROV outperforms TREW for all σ^a, σ^b, and σ^c, which indicates that the optimization in Section IV-C works effectively for reducing the tree row reconstruction error. Comparing the results of TROV and TRWV, we can see that the σ^a of TRWV is significantly smaller than TROV, while the σ^b of TRWV is slightly smaller. This indicates that by integrating the vanishing point detection, the orientation errors of the detected tree rows are greatly reduced. As a result, the σ^c of TRWV is much smaller than TROV.

Using TRWV to drive the robot along the tree rows, we measure the crosstrack errors with respect to the road center. Let σ^x be the standard deviation of the distance from the robot to the road centerline. As shown in Fig. 9, σ^x is in the same level as the σ^c of TRWV in Fig. 8 for all the six tests. The σ^x in Test 4 is slightly larger because the ground is relatively irregular and inclined. From the results, we can see that the robot is able to drive along the tree rows with reasonably small crosstrack errors.

VIII. CONCLUSION AND FUTURE WORK

In this paper, a monocular visual navigation approach is developed. The tree row reconstruction and the vanishing point detection are integrated in an EKF. The approach is tested on the robot in closed control loop with different environmental configurations.

Since this paper relies on the vertical edges in the image frames, we will explore other environmentally relevant features such as detecting the tree canopies. Furthermore, combining visual odometry is another avenue to be pursued. The work will replace the wheel encoder, and consequently, a navigation system with only a monocular visual sensor will be used to drive the robot autonomously in the orchard.

ACKNOWLEDGEMENT

We thank our colleagues at Carnegie Mellon University, especially S. Nuske, B. Grocholsky, S. Maeta, and Q. Wang for their insightful inputs and invaluable help.

REFERENCES

[1] T. Gillespie, *Fundamentals of Vehicle Dynamics*. SAE International, 1992.

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, Cambridge University Press, 2004.

[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] T. Gevers and A. Smeulders, "Colour based object recognition," *Pattern Recognition*, vol. 32, pp. 453–464, 1999.

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

[6] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 5, pp. 263–296, 2008.

[7] J. Courbon, Y. Mezouar, and P. Martinet, "Indoor navigation of a non-holonomic mobile robot using a visual memory," *Autonomous Robots*, vol. 25, no. 3, pp. 253–266, 2008.

[8] A. Das, O. Naroditsky, Z. Zhiwei, S. Samarasekera, and R. Kumar, "Robust visual path following for heterogeneous mobile platforms," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 2010.

[9] S. Soundararaj, A. Sujeeth, and A. Saxena, "Autonomous indoor helicopter flight using a single onboard camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, Oct. 2009.

[10] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet, "Visual navigation of a quadrotor aerial vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, Oct. 2009.

[11] A. Kim and R. Eustice, "Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, Oct. 2009.

[12] H. Hile, A. Liu, G. Borriello, R. Grzeszczuk, R. Vedantham, and J. Kosecka, "Visual navigation for mobile devices," *IEEE Multimedia*, vol. 17, no. 2, pp. 16–25, 2010.

[13] 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.

[14] V. Leemans and M. Destain, "Application of the Hough transform for seed row localisation using machine vision," *Biosystems Engineering*, vol. 94, no. 3, pp. 325–336, 2006.

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

[16] M. Kise, Q. Zhang, and F. Mas, "A stereo vision-based crop row detection method for tractor-automated guidance," *Biosystems Engineering*, vol. 90, no. 4, pp. 357–367, 2005.

[17] X. Chai, F. Wen, and K. Yuan, "Fast vision-based object segmentation for natural landmark detection on indoor mobile robot," in *International Conference on Mechatronics and Automation*, Beijing, China, Aug. 2011.

[18] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[19] J. Conroy, G. Gremillion, B. Ranganathan, and J. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Autonomous Robots*, vol. 27, no. 3, pp. 189–198, 2009.

[20] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "MAV navigation through indoor corridors using optical flow," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 2010.

[21] J. Civera, D. Bueno, A. Davison, and J. Montiel, "Camera self-calibraction for sequential Bayesian structure form motion," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 130–134.

[22] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyserb, and P. Saydb, "Generic and real-time structure from motion using local bundle adjustment," *Image and Vision Computing*, vol. 29, pp. 1178–1193, 2009.

[23] M. Farenzena, A. Bartoli, and Y. Mezouar, "Efficient camera smoothing in sequential structure-from-motion using approximate cross-validation," in *10th European Conference on Computer Vision*, Marseille, France, Oct. 2008.

[24] N. Nourani-Vatani, J. Roberts, and M. Srinivasan, "Practical visual odometry for car-like vehicles," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[25] D. Scaramuzza, "1-point-RANSAC structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *International Journal of Computer Vision*, vol. 95, pp. 74–85, 2011.

[26] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

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

[28] M. Aly, "Real time detection of lane markers in urban streets," in *IEEE Intelligent Vehicles Symposium*, Eindhoven, The Netherlands, June 2008.

[29] Q. Truong and B. Lee, "New lane detection algorithm for autonomous vehicles using computer vision," in *International Conference on Control, Automation and Systems*, Seoul, Korea, Oct 2008.

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

[31] Y. Huang and Y. Pan, "Fast algorithm for structured and unstructured road detection," in *The 2nd International Conference on Image and Signal Processing*, Tianjin, China, Oct 2009.

[32] C. Rasmussen, "Grouping dominant orientations for ill-structured road following," in *IEEE International Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

[33] S. Singh, M. Bergerman, J. Cannons, B. Grocholsky, B. Hamner, G. Holguin, L. Hull, V. Jones, G. Kantor, H. Koselka, G. Li, J. Owen, J. Park, W. Shi, and J. Teza, "Comprehensive automaton for specialty crops: Year 1 results and lessons learned," *Journal of Intelligent Service Robotics*, 2010.

[34] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981, pp. 121–130.

[35] D. Bertsekas, *Nonlinear Programming*. Cambridge, MA, 1999.

[36] J. Zhang and D. Song, "Error aware monocular visual odometry using vertical line pairs for small robots in urban areas," in *Proc. of the AAAI Conference on Artifical Intelligence*, Atlanta, GA, July 2010, pp. 2531–2538.

[37] D. Song, H. Lee, J. Yi, and A. Levandowski, "Vision-based motion planning for an autonomous motorcycle on ill-structured roads," *Autonomous Robots*, vol. 23, no. 3, pp. 197–212, 2007.