3D CITRUS GROVE RECONSTRUCTION BASED ON EMPIRICAL MEASUREMENT

By

SANGHOON HAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2010

To all the people who believed in me

ACKNOWLEDGMENTS

I would like to express my sincere thanks to Dr. Thomas F. Burks, my supervisory committee chair and advisor, for introducing me to the subject of robot vision and for his continuous faith in me throughout this work. His advice and support enabled me to successfully complete all my graduate studies at the University of Florida. I would also like to thank to Dr. Lee, who gave me special advice. I am grateful to my other committee members: Dr. Beck, Dr. Dankel and Dr. Dixon for being a part of my dissertation committee. Their valuable insight and guidance contributed to the completion of this research work.

I am thankful to all my friends in our lab Agricultural Robotics and Mechatronics group for discussing and helping me with all of the experiments. I express my special gratitude to Greg Pugh and Mike Zingaro for their invaluable technical support and building system. I thank the personnel in the Department of Agricultural and Biological Engineering for providing flexibility. I thank my family and all the people who kept me encouraged for long time. Special thanks go to Ms. Sarah Baybordi for proofreading this document and sharing a special friendship. Lastly, I gratefully acknowledge the United States Department of Agriculture, Florida Department of Citrus and Florida Legislative Initiative for Citrus Harvesting for providing the necessary funds to carry on this research.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

3D CITRUS GROVE RECONSTRUCTION
BASED ON EMPIRICAL MEASUREMENT
By
SangHoon Han

April 2010

Chair: Thomas F. Burks
Major: Agricultural and Biological Engineering

My study contributes to the development of unmanned and automated techniques associated with two applications for citrus groves: ground-based scouting autonomous vehicles and harvesting robot manipulators. The duty of ground-based scouting autonomous vehicles is to collect global information, threading along alleyways in a grove, and harvesting robot systems explore and map local fruit positions on a canopy using vision systems. Two major techniques considered in my study are a global image mosaicing technique for the ground-based scouting autonomous vehicle and a local 3D reconstruction for the harvesting robot manipulator. Vision-based techniques follow common procedures related to features: feature detection, feature tracking and feature matching.

Since it must be assumed that there are no artificial landmarks in canopy scenes, it is necessary to extract features from leaves. A leaf detection algorithm was developed by combinations of morphological operations designed to look for the center of the leaf. Leaf detection can preserve the viability of dominant interest points by means of segmenting individual. To be robust against the disturbance and disappearance of interest points, an active mesh method is introduced. It was originally designed to detect shapes of objects, however, the method is used in tracking with an assumption that objects are not deformable during

consecutive frames. A multilayered active tree was invented to be more robust to noises by managing interest points hierarchically. The multilayered active tree is not only stable, but it determines the dominant features. Some of the matched sets are used in the estimation of optical flow and reconstruction. For the horizontal movement, a parallax which causes discords between images occurs. A sequential image mosaicing was implemented by cropping and stitching the overlapped area between images. Difficulties of an 8 point algorithm were pointed out and a Plücker coordinates system which provides simpler and intuitive computations was suggested for the 3D canopy reconstruction. An alternative way to reconstruct canopies was also conducted through a LADAR and a GPS receiver.

CHAPTER 1
INTRODUCTION

The objective of chapter is to describe the motivation why this research starts.

Citrus Groves in Florida

Cultivation such as pomiculture is a work that needs a bunch of labors and care. Increasing labor costs is becoming a considerable part of reasons for increasing the production cost and weakening competitiveness. Citrus is a huge industry of Florida, producing more than 70% of US's supply of citrus. Present citrus groves face with many issues. A unit grove is becoming large scale by uniting small groves for years. This means work capacity per person increases. Compared with increment of work duration, number of labor is decreasing. In practice, imported labors are replacing scarce labor power. It is required for farm owners to equip temporary accommodations and facilities on near to groves for labors for harvesting season. Furthermore, employing non-US citizens has potentially immigration problems. Exposing to chemicals such as spraying is also an obstacle which makes labors scruple at working. These challenges are being the pressing reasons that demand unmanned systems in applications for citrus groves.

Unmanned System

A fully automated unmanned system should be able to detect and analyze the environment by itself. Unmanned harvesting techniques do not only strengthen the price competition of farm products on international markets, but also contribute to the technical development of industry. Figure 1-1 shows a concept of a robot harvesting vehicle system. Two major applications can be briefly divided – autonomous vehicle system and harvesting robot system.

Autonomous Vehicles for Citrus Groves

The main function of autonomous vehicle systems requires a capability of navigating alleyways by itself. There are many trials not only agricultural applications, but also urban applications. My study focuses on only vision applications rather than automatic navigation.

Robotics for Citrus Groves

In practice, Florida oranges are being harvested by using mass product type machines because 95% of oranges are used to produce juice rather than raw fruits. However, individual fruit harvesting techniques are worth to develop for the fresh fruit market. A robot harvesting system which automatically maps fruit positions and harvest them can be one of agricultural applications about unmanned management.

Figure 1-1. Concept design of an unmanned robot harvesting and autonomous scouting system.

CHAPTER 2
LITERATURE REVIEW

This chapter reviews the previous research conducted in the area of agricultural applications and general concepts associated with the vision-based 3D reconstruction.

Agricultural Applications using Computer Vision System

Machine vision systems are a common acquisition device in various applications from industry to everyday life. It is also coming to play a critical part in agricultural applications associated with unmanned systems. Briefly, two kinds of platform can be considered in my work.

Autonomous Vehicle

Since most agricultural applications often cover a spacious field area, it is essential to include machine vision in the autonomous vehicles. Younse (2005) applied a machine vision system to an autonomous greenhouse sprayer, which tracks ground-based artificial landmarks as shown in Figure 2-1 A). Subramanian (2006) developed a steering control system of an unmanned tractor which combines with a camera and a laser range sensor as shown in Figure 2-1 B).

Harvesting Robot

Harvesting robot in Figure 2-2 A) is one of the most complicated agricultural applications at present. In the case of detecting fruits, spectrum analysis or shape detection techniques can be employed (Hannan 2004). Mehta (2006) studied a visual servo system which controls the positioning of a grabber. A vision system together with a range sensor (Rovira-Mas 2006) can be applied not only as a sensor to control, but also as a device to measure 3D space density maps (Rovira-Mas 2003; Kise 2005).

Requirements of Vision-Based 3D Reconstruction

When rendering a large or long 3D model using successive images, the whole image is aligned with sub-images taken from different views. To form the continuous image, an image mosaic technique can be adopted. The vision-based 3D reconstruction method consists of several procedures as follows.

- Matching correspondences – Feature Extraction, Feature Tracking
- Image mosaicing – Homography, Registration,
- 3D reconstruction – Depth Information, Texture Mapping, Virtual Reality

The following will be a series of detailed literature reviews. Due to the enormous amount of studies in the computer vision areas, only selected works will be cited, with a stronger focus on techniques considered and related with the issues of this dissertation.

Feature-Based Image Processing

Since the feature extraction depends on conditions or situations of an image, there is no unique way to apply it. It is necessary to take into account various methods before applying feature extraction. The first step in processing, when dealing with sequential or multiple images, establishes relationships between the different images.

Feature Matching

Feature matching searches for identical characteristics in multiple images, oftentimes referred to as 'correspondences'. The feature correspondences which should be known in order to achieve a mosaic, can be roughly divided into two classes: area-based and feature-based. Since the pixel-based method has weaknesses such as illumination, shape, occlusion, appearance or local minimum, the feature-based method is normally preferred. In the feature extraction, various features such as points (or colors), lines, areas or contents can be employed. For a planar image, an edge detector or a corner detector are used in most cases.

Area-based matching

Area-based matching or Template-based matching is a rudimentary method which assumes that there exist similar regions between images. There are two different approaches: Normalized Cross-Correlation (NCC) and Sum of Squared Differences (SSD). NCC measures the similarity, and SSD measures the dissimilarity between two patch images.

Normalized Cross-Correlation: To observe similarity between patch images, Normalized Cross-Correlation (NCC) is computed by subtracting the mean and dividing by the standard deviation. The formula can be represented as,

$$CC(i,j) = \frac{\sum_{W}(W - E(W))(I_{(i,j)} - E(I_{(i,j)}))}{\sum_{W}(W - E(W))^2 \sum_{I_{(i,j)}}(I_{(i,j)} - E(I_{(i,j)}))^2}, \tag{2-1}$$

where W is a sub-image and I is a template image. The sub-image should be larger than the template image.

Sum of Squared Differences: Sum of Squared Differences (SSD) is based on inferential statistics. A distance from a point to the mean of an image is a deviation. If the deviation is squared, then we have the sum of squares for the image.

$$E_{SSD}(u) = \sum_{i}(I_1(x_i + x) - I_0(x_i))^2 = \sum_{i} e_i^{\,2}, \tag{2-2}$$

where $u = (u,v)$ is the displacement, e_i called the residual error. When a template patch image is placed on a similar area, the result of computation has a pick point over the image. The performance between NCC and SSD is the same.

Feature-based matching

Feature extraction is a process that determines a relationship between images in a local image. A variety of features such as colors, lines, curves, patterns or multiple-combinations can

be chosen, but mostly the point feature invariant to transformations and directions is used in sequential images in which there is camera motion. The feature extraction technique is used for motion detection, object tracking, feature matching, image mosaicing, panorama, 3D reconstruction and object recognition.

Interest point (Harris corner detection) Naïve: The corner detection devised by Harris and Stephens is a universal approach used to detect point features, and many enhanced detection techniques have been proposed based on the corner detection. A corner is defined with an intersection between two edges, while an interest point is an invariable position to detect. Since the detector does not detect only actual corners, an 'interest point' in general is referred to as a 'feature' more than a 'corner'. In the literature, 'corner', 'interest point' and 'feature' are referred to interchangeably.

One important assumptions of the corner detection is that it can be derived from neighborhood points. The neighborhood of a feature should be different enough from another neighborhood after moving. The approximation of the second derivative between a patch window w and a shifted window $w(dx, dy)$ is given by following the equation.

$$D(dx,dy)=\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}M\begin{bmatrix}\Delta x & \Delta y\end{bmatrix}, \text{ where } M=\iint\begin{bmatrix}\frac{\partial I}{\partial x}\\ \frac{\partial I}{\partial y}\end{bmatrix}\begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix}w(x,y)dx,dy\,. \tag{2-3}$$

M should have two large eigenvalues with respect to the interest point. If the eigenvalue has a positive large number, it would be an edge. If both eigenvalues have zeros, it can be inferred there is no feature. Since the calculation of the eigenvalue is computationally expensive, it can be achieved by minimizing the following corner response function: (source:)

$$R=\lambda_1\lambda_2-\kappa(\lambda_1+\lambda_2)^2=\det(M)-\kappa\,\mathrm{trace}^2(M)\,, \tag{2-4}$$

where k is a tunable sensitivity parameter. Since the corner detection itself is not very robust in general, it frequently needs expert supervisions or redundancies introduced to prevent the effect of individual error.

Color-based method

In the case of autonomous mobile robots which work indoors or in the industry, artificial landmarks such as colors or specified patterns are employed to detect the location. Yoon (2001) proposed an artificial landmark resistant to change caused by altering intensities or shapes. The designed color checkered pattern uses a global/local histogram and an HSV color model which is less affected by illumination. Since the histogram is invariable within the pattern, it is impervious to noise.

Descriptor-based method

It is not enough for a single method to cope with incorrect matches caused by a variety of conditions such as change of scale, rotation and illumination. To avoid these varieties, particularly designed approaches have been suggested. The descriptor-based method is becoming the most promising approach.

Scale invariant feature transform: A Scale Invariant Feature Transform (SIFT) which is devised by Lowe (2004) is one of the advanced feature matching methods. The SIFT briefly consists of a key point localization and an identified descriptor around the key point as shown in Figure 2-3. The key point is determined by covering invariance to varieties. To detect a stable location of a key point, a Difference of Gaussian (DoG) approach is used over the scale-space representation of an original image. This convolution makes it resistant to scale variations. After the localization, the key point's orientation is assigned in order to cover rotational invariance. The descriptor composed of 128 normalized vectors is computed by local gradient orientations and magnitudes around the key point. The SIFT features have a lot of advantages such that they

are invariant to scale and rotation, highly distinctive, resistant to distortions, occlusions, noises and change of illumination. Due to its merits, many researchers are presently trying to apply it increasing presently (Lepetit et al., 2004; Heikkilä et al., 2005; Gräβl et al., 2005; Mortensen et al., 2005; Moreels et al., 2005). Ke et al. (2004) proposed a PCA-SIFT algorithm in order to reduce dimensions of descriptors. They applied Principle Component Analysis (PCA) to the normalized gradient patch instead of smoothly weighted histograms. Thus they insist that it should be faster and more resistant than the original SIFT.

Speeded up robust features: Speeded Up Robust Features (SURF) devised by Bay et al. (2006) is a descriptor-based feature matching method like SIFT. The descriptor illustrates distribution of intensity such as the gradient around an interest point. Unlike SIFT, SURF is designed to enhance computational speed by utilizing responses of the first order of Haar wavelet. SURF uses a Fast-Hessian detector for key point detection and Haar wavelet for descriptor. The response of Haar wavelet is fundamentally invariant to illumination changes like shadow as well as contrast invariant by taking the unit vector.

- Fast-Hessian Detector

Fast-Hessian is based on an approximation of a Hessian matrix. Given a point $x = (x, y)$ in an image, the Hessian matrix $H(x, s)$ in x at scale s is defined as follows.

$$H(x,s) = \begin{bmatrix} L_{xx}(x,s) & L_{xy}(x,s) \\ L_{xy}(x,s) & L_{yy}(x,s) \end{bmatrix}, \tag{2-5}$$

where $L_{xx}(x, s)$ is the convolution of the Gaussian second order derivative .

- SURF Descriptor

SURF explores 64 sub areas while SIFT explores all pixels within the frame. This simplification enhances not only computation speed but resistance because it is less sensitive to

noise. The first step for a descriptor is to determine a reproducible orientation from the circular neighborhood around the interest point. The next step is to extract a rectangular area along the orientation and then determine the descriptor with respect to this area.

- Orientation

Consideration of the circular neighborhood yields invariance to rotation. First it computes the response of the Haar wavelet at the circular neighborhood followed by determining the dominant orientation based on distributed responses from the weighted Gaussian center. To keep consistency of orientation, it is necessary to ascertain the appropriate size of the window through empirical means.

- Descriptor

Descriptor vectors are computed within the rectangular area. The rectangular area angled and centered on the interest point is extracted along the orientation determined in the previous step. The extracted rectangle is divided again into subareas with a 4x4 grid and then a Haar wavelet is performed at the sampling points distributed on the grid. Each subarea contains 4 descriptor vectors as follows.

$$\mathbf{v}_{i,j} = \left[\sum d_x \quad \sum d_y \quad \sum |d_x| \quad \sum |d_y|\right], \tag{2-6}$$

where $\sum d_x$ and $\sum d_y$ are sums of the responses, and $\sum |d_x|$ and $\sum |d_y|$ are sums of absolute values of responses with respect to the X and Y axis. Therefore, the descriptor with respect to an interest point has 64 vectors as shown in Figure 2-4.

Feature Tracking

Feature tracking is for matching correspondences in sequential images. The disparity between features gives us depth information to compute a homography. A success of registration and reconstruction depends on how well the features are tracked.

Motion estimation

Optical flow: An optical flow is defined as a distinct motion of image intensities. There are two major assumptions in the motion estimation. First, the brightness is steadily changed. Second, there is no object moving individually in the scene. When an object moves δx, δy in the scene while δt, the first assumption can be represented with Taylor series expansion.

$$I(x+\delta x, y+\delta y, z+\delta z, t+\delta t) = I(x,y,z,t) + \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial z}\frac{\partial z}{\partial t} + \frac{\partial I}{\partial t}\frac{\partial t}{\partial t} + O(x,y,z,t). \quad (2\text{-}7)$$

According to the second assumption that terms are constant, derivatives are zero.

$$\begin{aligned} &\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial z}\frac{\partial z}{\partial t} + \frac{\partial I}{\partial t}\frac{\partial t}{\partial t} = 0 \\ &I_x V_x + I_x V_y + I_x V_z = -I_t \end{aligned}. \quad (2\text{-}8)$$

$$\begin{aligned} &\nabla I \cdot V = -I_t \\ &\begin{bmatrix} I_{x_1} & I_{y_1} & I_{x_1} \\ I_{x_1} & I_{y_2} & I_{y_2} \\ \vdots & \vdots & \vdots \\ I_{x_n} & I_{y_n} & I_{z_n} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}. \\ &AV = -B \end{aligned} \quad (2\text{-}9)$$

This equation is called optical constraint. Since this equation has many solutions, more constraints are needed. To determine the optical flow, many methods such as a Lucas-Kanade method and a Horn-Schunck method are proposed.

Linear tracking model

LK tracker: A Lukas-Kanade method is one popular optical flow estimation. In this differential method, it is assumed that information based on intensities and colors or objects is not changed significantly. Assuming the optical flow of every pixel in the small window is the same, the computation can be simplified with solving a linear equation. Although a solution is

obtained with two points in a non-singular system, more points are more effective. For this over-determined system, a least-square approximation is used in common. For points, a weight function such as Gaussian is used.

$$\begin{aligned}\sum_{x,y} w(x,y) I_x I_y u + \sum_{x,y} w(x,y) {I_y}^2 v &= -\sum_{x,y} w(x,y) I_y I_t \\ \sum_{x,y} w(x,y) {I_x}^2 u + \sum_{x,y} w(x,y) I_x I_y v &= -\sum_{x,y} w(x,y) I_x I_t\end{aligned} . \tag{2-10}$$

The iteration also yields a better result. Yielded offsets in the computation are used to determine a new window.

$$\begin{aligned}&\nabla I \cdot V = -I_t \\ &\begin{bmatrix} I_{x_1} & I_{y_1} & I_{x_1} \\ I_{x_1} & I_{y_2} & I_{y_2} \\ \vdots & \vdots & \vdots \\ I_{x_n} & I_{y_n} & I_{z_n} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}. \\ &AV = -B\end{aligned} \tag{2-11}$$

To solve the taking pseudo inverse,

$$\begin{aligned}&V = (A^T A)^{-1} A^T (-B) \\ &\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \sum {I_{x_i}}^2 & \sum I_{x_i} I_{y_i} & \sum I_{x_i} I_{z_i} \\ \sum I_{x_i} I_{y_i} & \sum {I_{y_i}}^2 & \sum I_{y_i} I_{z_i} \\ \sum I_{x_i} I_{z_i} & \sum I_{y_i} I_{z_i} & \sum {I_{z_i}}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \\ -\sum I_{z_i} I_{t_i} \end{bmatrix}.\end{aligned} \tag{2-12}$$

Lukas-Kanade method is usually carried out in a coarse-to-fine iterative manner as SIFT. This algorithm does not yield a very high density of flow vectors. The optical flow fades out quickly across motion boundaries and the inner parts of large homogenous areas show little motion. Therefore, it is comparatively resistant against the presence of noise. Rav-Acha et al. (2004) uses a modification of the Lucas-Kanade for a direct 2D alignment.

Kalman filter: A Kalman filter that originated from a control theory is a traditional tracking method, which estimates coefficients of linear equations. It is applied to tracking

moving objects. Davison et al. (2003) emphasized that an Extended Kalman Filter (EKF) shows a great success in estimating locations of the robot with a single camera.

Non-linear tracking model

There are many applications that linear models cannot cover.

Horn-Schunck tracker: A Horn-Schunck method is designed to solve an aperture problem. A flow density is around a boundary, on the other hand, it is sensitive to noise.

Entended Kalman filter: An extended Kalman filter is for a non-linear version of the linear Kalman filter. In some research, these techniques are combined to take each advantage.

Stochastic tracking model

Conditional density propagation: Isard et al. (1998) devised a CONditional DENSity propagATION (CONDENSATION) algorithm based on a stochastic tracking for an active contour. In this algorithm, each position of feature is estimated based on the probability in the previous stage. The CONDENSATION algorithm applies factored samples iteratively to compute the *a posteriori* density. In the iterative step, it starts with a sample set s_1 which represents *a posteriori* density from the previous step $t-1$. Applied factored sampling, a set s_3 formed from s_2 which describes the *a priori* density $p(x_t \mid Z_{t-1})$ is represented as follows

$$p(x_t \mid Z_{t-1}) p(z_t \mid x_t) \propto p(x_t \mid Z_t). \tag{2-13}$$

While one formulation in the Kalman filter tracks only one feature, CONDESATION can expand the number of features. It is resistant against occlusions and temporary disappearances. Meier et al. (1999) applied this algorithm to the content-based tracking for mobile robots.

Geometric tracking model

Tracking models, as reviewed before, are based only on extracted features from images. Geometric tracking models such as an active contour and an active mesh refer to geometric

relationships among the features (nodes) as well as image-based features. These terms are defined as energy equations and controlled by weights. The active contour method is effective in detecting shapes or tracking geometric deformation.

Active contour: An active contour is an iterative framework delineating an object profile by minimizing energy which is defined with image properties. The energy is categorized by two aspects: internal and external energy.

- The external energy is minimal when a contour stays at the boundary of an object. It is measured by taking gradient between pixels.
- The internal energy is minimal when a contour matches a desired shape. It is based on an elongation (elastic force) or a curvature (bending force) deformation.

This approach that quantifies morphological constraints with energy has a potentiality which extends to other approaches like shape detection as well as tracking. Figure 2-5 shows an example result from an active contour.

Active mesh: Molly, D. (2000) proposed an active mesh method which extends from an active contour. While an active contour method defines a one dimensional relationship among nodes, an active mesh method defines multiple relationships among the closest nodes as shown in Figure 2-6. A Delaunay triangulation is one of typical grouping methods for nearby nodes. The active mesh is effective in the background of a scene which is not deformed or extracting moving objects from the background. Thereby, it is applied to video compression as well as tracking.

While feature extractions in the active contour have been conducted every frame, Griffin, (2001) suggested a content-based active mesh method. This approach finds node positions that minimize errors between the patch image and the reference image. Since it is assumed that each patch image is a planar (affine transformation), it does not seem to be effective in a sudden occlusion.

Structure and Motion Estimation

Once correspondences are determined through various image processing methods mentioned in the previous section, then the camera extrinsic parameters and the depth of features based on the correspondences can be estimated. This reconstruction is often called Structure From Motion (SFM). In this section, the basic theory behind the procedure and some applications which expand those concepts is discussed. A camera is a device which projects a 3D scene into a 2D image. Therefore, a camera can be modeled as a projection matrix. (source: Ma et al. 2004)

Camera Model and Geometry

Pinhole camera model

Throughout my study, an ideal pinhole camera model is used, which is referred to generally. As shown in Figure 2-7, the projection center C is placed as far as focal length f on the principle axis perpendicular to the center c of an image plane $\Re$ (retinal plane). An image point $\mathbf{m} \in \Re^3$, which intersects the image plane $\Re$ and a line connecting the projection center C and an object $\mathbf{M} \in \Re^4$ in the scene, can be simply modeled by a projection matrix $\mathbf{\Pi}_0 \in \Re^{3\times4}$ as

$$\mathbf{m} \cong \mathbf{\Pi}_0 \mathbf{M} . \tag{2-14}$$

Intrinsic parameters

For an input data $\mathbf{m}_\Pi \in \Re^3$ from a CCD sensor to indicate projected point $\mathbf{m} \in \Re^3$, a transformation matrix $\mathbf{K}_{\text{int}} \in \Re^{3\times3}$ between them, which is called the intrinsic parameters of the camera, is needed. The intrinsic parameters are specified with several fixed properties of the camera.

$$\mathbf{m} = \mathbf{K}_{\text{int}}\mathbf{m}_{\Pi}$$
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & o_x \\ & f_y & o_y \\ & & 1 \end{bmatrix} \begin{pmatrix} x_{\Pi} \\ y_{\Pi} \\ 1 \end{pmatrix} \tag{2-15}$$

where f_x, f_y are focal length with respect to x, y axis. o_x, o_y are the center of an image axis. s is a skew angle as shown in Figure 2-8.

Extrinsic parameters

A 3D point $\mathbf{M}$ can be represented with a reference point $\mathbf{M}'$ by a pose matrix with respect to the base camera.

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{M}', \tag{2-16}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{T}$ is a translation vector. If a camera pose has changed, the reference point is simply given by taking an inverse of the pose, which is called the extrinsic parameters of a camera $\mathbf{K}_{\text{ext}} \in \Re^{4\times 4}$.

$$\begin{aligned} \mathbf{M}' &= \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{M} \\ \mathbf{M} &= \mathbf{K}_{\text{ext}}\mathbf{M}' \end{aligned}. \tag{2-17}$$

Camera projection matrix: Combining those specified parameter matrices, a generalized camera projection matrix can be obtained:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \cong \begin{bmatrix} f_x & s & o_x \\ & f_y & o_y \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2-18}$$
$$\begin{aligned} \mathbf{m} &\cong \mathbf{K}_{\text{int}}\mathbf{\Pi}_0\mathbf{K}_{\text{ext}}\mathbf{M} \\ \mathbf{m} &\cong \mathbf{K}_{\text{int}}\begin{bmatrix}\mathbf{R}^T & -\mathbf{R}^T\mathbf{T}\end{bmatrix}\mathbf{M} \end{aligned}.$$

Image Mosaic

Since cameras have a limited view, it is necessary to merge multiple images to be a wide single like Figure 2-9. Image mosaicing is a framework that builds a high resolution image with multiple subimages. Features extracted from the duplicated area are used to estimate camera motions. Appropriate warping transformation is carried out based on the estimated camera motion. The warped image can be folded exactly. Approach depends on the camera motion type.

Motion type

Parallax: A parallax is a phenomenon that occurs when the background of an object is changed with respect to view points. Since this effect is based on a distance from the object, the optical flow is also used to measure the distance of objects. This effect does not happen during a pure rotation.

Pure rotation: An image mosaic uses either a cylindrical type or a spherical type of projection model in general. These projection models have a single common camera center. Since no parallax happens with the models, it is simple to achieve a mosaicing with pixel-based matching methods. If images are taken with translational motion, the distance of objects and the estimation of camera motion should be considered.

Estimation of motion

To combine images, there should be an overlapped area between images. For the estimation of camera motions, a homography is widely used.

Homography: A homography in computer vision is defined as a collineation which maps between sets of correspondences obtained from different planes. This relationship can be represented with a transformation matrix $\mathbf{HII}_i$ with respect to plane $\mathbf{II}_i$ as shown in Figure 2-10. There are a lot of applications using homography and it provides ways of composing images or video frames.

The features may be obtained from a plane such as an artificial structure on occasion. In this degenerative configuration, an 8 point algorithm would not give a unique solution. Thus different approaches are needed. A plane equation can be represented as follows

$$\mathbf{N}^T\mathbf{M} = n_1 x + n_2 y + n_3 z = d\,,\ \frac{1}{d}\mathbf{N}^T\mathbf{M} = 1\,, \tag{2-19}$$

where d is the distance from the origin to the plane. Substituting this to the rigid-body equation, we obtain

$$\mathbf{M}_2 = \mathbf{R}\mathbf{M}_1 + \mathbf{T} = \mathbf{R}\mathbf{M}_1 + \mathbf{T}\frac{1}{d}\mathbf{N}^T\mathbf{M}_1 = \left(\mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T\right)\mathbf{M}_1 = \mathbf{H}\mathbf{M}_1\,,\ \mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T\,, \tag{2-20}$$

where $\mathbf{H} \in \Re^{3\times 3}$ is called a collineation or a homography. It is a 3x3 homogeneous matrix whose rank is 2. However, there is no information that extracted features are obtained from a plane. If a camera moves with pure rotation, i.e. $\mathbf{T} = 0$ or a camera is infinitely far away from objects i.e. $d = \infty$, the homography becomes a function of only $\mathbf{R}$. Each depth of features d and planar structure $\mathbf{N}$ to determine the homography do not matter. Thus many image mosaicing methods have been carried out with image sets taken from purely rotating camera.

$$\hat{\mathbf{m}}_1\mathbf{R}\mathbf{m}_2 = 0\,. \tag{2-21}$$

Composition

Warping: Warping is a process which deforms an image to fit to a projection model. Since images taken with different view point do not have the same projection plane, the overlapped areas between images are not identical. Deforming images based on the base camera motion, images can be combined with seams as shown in Figure 2-11

Blending: When mosaicing with images taken from outdoors, the boundary edges between images aligned might be salient due to changes of illumination. As shown in Figure 2-12, Hasler et al. (2004) proposed a method to correct for color differences between images stitched together.

The color correction consists in retrieving linearized relative scenes referred to data from uncalibrated images by estimating the opto-electronic conversion function (OECF) and by correcting for exposure, white-point, and vignetting variations between the individual pictures.

Applied techniques

A mosaicing technique is used in various applications such as topological terrain maps, satellite imagery, high resolution medical images, video compression and so on.

Manifold: Jun J.C. (2004) achieved an image mosaicing and 3D reconstruction by using a manifold projection. The manifold projection contains an individual projection plane every frame. He assumed that the path of a camera is two dimensional, and the extrinsic parameters are estimated by measuring the optical flow with three images. However, incongruity on the projection plane is incurred as much as the difference of depth. Even though it cut off a discord boundary with a curve estimated with the same pixels, errors are incurred in the distinct non-planar scene because it approximates the boundary of the projection with compensation between intersections.

Time warping: Rav-Acha et al. (2004) estimated depth information by means of time warping, which expands a time axis from an image plane. To simplify the computation, a camera moves in a one dimensional direction with a constant speed and the camera view is orthogonal to the path. Therefore, no transformation is used. Since the assumed constant moving speed forms a line on the time axis, the depth of each feature can be estimated as much as the line is declined. That is, points that are too far away become almost a vertical line, and the closer points become the declined line as shown in Fig 2-13. The feature patches are divided by a Voronoi diagram and set the width of patch as much as the depth. Since the depth information is represented by a line on the time axis, it can be estimated even though features disappear while tracking sometimes.

3D Reconstruction

Correspondences do not generally come from only a plane. Thus a homography cannot be applied in terms of arbitrary motions of the camera. This section induces equations which represent the general relationship between correspondences from a pair of images based on an epipolar geometry. (source: Ma et al. 2004)

Estimation of structure and motion

Depth information with a camera is estimated by using correspondences taken from over two images. For these correspondences to reflect actual features of an object, those should basically satisfy an epipolar constraint. The epipolar geometry can be described through several ways.

- If neither intrinsic nor extrinsic parameters of a camera are known, the epipolar geometry is represented by a Fundamental matrix.
- If only intrinsic parameters of a camera are known, the epipolar geometry is represented by an Essential matrix.
- If both intrinsic and extrinsic parameters of a camera are known, the epipolar geometry is represented by a Projection matrix.

The relative motion (rotation and translation) of a camera can be estimated with the epipolar constraint. Once the camera extrinsic parameters are determined, spatial positions of features are also determined with a transformation matrix easily.

Estimation of essential matrix

Epipolar constraint: As shown in Figure 2-14, when calibrated correspondences x on each image plane reflect a spatial point X, the relation between them can be represented with a multiplication of matrix as

$$\lambda \mathbf{m} = \mathbf{\Pi}_0 \mathbf{M}, \tag{2-22}$$

where $\lambda \in \Re^+$ is a scalar and $\mathbf{\Pi}_0 = [\mathbf{I} \quad 0]$ is a camera projection matrix. Setting one of two cameras as a reference frame, a spatial point $\mathbf{M}_2 \in \Re^3$ with respect to the other camera o_2 can also be represented with a relative pose $\mathbf{R}$, $\mathbf{T}$ and an identical point $\mathbf{M}_1 \in \Re^3$ with respect to the reference frame o_1 as

$$\mathbf{M}_2 = \mathbf{R}\mathbf{M}_1 + \mathbf{T}. \tag{2-23}$$

Since $\mathbf{M}_i = \lambda_i \mathbf{m}_i$, where $\mathbf{m} \in \Re^3$ is homogenous coordinates, the relationship of correspondences is represented as follows once again.

$$\lambda_2 \mathbf{m}_2 = \mathbf{R}\lambda_1 \mathbf{m}_1 + \mathbf{T}. \tag{2-24}$$

To remove scalars λ_i which denote physical depths, applying a skew matrix $\hat{\mathbf{T}}$ of the vector $\mathbf{T}$ to both sides, it becomes

$$\lambda_2 \hat{\mathbf{T}}\mathbf{m}_2 = \hat{\mathbf{T}}\mathbf{R}\lambda_1 \mathbf{m}_1, \tag{2-25}$$

where $\hat{\mathbf{T}}\mathbf{m}_2 = \mathbf{T} \times \mathbf{m}_2$. Since $\hat{\mathbf{T}}\mathbf{x}_2$ is perpendicular to x_2, $\mathbf{m}_2{}^T \hat{\mathbf{T}}\mathbf{m}_2 = 0$. Multiplying $x_2\mathbf{T}$ to both sides again, the correspondences have the following relationship regardless of scalars λ_i,

$$\mathbf{m}_2{}^T \hat{\mathbf{T}}\mathbf{R}\mathbf{m}_1 = 0 \text{ or } \mathbf{m}_2{}^T \mathbf{E}\mathbf{m}_1 = 0, \tag{2-26}$$

which is called an epipolar constraint and $\mathbf{E} \in \Re^{3\times3}$ is called an essential matrix. Since $\det(\mathbf{E}) = 0$, it is singular and a homogenous quantity. It has only five degrees of freedom: 3 in rotation and 2 in translation up to scalar. Since $\mathbf{E}$ contains only $\mathbf{R}$ and $\mathbf{T}$ information, solving the $\mathbf{E}$ that satisfies the equation with given correspondences, we can determine $\mathbf{R}$ and $\mathbf{T}$.

Estimation of motion

To solve $\mathbf{E}$, the epipolar constraint equation is modified into a linear form.

$$\mathbf{A}^T \mathbf{E}^s = 0, \tag{2-27}$$

where $\mathbf{E}^s = [e_{11} \quad e_{21} \quad e_{31} \quad e_{12} \quad e_{22} \quad e_{32} \quad e_{13} \quad e_{23} \quad e_{33}]^T \in \Re^9$ is referred to as the stacked version of $\mathbf{E}$. $\mathbf{A}^T = \mathbf{m}_1 \otimes \mathbf{m}_2 \in \Re^{n\times 9}$ and operator $\otimes$ denotes a Kronecker product. For example, given $\mathbf{m}_1 = (x_1 \quad y_1 \quad z_1)^T$, $\mathbf{m}_2 = (x_2 \quad y_2 \quad z_2)^T$,

$\mathbf{m}_1 \otimes \mathbf{m}_2 = (x_1x_2 \quad x_1y_2 \quad x_1z_2 \quad y_1x_2 \quad y_1y_2 \quad y_1z_2 \quad z_1x_2 \quad z_1y_2 \quad z_1z_2)$.

A solution of $\mathbf{E}$ satisfying the equation is an eigenvector of $\mathbf{A}$ which corresponds to an eigenvalue equal to zero. Applying Singular Value Decomposition (SVD) to the coefficient matrix $\mathbf{A}$, then the last column of $\mathbf{V}$ which corresponds to zero of the eigenvalue becomes $\mathbf{E}$.

$$\begin{aligned}\mathbf{E} &= \mathbf{U}\Sigma\mathbf{V}^T \\ &= \mathbf{U}\mathbf{R}_z\left(\pm\tfrac{\pi}{2}\right)\mathbf{\Sigma}\mathbf{R}_z\left(\pm\tfrac{\pi}{2}\right)\mathbf{V}^T \\ &= \mathbf{U}\mathbf{R}_z\left(\pm\tfrac{\pi}{2}\right)\mathbf{U}^T\mathbf{\Sigma}\mathbf{U}\mathbf{R}_z\left(\pm\tfrac{\pi}{2}\right)\mathbf{V}^T \\ &= \hat{\mathbf{T}}\mathbf{R}\end{aligned}, \tag{2-28}$$

where $\hat{\mathbf{T}} = \mathbf{U}\mathbf{R}_z\left(\pm\tfrac{\pi}{2}\right)\mathbf{U}^T, \mathbf{R} = \mathbf{\Sigma}\mathbf{U}\mathbf{R}_z\left(\pm\tfrac{\pi}{2}\right)\mathbf{V}^T$. There is now a pair of motions. However, in practice the eigenvalue would not be zero exactly. Therefore, E is determined by minimizing the norm of difference between the estimated solution $\tilde{\mathbf{E}} = \mathbf{U}diag(\lambda_1, \lambda_2, \lambda_3)\mathbf{V}^T$ and the exact solution $\mathbf{E}_0 = \mathbf{U}diag(\sigma_1, \sigma_2, 0)\mathbf{V}^T$.

$$\min\left\|\tilde{\mathbf{E}} - \mathbf{E}_0\right\|^2. \tag{2-29}$$

Notice that $-\mathbf{E}$ also satisfies the same set of the epipolar constraint resulted from $\mathbf{E}$. Thus there are four possible cases and three of them from B) to D) are unrealistic as shown in Figure 2-15. A realistic one A) can be chosen by inspecting positive depth.

Random sample consensus: As seen before, the estimation of the relationship between correspondences is carried out with the assumption that the measurement of feature points is exact. In practice, however, the correspondences are exposed enough to noises, and a little

contamination of noises can make the final results useless. The problem is that it is difficult to segment the set of input data into inliers and outliers before the exact solution is obtained. One solution is RANdom SAmple Consensus (RANSAC), which is a popular algorithm in the computer vision area. This algorithm is an iterative method to estimate a mathematical model from a set of data and it can be applied to various problems. RANSAC also assumes that in a given set of inliers there exists a process which describes the data or estimates an appropriate model. The objective of this algorithm is to segment observed input data into inliers fitting to parameters of a model and outliers such as noise. The outlier can be caused by amplified noises, coarse measurements or incorrect hypotheses. The segmented inliers are used not only to refine the solution, but will be used to reconstruct the structure of a scene in the continuous frame in which it is important for features to survive as long as possible.

RANSAC is iteratively carried out by randomly selecting a subset of data. The subset of data is hypothetical inliers and they are tested as the following steps:

1. take a subset of hypothetical inliers from observed input data and estimate hypothetical parameters of a model.
2. count the number of data which fit to the model by applying other data to the model.
3. after repeating these steps, choose a hypothetical model which has large number of data.
4. segment observed data into inliers and outliers by applying data to the chosen model.

In the case of a Homography as seen before, when all feature points are not extracted from a plane, the homography induced from the plane creates a virtual parallax. Since a homography assumes that points originate from a plan, it is effective for RANSAC to segment feature points into the points originated from a plane, i.e. inliers and the others. RANSAC is designed so that it can just estimate one model for a specific set of data. In case that one is multiplying a homography, therefore, RANSAC should be applied for each homography individually.

One advantage of RANSAC is the ability to robustly estimate parameters of a model with a high accuracy even in the presence a significant number of outliers. In contrast, one disadvantage is that it requires thresholds specified for each problem. There is no time limit to repeat the iteration. When one stops at some iteration, there is no guarantee that it would be an optimal solution.

Bundle adjustment: A bundle adjustment is one kind of optimization techniques. It has increasingly been used for the last step of feature-based 3D reconstruction in the computer vision area. This method is used to minimize the distance between observed feature points and re-projected feature points in terms of estimated parameters of a camera model.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} d(P(x_j, X_i), x_{ij})^2 , \tag{2-30}$$

where $P(x_j, X_i)$ is the evaluated projection of point X_i on image x_j and $d(x, y)$ denotes the Euclidean distance between image coordinates. When the minimization problem is solved, the equation may have a sparse block due to irrelevances between points and camera. Thus a Levenberg-Marquardt algorithm is also used in the optimization problem, which robustly converges even if the initial guesses begin far away from the final minima.

Estimation of structure

As seen before, given more than eight points, an 8 point algorithm can determine the relative $\mathbf{R}$ and $\mathbf{T}$ between cameras accompanied with a scale factor $\gamma \in \Re^+$. If $\gamma = 1$, the obtained pose is equal to a translation with unit length. Now that the motion of a camera is determined, the structure can be reconstructed based on the pose and the correspondences used for an 8 point algorithm (source: Ma et al, 2004). Rewriting the basic rigid body equation (2-24) in terms of these parameters,

$$\lambda_2^j \mathbf{x}_2^j = \lambda_1^j \mathbf{R}\mathbf{x}_1^j + \gamma \mathbf{T}, j = 1,2,\ldots,n. \tag{2-31}$$

Either λ_1 or λ_2 is redundant. To eliminate λ_2, modifying the basic rigid-body equation by multiplying $\hat{\mathbf{x}}_2$ to both sides, we obtain

$$\lambda \hat{\mathbf{x}}_2{}^j \mathbf{R}\mathbf{x}_1{}^j + \hat{\mathbf{x}}_2{}^j \mathbf{T} = 0, j = 1,2,\ldots,n, \tag{2-32}$$

which can be represented with a linear equation as

$$\mathbf{M}^j \bar{\boldsymbol{\lambda}}^j \cong \begin{bmatrix} \hat{\mathbf{x}}_2^j \mathbf{R} \hat{\mathbf{x}}_1^j & \hat{\mathbf{x}}_2^j \mathbf{T} \end{bmatrix} \begin{bmatrix} \lambda_1^j \\ \gamma \end{bmatrix} = 0, \tag{2-33}$$

where $\mathbf{M}^j \in \Re^{3\times 2}$, $\bar{\boldsymbol{\lambda}}^j \in \Re^2$, for all $j = 1,2,\cdots,n$. To give a unique solution, $\mathbf{M}^j$ should have rank 1, which is not the case that points lie on the base line between the center of the cameras o_1, o_2, i.e. $\hat{\mathbf{x}}_2 \mathbf{T} = 0$. To simplify the equation, defining $\mathbf{M} \in \Re^{3n\times(n+1)}$, $\bar{\boldsymbol{\lambda}} \in \Re^{n+1}$ for j as

$$\bar{\boldsymbol{\lambda}} = \begin{bmatrix} \lambda_1^j & \cdots & \lambda_1^n & \gamma \end{bmatrix} = 0, \tag{2-34}$$

$$\mathbf{M} \cong \begin{bmatrix} \hat{\mathbf{x}}_2^1 \mathbf{R}\mathbf{x}_1^1 & 0 & 0 & 0 & 0 & \hat{\mathbf{x}}_2^1 \mathbf{T} \\ 0 & \hat{\mathbf{x}}_2^2 \mathbf{R}\mathbf{x}_1^2 & 0 & 0 & 0 & \hat{\mathbf{x}}_2^2 \mathbf{T} \\ 0 & 0 & \ddots & 0 & 0 & \vdots \\ 0 & 0 & 0 & \hat{\mathbf{x}}_2^{n-1} \mathbf{R}\mathbf{x}_1^{n-1} & 0 & \hat{\mathbf{x}}_2^{n-1} \mathbf{T} \\ 0 & 0 & 0 & 0 & \hat{\mathbf{x}}_2^n \mathbf{R}\mathbf{x}_1^n & \hat{\mathbf{x}}_2^n \mathbf{T} \end{bmatrix}. \tag{2-35}$$

We obtain the same linear equation

$$\mathbf{M}\bar{\boldsymbol{\lambda}} = 0, \tag{2-36}$$

where a vector $\bar{\boldsymbol{\lambda}}$ which represents the depth of each correspondence up to a single universal scale is an eigenvector of $\mathbf{M}^T\mathbf{M}$ corresponding to the minimum eigenvalue. To obtain a non-trivial solution of $\bar{\boldsymbol{\lambda}}$, $\mathbf{M}$ should be zero. As we have done at the essential matrix, the solution is also obtained by applying SVD in which the column of $\mathbf{M}$ corresponding to the minimum eigenvalue becomes the structure position. Since these spatial positions are solved

with the assumption that the translation was normalized, actual positions are determined by multiplying the actual distance between two cameras.

Applied techniques

Since 3D reconstruction is essentially based on a pair of images, a stereo vision system is widely used as shown in Figure 2-16. However, a moving single vision system can also provide multiple view, some researchers try to utilize it for 3D reconstruction.

Multiple camera: Depth is estimated by measuring a disparity between images. Since a stereo vision system provides two images with known configurations, a reconstruction can be achieved with stationary motion. It is the best way to be robust against potential errors.

Kise et al (2006) applied a stereo system to a weed detection and a furrow reconstruction. They assumed the camera was fixed on a vehicle which moves straight forward. Three or more cameras can be also used for measuring depth (Kim 2003; Klechenov et al. 2002). Shlyakhter et al (2001) rendered a 3D vegetation model based on multiple real images. Each of input image is segmented into a tree and a background. Silhouettes are used to construct a visual hull. Figure 2-17 shows that their system constructs a tree skeleton. This research assumes the extrinsic parameters are known. The surface of the model is formed based on Voronoi approximation.

Single Camera: The cost of a stereo system is expensive, and it is complicated to implement. Many researchers are trying to overcome such potential errors which occur with a single camera. The issue in a single camera is mainly associated with the estimation of a camera motion.

Dornaika, F. (2001) tried to overcome a parallax problem which makes occlusions during translational motions. Nevertheless, this parallax problem also provides the depth information. Zhu (2003) estimated depth information through stereo mosaic images as before. Taking images of non-planar scene with translational motion, the left and right sides of an image with respect to

moving direction show the different lateral face of a structure. The stereo mosaic images generated by the left and right strip provide disparity information about correspondences as shown Figure 2-18.

Augmented reality: An Augmented Reality (AR) is a rising field of computer research which is associated with a combination of generated 3D graphics and real image as shown in Figure 2-19. Mostly fiducial markers are used to make it easy to estimate the camera motion. Once particular markers are detected, 3D models are mapped based on the coordinates of markers.

Simultaneous localization and mapping: A Simultaneous Localization and Mapping (SLAM) is a technique mostly used by autonomous vehicle systems which navigate in the unknown environment. This kind of machine requires abilities to keep tracking its location and to generate mapped environments accurately. Klein et al. (2007) showed state-of-the art advances in terms of markerless tracking and mapping. Markerless means that they keep generating new markers from new features, not particular fixed markers. They used enhanced-Kalman filter SLAM for predicting features. Their approach keeps many key frames as ground truth. Figure 2-20 shows features which are tracked and mapped on a real image.

Different approaches

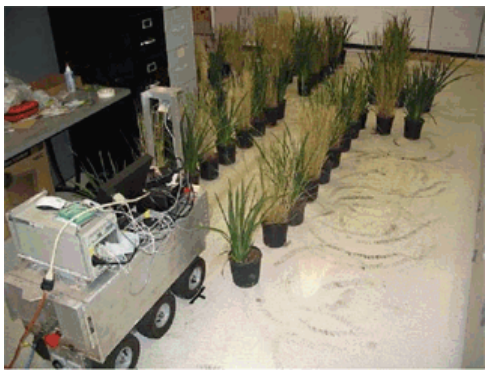
Various range sensors such as laser range sensor or an ultrasonic sensor can be used to measure a depth in remote sensing. Ground-based remote sensing techniques have been studied using laser range sensors or ultrasonic sensors for the last half-decade.

Laser Detection And Ranging (LADAR): Even though aerial-based remote sensing has a limitation to measure a depth (height of canopy), Meron et al. (2000) studied measurement of the shape and volume of the canopy with aerial photogrammetry. However, a range sensor is more popular with a tree-specific management. Wei et al. (2004) developed a system measuring the

volume of the citrus canopy using multiple laser scanners. As shown in Figure 2-21 A), their system scans like a semicircle, but the beam source moves along the guide frame which is fixed on the ground. They designed the system for only one canopy.

Ultrasonic sensor: An ultrasonic sensor can also be considered to measure the volume of the canopy. Zaman et al. (2005) developed a measurement system with an ultrasonic sensor and a DGPS receiver. Several pieces of ultrasonic sensors are attached to a pole on a moving trailer instead of a moving the direction of sources as shown in Figure 2-21 B). Tumbo et al. (2001) compared correlations among laser, ultrasonic and manual measurements. They concluded that a laser measurement can provide a good estimation of canopy volumes especially in a grove where there are significant numbers of partially defoliated trees or small replants. The other advantage of the laser measurement over an ultrasonic system may be the high speed of data acquisition with the former.

So far, successful papers have been reviewed. These papers are achieved under their assumptions. It is important to take into account whether or not their assumptions are suitable enough to the field.


A

B

Figure 2-1. Vehicle applications. A) Six-wheel greenhouse sprayer (Younse, 2005). B) Autonomous tractor (Subramanian, 2005).


A

B

Figure 2-2. Robotics applications. A) Vision-based harvesting robot (Hannan et al., 2004). B) Vision system for aiming fruit (Hannan et al., 2004).

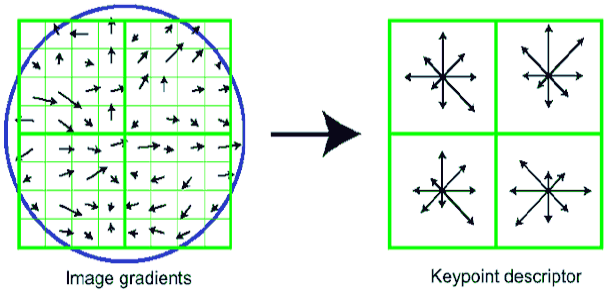


Figure 2-3. SIFT feature descriptor (Lowe, 2004).

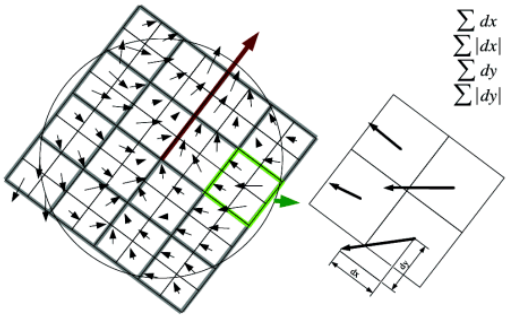


Figure 2-4. SURF 64 descriptor vectors at an interest point (Bay et al., 2006).

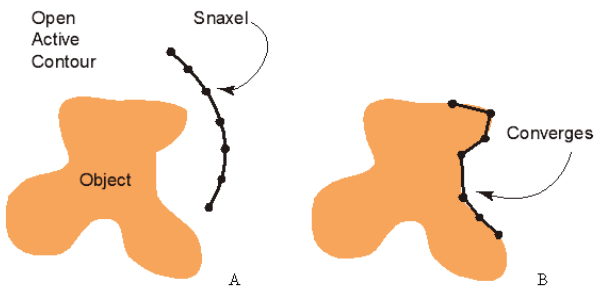


Figure 2-5. Active contour model (Molly, 2001).

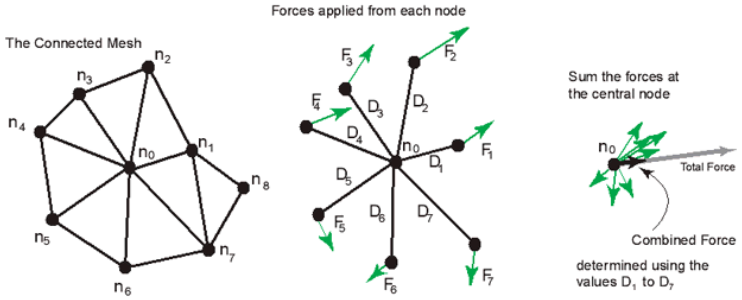


Figure 2-6. Active mesh model (Molly, 2000).

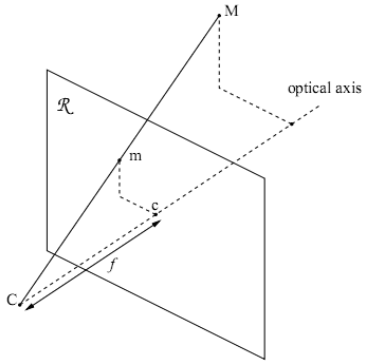


Figure 2-7. Pinhole camera model.

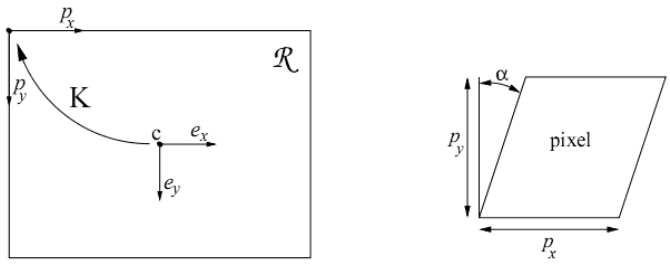


Figure 2-8. Intrinsic camera parameters.

Figure 2-9. Panorama from pure rotation.

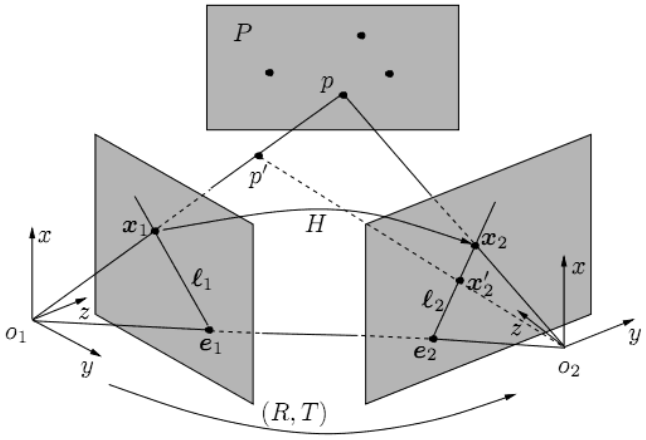


Figure 2-10. Homography (source: Ma, 2004).

Figure 2-11. Warping. A) Warping model. B) Registration after warping (Kanazawa, 2004).

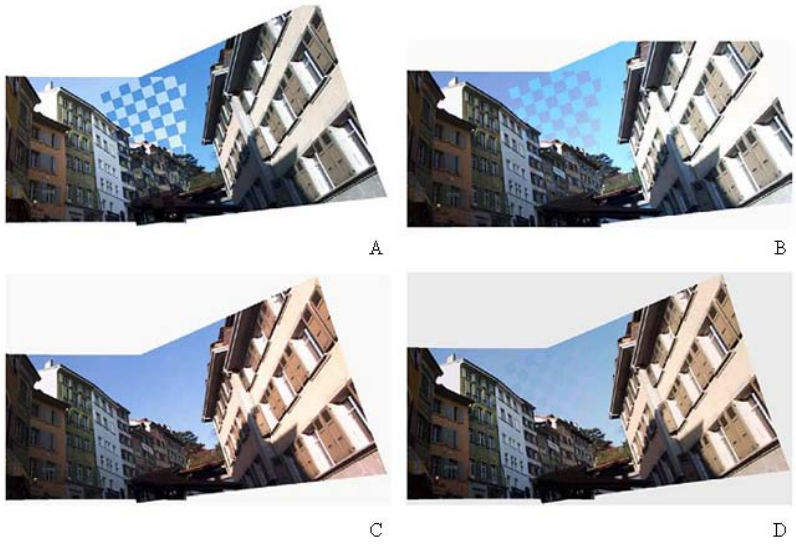


Figure 2-12. Color correction process (Hasler et al., 2004). A) Original mosaic. B) Exposure compensation by extrinsic parameter. C) Exposure compensation and white balance with polynomial OECF. D) Exposure compensation and white balance with Laguerre OECF.

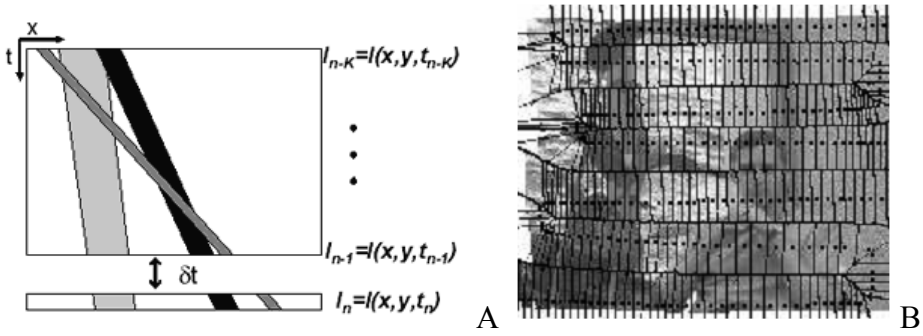


Figure 2-13. Time warping (Rav-Acha. et al., 2004). A) Features movement in time axis. B) Patches divided by Voronoi diagram.

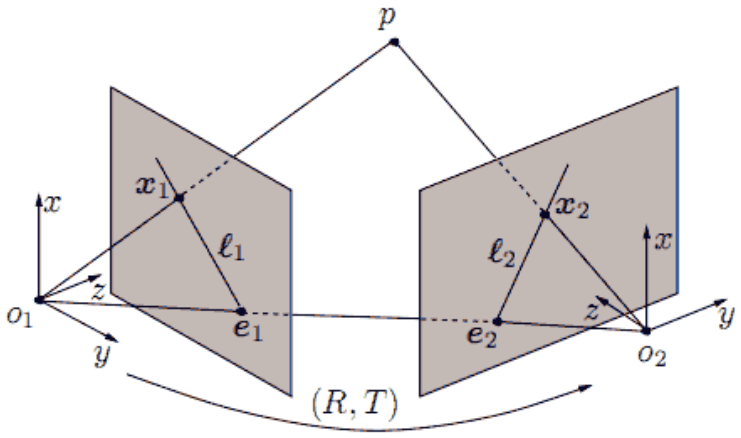


Figure 2-14. Epipolar geometry (source: Ma et al., 2004).

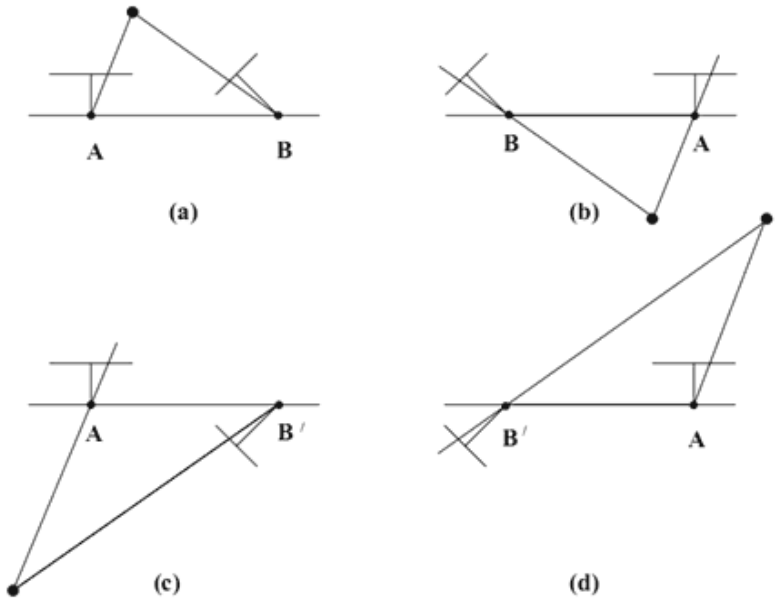


Figure 2-15. Four possible cases of reconstruction (source: Ma, 2004).

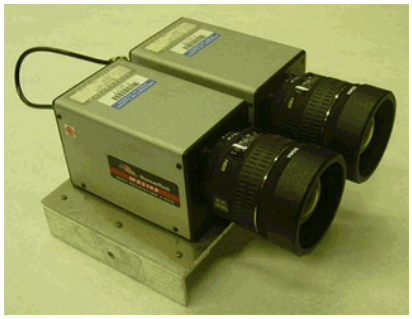


Figure 2-16. Stereo vision system (Kise et al., 2006).

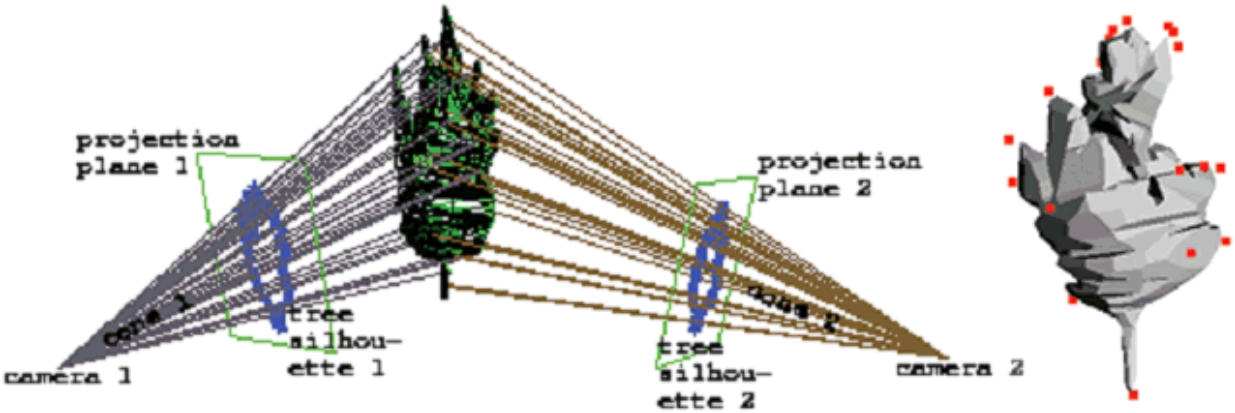


Figure 2-17. 3D reconstruction of tree model (Shlyakhter et al., 2001).

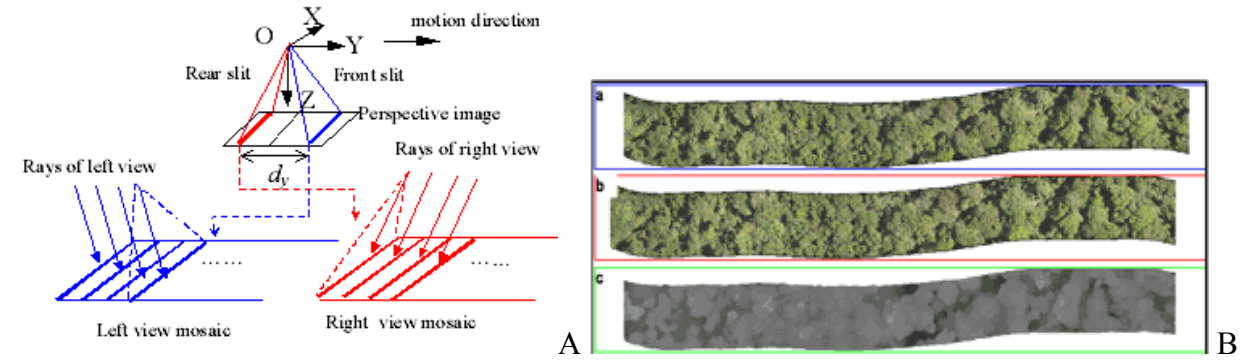
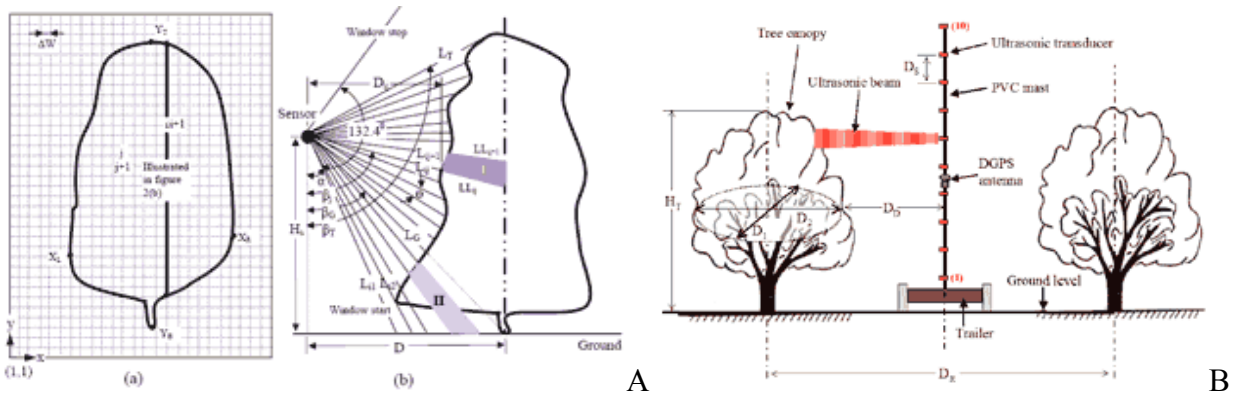


Figure 2-18. Parallel-perspective stereo mosaic (Zhu et al., 2003). A) mosaic model B) Depth map obtained from stereo mosaic. a) Left Mosaic. b) Right Mosaic. c) Depth map from stereo mosaics.

Figure 2-19. Augmented reality.

Figure 2-20. Augmented reality with SLAM (Klein et al., 2007).



A B

Figure 2-21. Measurement of canopy. Measurement of canopy with A) laser sensors (Wei et al., 2004), and with B) ultrasonic sensors (Zaman et al., 2005).

CHAPTER 3
OBJECTIVES

Objective Statement

The overall objective of this dissertation is to develop techniques required to generate a 3D canopy model based on an actual citrus grove.

Scope

These techniques can be used for scouting autonomous vehicle systems to inspect disease or growth status of plants throughout a global grove or for a harvesting robot systems to track and map fruit positions within a local canopy. Moreover, the 3D canopy model can provide farmers with a visual interface to manipulate unmanned systems remotely.

In an effort to generate 3D models, range sensors could be used as well. However, vision systems such as cameras are more commonly used. In my study, the primary focus will be on vision systems. Feature points obtained from images are approximated to vertices which forms a surface of 3D canopy models. Since a single image cannot cover the whole canopy, a wide panoramic image should be prepared to map the image on the surface of 3D models. Therefore, image mosaicing approach is developed to make a panoramic image, so that 3D models can be rendered. To achieve these goals, additional sub-objectives are determined as follows:

Sub objectives

Generating a mosaic image of a grove scene

- Develop a feature extracting method customized for a grove scene, including image enhancements.
- Develop a feature tracking method which is robust against a parallax problem. Features tracked contribute to both a determination of the optical flow and registration for mosaicing.
- Generate a wide panorama image of a grove scene, including blending, which can be useful for a scouting vehicle system.

3D canopy models

Reconstruct the surface of an actual canopy based on empirical measurements taken from both a vision system and a range sensor system. Motion models of the vision system and the range sensor system are associated with a scouting vehicle system and a harvesting robot system, respectively.

CHAPTER 4
METHODS AND PROCEDURES

The objective of this chapter is to describe the technical procedures to meet each sub-objective.

Introduction

Before describing technical procedures, it is necessarily to review application scenarios and limitations under which techniques were designed. My study can be applied to two different agricultural applications in a citrus grove: 1) scouting system and 2) a harvesting system. Each scenario is briefly discussed in following sub-sections.

Application Scenarios

Scouting autonomous vehicle system

It is not important what type of vehicle is used in terms of gathering information. A frontal steering type shown in Figure 4-1 is assumed in my study. When a frontal steering vehicle tries to change directions, the rear axle has the least variation in coordinate position with respect to the lateral side of body. Therefore, it is assumed that sensors such as cameras or range sensors will be mounted on near the rear axle of vehicle.

As the vehicle gathers information from the canopy, moving along alleyways in a grove, it is assumed that the vehicle moves in a straight line motion and the camera is fixed facing the canopy in direction Z, orthogonal to the moving direction X as shown in Figure 4-1 B). Therefore, it is also assumed that lateral variations are ignored even though the vehicle may oscillate upward and downward.

Harvesting robot

It is not critical what type of robot is used for harvesting. An R1207 which has a series of axial-rotary joints is used for my study as shown in Figure 4-2 A). Various devices such as a

gripper or sensors can be mounted on the end-effector of robot. The mounting position of these devices on the end-effector is not important, as long as all extrinsic parameters are properly accounted for.

A robot gathers images of the canopy, passing through various positions around the canopy as shown in Figure 4-2 B). It is assumed that the base trailer, which supports a robot system, is stationary while the robot is acquiring data. All techniques in my study were designed by assuming following environments and conditions.

Assumptions and Limitations

Citrus canopy scene

Only citrus canopies and a citrus grove are considered as a target object. It is also assumed that there are only stationary citrus leaves without any artificial structures or moving objects in the scene. Other severe outdoor conditions such as direct backlights and windy conditions are also ignored.

Video frame image

Image quality does not matter as long as features are discernable. However, image filtering techniques were designed for the RGB color images to enhance feature extraction. Since most image processing techniques developed in my study are designed for sequential images and implementations are considered to work in real-time applications, it is assumed that vision systems can record video clips.

Single vision system

Since a reconstruction associated with vision systems are typically carried out using on a two-view geometry, where two or more images are required. As a result, stereo vision systems are widely used for a reconstruction. Since a stereo vision system can carry out reconstruction under stationary conditions, it could be appropriate for mobile vehicles scouting. The

performance of reconstruction using a stereo vision system is strongly influenced by our ability to match image features. To insure high performance in matching, both cameras must be able to provide similar quality images. When considering robotic manipulator applications, one must consider whether the weight, size or volume of the stereo camera system will hinder robot maneuverability. As the robot harvest the fruit, it will continuously be moving in and out of the canopy. Therefore, a light weight low profile sensor systems is the best. Therefore, my study uses a single camera vision system. Parts of frames are saved as base frames to get multiple views during exploring. Base frames are used until correspondences remain enough between frames.

Motion model and large objects than view

Since alleyways in an ordinary citrus grove are very long with respect to camera field of view, a single camera cannot capture the whole scene with sufficient detail. Therefore, we need to develop techniques that can reconstruct the whole canopy from successive images. Most literature which deals with 3D reconstruction has demonstrated their results in scenes where the whole objects are shown in every view. In this case, there is no problem to reconstruct objects with successful matching features. However, in the case of a canopy scene, since initial features extracted from the base frame are supposed to disappear at other frames taken from different views, it is necessary to continuously track features along the frames. This is especially true for the base features, which are used for the estimation of extrinsic camera parameters, must be propagated precisely.

Methods and Procedures

The process using an image can be briefly categorized into two parts: image processing and geometric calculation. Image processing deals with features in three steps: Feature detection, Feature tracking and Feature matching.

A feature detection algorithm (Chap.5) extracts meaningful features from an image obtained from vision systems. Feature tracking (Chap.6) monitors and retains the change of feature position in the image coordinates during sequential images. It is also used to measure optical flow. Feature matching is used to match indices of features between images. Once an index of features is arranged, then geometric applications such as image mosaicing (Chap.7) or 3D reconstruction (Chap.8) can be conducted. Sequential image mosaicing creates a wide panoramic image by means of stitching sequential images based on the optical flow. The wide image can be used for texture mapping images on virtual canopy models. A 3D reconstruction of a canopy surface is carried out with correspondences from feature matching. In addition, alternative 3D reconstruction using a LADAR & GPS (Chap.9) is conducted. The relationship among these procedures is shown in Figure 4-3. The 3D reconstruction can then be implemented to work with the harvesting robot model, and sequential image mosaicing and 3D reconstruction based on a LADAR & GPS for the autonomous scouting vehicle model.

Conclusions

This chapter has briefly reviewed the applications and assumptions considered in my work. Technical details of these topics and conclusions are discussed in next all chapters.

Figure 4-1. Vehicle system. A) Scouting autonomous vehicle (Subramanian, V., 2006). B) Motion model of vehicle.

Figure 4-2. Robotics sytem. A) R1207 robot manipulator. B) Motion model of robot.

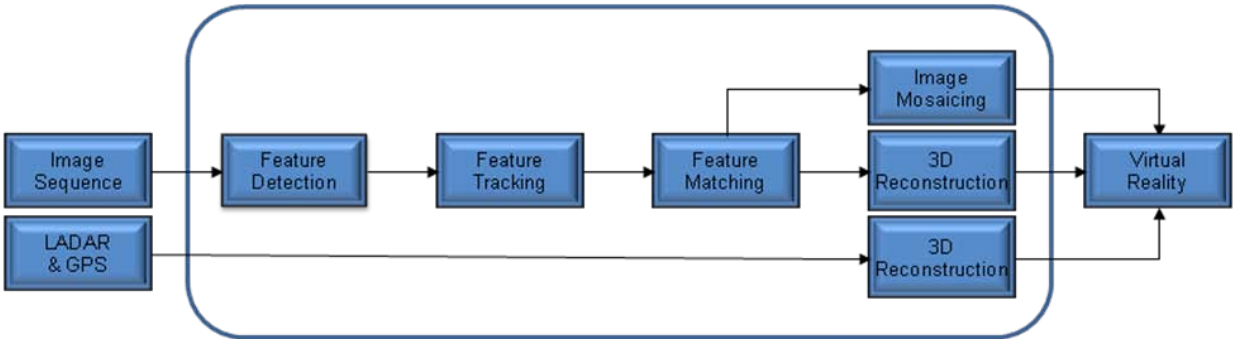


Figure 4-3. Overall procedures.

CHAPTER 5
FEATURE DETECTION

The objective of this chapter is extracting features from canopy scenes.

Introduction

It is more difficult to extract features from a natural landmark scene than an artificial well-structured scene. Most feature-based image processing methods have tried to find and extract features from salient artificial landmarks such as buildings, roads, etc. Natural landmarks like trees were considered more difficult objects. In my study, it cannot be assumed that there will always be artificial landmarks in a grove scene. Consequently, feature detection should be able to extract features from a canopy scene which consists only of leaves or fruit. A ripe piece of fruit could be a good feature because it can be segmented by color. However, image processing in my study cannot depend solely on harvest seasons.

Methods and Procedures

In my study, a 'feature' denotes a point feature. A corner is a feature used commonly because it is invariant to directions. A Harris corner detection method is basically used for feature detection. Initial feature points used for tracking are selected by using Harris corner detection. Harris corner detection used here allows us to set the number of corners and the minimum distance among corner points. The output of this detector has different results based on the region of interest because its output depends on a threshold which has to be applied in a gradient based on the given image. However, the image quality influences the corner detection directly. Therefore, it is necessary to maintain uniform image quality through the use of image filters.

Image Enhancement

Change in illumination which occurs outdoors influences the uniformity of image quality. Ordinary cameras are designed to adjust brightness and contrast automatically. In the case of an image obtained under strong sunlight or backlight, the contrast of the image becomes strong, or the histogram distribution is prone to be biased. Therefore, features may not be detected uniformly or sufficiently.

Image filtering is used to equalize and enhance image quality acquired from a camera. To make feature extraction clear, several image enhancement techniques such as histogram equalization and sharpening can be applied. The simplest way to be invariant to illumination is to convert RGB into HSV color space, which is a linear transformation. The other way is to equalize histogram distribution. Figure 5-1 shows that the shadowed area becomes enhanced though histogram equalization.

Another method is to use an embossing filter which regulates contrast and emphasizes edges. An example of the effect of the embossing filter is shown in Figure 5-2. Figure 5-2 A) shows corners detected from a backlit image. Few corners were detected in the shadowed area. On the other side, corners were detected uniformly from an embossing filtered image as shown in Figure 5-2 B). Results from the optical flow based on the corner detection were shown in Figure 5-2 C) and D). Figure 5-2 D) shows that an embossing filter gives more chances to detect features in a backlit image. These filtering combinations are helpful to keep a uniform image even with backlit scenes.

Leaf Detection

As long as a camera moves, features are prone to disappear. Therefore, robust features are needed that are able to withstand this disappearance. The aim of this approach is to find out the best combination of morphological operations at the initial step so that the best feature point is

chosen and it will last as long as possible. Features such as the tip and junction of a leaf could be promising features, but it seems that it is not only unstable but also expensive to calculate during the sequence.

After enhancing image quality, an opening operation is applied, which is a combination of erosion and dilation. An opening operation can be expressed with

$$A \circ B = (A \otimes B) \oplus B, \tag{5-1}$$

where $\otimes$ and $\oplus$ are an erosion and a dilation operation, respectively. An opening operation has the effect of simplifying complex textures into a simple level of intensity as shown in Figure 5-3 B). From there, a watershed segmentation is applied with results illustrated in Figure 5-3 C). With a large mask for opening operation, inadequately small segment can be avoided.

Even though segmented, all patches might not be features. Since a patch in each segment has a different level of intensity from other patches, shadowed leaves could disappear easily. To filter patches by suitableness, a threshold is applied to the patches. Figure 5-4 illustrates an intensity patch in a segment. The threshold increases from zero until an area of the binary image becomes a quarter of the area of the segment. When it reaches this condition, the center point of the binary image is recorded. This algorithm is also implemented automatically to ignore thinner segments by computing the ratio of area.

Once a patch over the threshold is selected, compute the center of the patch as shown in Figure 5-5 A). At this moment, to reflect the actual image, another opening operation is used again which has smaller mask like Figure 5-5 B). After the threshold is overcome and the segments are selected, considering all of the patches, Figure 5-5 C) is finally obtained. These feature points provide the initial position for feature matching. In summary, the entire procedure is shown in Figure 5-6.

Results

Fruit Detection

Even though leaf detection is designed to detect the center of leaves, it can also detect the center of fruit. Figure 5-7 A) and B) shows the result of using a ripe citrus fruits scene and an unripe citrus fruit scene, respectively. Most fruits are pointed regardless of their color. There is a problem separating fruits and leaves in terms of fruit detection. However, it would be useful if fruit candidates were chosen regardless of color because the color-based fruit detection approach cannot be applicable to green fruit such as limes. Counted number of fruit was shown in Table 5-1. Mature fruit was detected more than immature fruit.

Size Problem

Since a segment size cannot fit to a leaf size automatically, it is necessary to adjust the size factor appropriately depending on the image status. Figure 5-8 shows the results of applying inappropriate size factors. It is not necessary to fit a segment to each leaf. However, segmentation could vary and features may be unstable.

Conclusions

Outdoor scenes are exposed to unbalanced illumination such as shadow or backlight. Therefore, it is necessary to apply image filters in order to enhance image quality. An embossing filter detects and shows features uniformly.

A leaf detection method was devised to provide more stable features than corner detection. The key point of leaf detection is to segment leaf areas through the combination of morphological operations. Once segmented, the center of a leaf in each segment is set as a feature point. Leaf detection was designed to apply discriminating thresholds to each segment to secure as many features as possible. However, it is not used in every frame. Feature detection is

carried out only when new features are needed because the detection may not always give the same points repeatedly.

Future Work

A segment size is a critical factor in leaf detection. Therefore, it is necessary to develop a method to adjust a segment size automatically. It is expected to find out appropriate factors by varying the size of segment iteratively.

Table 5-1. Comparison detected number of fruit.

	Figure 5-7 A)	Figure 5-7 B)
Number of fruit	15	14
Detected number of fruit	13	8
Detection rate	86.6%	57.14%

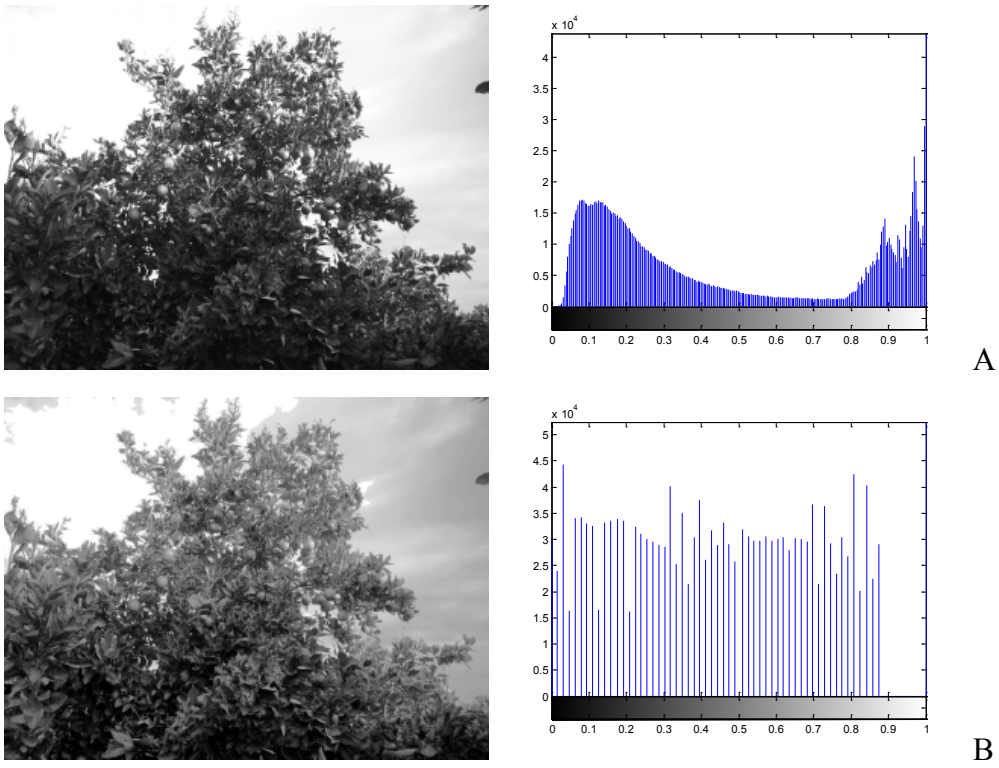


Figure 5-1. Histogram equalization. A) Original image and its histogram . B) Equalized image and its histogram.

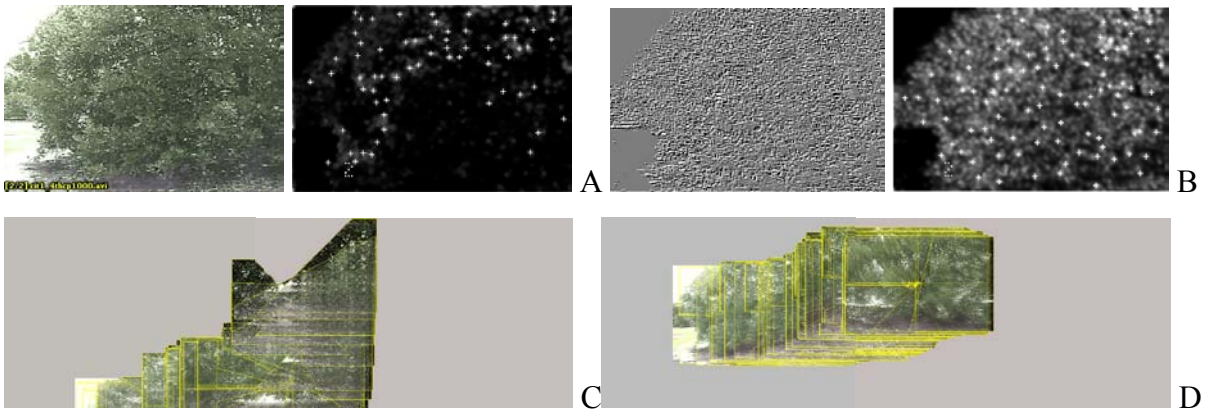


Figure 5-2. Embossing filter effect. A) Detected features with an original image and. B) Detected features with an embossing filtered image and a mosaicing image based on the optical flow. C) Mosaicing image based on the features from an original image. D) Mosaicing image based on the features from an embossing filtered image.

Figure 5-3. Leaf segmentation. A) Original Image. B) An opening with large mask applied to the image. C) A watershed segmentation applied to the image.

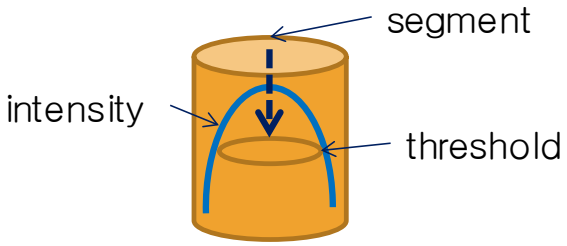


Figure 5-4. Threshold adjustment.

Figure 5-5. Detect the center of segment. A) Segmented patch. B) Image applied an opening with small mask. C) Detected feature points.

Image → Sharpness → Opening (Large Mask) → Watershed Segmentation → Selecting segment by threshold → Opening (Small Mask) → compute Center of segment.

Figure 5-6. Morphological operations of leaf detection.

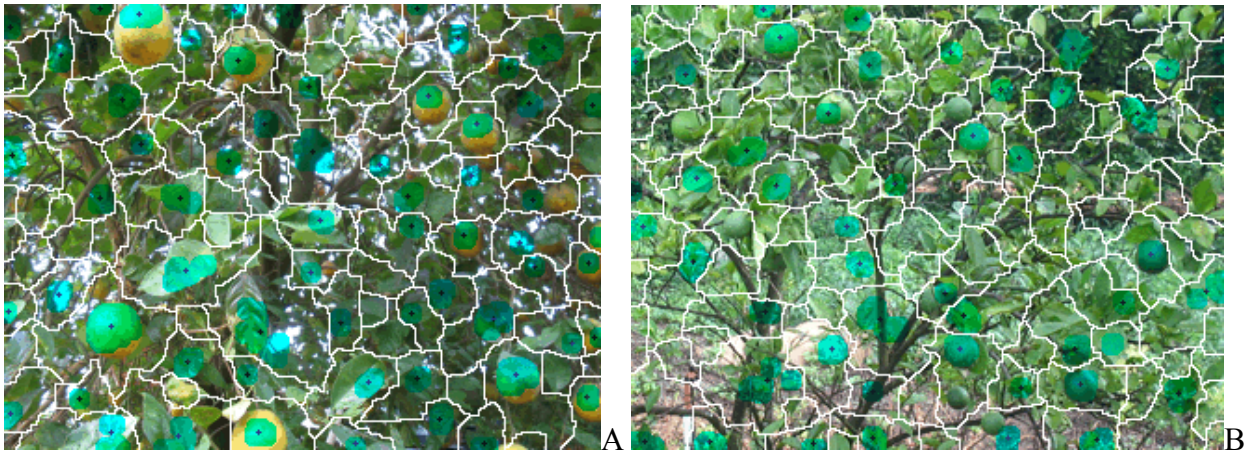


Figure 5-7. Fruit segmentation. Fruit segmentation A) with a ripe fruit image, B) with an unripe fruit image.

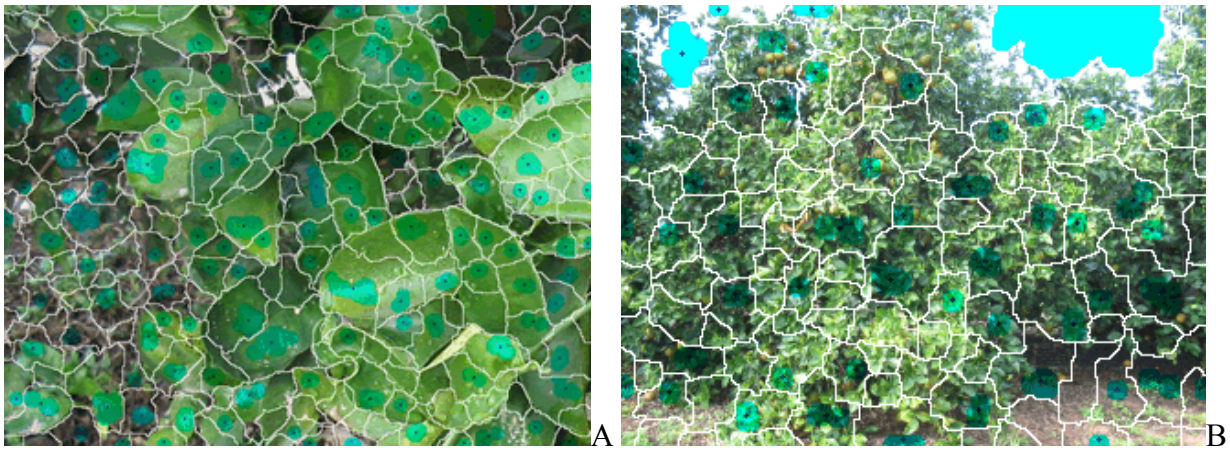


Figure 5-8. Segmentation size. Fruit segmentation A) with a zoomed-in scene. B) with a zoomed-out scene.

CHAPTER 6
FEATURE TRACKING

The objective of this chapter is tracking features between frames.

Introduction

Template images obtained near every feature point are used to compare how the feature points are moved during tracking features. Tracking is a process to measure the motion vector of a feature point. Although a great number of methods have been suggested, existing methods that extract feature points based on specific images have limitations that apply to various types of images in general. An ambiguous scene or occlusion can cause tracking failure. To keep tracking robustly, an active mesh method was used in my study. The basic concept of active mesh in terms of tracking is that the tracking does not depend only on the image, but also focuses on the geometric relationship among connected feature points. This approach makes it more robust to track in the case that only the camera is moving. Since features in outdoor scenes are exposed to a variety of disturbances, they are prone to blink or disappear as shown in Figure 6-1. When they lose their capacity as a feature, new features must be detected again. Such short-lived features lead to accumulated errors. Therefore, it is necessary for features to survive as long as possible.

Methods and Procedures

Mesh-based tracking is an extended application of an active contour. An active contour defines linear relationships between nodes while an active mesh takes care of multiple connections. Meshes can fit to gradual deformations and are regarded as the surface of an object. Blinking features which repeatedly appear and disappear can be sustained through developing these meshes. Molly (2001) proposed a basic idea about an active mesh method and suggested various potential features to be forces. In my study, part of the equations and features were employed for force equations.

Active Mesh

In order to be robust to sudden changes of features, it is necessary to reduce the effects of individual features. An active mesh considers the movement of overall features by connecting feature points. To define the relationship with neighboring feature points (nodes), meshes are generated by Delaunay triangulation. Optical flow is regarded as a force vector in this scenario. The new position $P(x, y)$ of feature i at frame k_{th} is estimated as follows:

$$P_i^{(k)} = P_i^{(k-1)} + \alpha S_i, \tag{6-1}$$

where S_i is the sum of the resultant forces and α is the weight factor.

$$S_i = \sum_{j=1}^{n} F_{i,j}^{(k)}, \tag{6-2}$$

$$\alpha = \frac{\left\| F_{i,j}^{(k)} - F_{i,j}^{(k-1)} \right\|}{\|S\|}, \tag{6-3}$$

where $F_{i,j}^{(k)}$ which is a resultant force of neighboring node j at frame k_{th}.

$$F_{i,j}^{(k)} = aF_{j,External} + bF_{j,Internal}, \tag{6-4}$$

where $F_{j,External}$ is affected by image intensity, and $F_{j,Internal}$ is based on the relationship between nodes. Fixed ratios a and b are adjusted experimentally (a=0.85, b=0.15) with the sum of resultant force S_i reflected on the center feature $P_i^{(k)}$ as shown Figure 6-2.

External force

External forces $F_{j,External}$ are defined by the difference of neighboring node P_j between frames k_{th} and $k_{th}-1$.

$$F_{j,External} = \beta_j^{(k)}\left(\hat{P}_j^{(k)} - P_j^{(k-1)}\right), \tag{6-5}$$

where $\beta_j^{(k)}$ is the weight factor of neighboring node j at frame k_{th}. The variable β is the inverse relationship determining that the farther the node, the less influential it will be on the calculations. However, for this to be a reasonable assumption, the elevation difference between the center and neighbor node cannot be too large.

$$\beta_j^{(k)} = 1 - \frac{d_j^{(k)}}{\sum_{l=1}^{n} d_l^{(k)}}, \tag{6-6}$$

$$d_j^{(k)} = \left| \hat{P}_j^{(k)} - \hat{P}_i^{(k)} \right|, \tag{6-7}$$

where $d_j^{(k)}$ is distance between current node $\hat{P}_i^{(k)}$ and neighboring node $\hat{P}_j^{(k)}$. Tracked features give us the degree of correlation and the position $\hat{P}^{(k)}$.

Internal force

Internal forces $F_{j,Internal}$ are defined by an elastic coefficient which represents the relationship between feature points.

$$F_{j,Internal} = \gamma_j^{(k)} \left(\hat{P}_j^{(k)} - \hat{P}_i^{(k)} \right), \tag{6-8}$$

where $\gamma_j^{(k)}$ is the elastic coefficient which is a ratio of difference between previous $d_j^{(k-1)}$ and current distance $d_j^{(k)}$ to distance $d_j^{(k)}$.

$$\gamma_j^{(k)} = \frac{d_j^{(k-1)} - d_j^{(k)}}{d_j^{(k)}}. \tag{6-9}$$

Internal force not only makes the calculations less sensitive to the high frequency motion, but sustains the previous force when features disappear temporarily.

Feature management

Features are expected to disappear when the camera moves, some features, go out of the field of view boundary, while others disappear due to occlusion or are overlapped with other

features. These features should be deleted immediately so that new feature points can be added to maintain the total number of points. New features are selected among candidates that are the farthest from the existing feature points.

Simulation Results

The performance of the active mesh was examined using a virtual object and 2D sinusoidal camera motion. The testing code was implemented with MATLAB. While the camera moves, some features are intentionally discarded and overall optical flow is estimated.

When absent features are ignored, the motion estimation appears turbulent. On the other hand, active mesh tracking follows original motion with sustainable features as shown in Figure 6-4. Each axis indicates 2D motion coordinates in the plot. When an occlusion of over 10% occurs, the estimation diverges as illustrated in Figure 6-4 B) because too many occlusions violate the assumption of an active mesh. Figure 6-4 C) shows that an active mesh estimates the input optical flow better than a normal estimation. This experiment examined only the effect of occlusion, and does not represent local accuracy.

Multilayered Active Tree

An active mesh method has several disadvantages. Because it searches all nodes exhaustively, computation speed is slow. Furthermore, wrong estimation can still influence normal neighboring nodes. To enhance the shortcomings of an active mesh method, a multilayered active tree method was invented in my study. Multilayered active tree expanding from a 2D mesh to a 3D hierarchical structure as the active mesh extended from an 1D line to a 2D mesh.

Generation hierarchical structure

Scale invariant methods use coarse-to-fine multiple images referred to as pyramids. The basic idea of a multilayered active tree is to connect nodes between multiple images. First,

multiple coarse-to-fine images are prepared by applying Gaussian filtering as shown in Figure 6-5 A). Feature detection is carried out at each image as shown in Figure 6-5 B). Then a Voronoi diagram is applied based on features at each layer as shown in Figure 6-6 C). Features within a segment of an upper layer are grouped and belong to the segment as shown in Figure 6-7. This procedure is repeatedly conducted for every segment and every layer. Therefore, this loop can be implemented by a recursive function as show in Figure 6-8.

Features form a hierarchical structure which is automatically generated as shown in Figure 6-9. Since exploring is based on hierarchical paths, it is faster and less influenced by neighboring nodes than an exhaustive active mesh. Common features between layers are regarded as parent nodes. Features detected in the coarse image can be regarded as dominant features. Figure 6-10 shows dominant features.

External force and internal force

Once the hierarchical structure is built, the active mesh method is applied at each layer. Basic formulas are the same as those of an active mesh; however, the exploring path is different. An active mesh considers neighboring nodes, while a multilayered active tree considers subnodes. Figure 6-11 shows the difference between an active mesh and a multilayered active tree in terms of forces. Thick arrows in Figure 6-11 B) denote a large weight factor. Since all nodes in an active mesh are equally handled, unnecessary evaluations are repeatedly conducted on the same node. In addition, some nodes that are erroneously estimated can influence other nodes. On the other hand, because it is possible to assign discriminating weight factors on dominant or high level nodes in a multilayered active tree, the effect of unstable nodes can be controlled. Figure 6-12 shows how forces are applied to sub nodes.

In this example, 3rd node is a top dominant node. Computation starts from the top node. Once new top nodes (N_3) are determined by an external force, they become a reference (R_3) for

their subnodes. New nub nodes (N_2) are determined by both an external force and an internal force with altering weight factors. In the equation (6-4), weight factors are determined by an angle between optical flow vectors of upper and lower nodes. The sum of weight factors is equal to 1. When a noise feature makes the difference, weight factors enable to refer to the internal forces more than the external force.

Simulation and results

This experiment tests a multilayered active tree's robustness against to the effect of noise and occlusions by means of estimating the optical flow. The camera motion model was the same with the active mesh. Three layers were generated for the hierarchical structure and 19 feature frames were used. It was assumed that features were consistently detected for the noise experiment. An image obtained through a Gaussian filter possesses the relatively small number of features yet the features are prone to be less sensitive to noise. Therefore, smaller noises are assigned to higher nodes. Noises are added to all frames with random variation. Figure 6-14 shows the result in optical flow estimation. Since noise could influence neighboring nodes in the active mesh, errors could be accumulated as process goes. However, in the multilayered active tree, since the lower node's noise does not affect neighboring nodes or higher nodes, errors were not accumulated as shown in Figure 6-14 A). Figure 6-14 B) shows that the norm error of a multilayered active tree was smaller than that of noise-added features.

An occlusion case was simulated by randomly removing some nodes in a certain frame. A multilayered active tree holds occluded nodes during occlusion frames. Figure 6-15 shows that occlusion did not affect its optical flow estimation after occlusion occurs. Therefore, multilayered active tree tracking method can be regarded less sensitive to the sudden change of optical flow.

Conclusions

To be unaffected by sudden changes in feature points, an active mesh feature tracking method was employed. Basically, an active mesh consists of two forces, an external force and an internal force. The external force refers to feature points with a weight factor, and the internal force is determined by an elastic coefficient. The ratio between forces is experimentally adjusted. A simulation for occlusion was conducted with a virtual object and a sinusoidal camera motion. The effect of occlusion was tested by estimating the overall optical flow with different number of absent feature points. The results in simulation showed that it is effective with a small amount of occlusions such as blinking and disappearing features.

However, even though an active mesh is resistant to the sudden changes, it requires exhaustive computation and it is possible for erroneous features to disturb normal features. Therefore, a multilayered active tree method was invented, which expands a 2D mesh into a 3D mesh. The creation of multiple layers is based on coarse-to-fine images by means of Gaussian filtering. Features extracted from each image recursively form hierarchical structures by connecting sub-features within an area which is segmented by a Voronoi diagram. Since multilayered meshes can evaluate the strength of features, it can control the disturbance from unnecessary features. Subsistence of features was evaluated with the comparison of optical flow. The optical flow through these methods resulted in resistance to noise. The computation speed of was faster over ten times than that of an active mesh.

Future Work

The next step is to apply the algorithm to actual sequential images. Since this algorithm works with several assumptions, it is necessary to verify how appropriate those assumptions are on the actual sequential images. To compare the accuracy of optical flow estimation, the camera motion must be known.

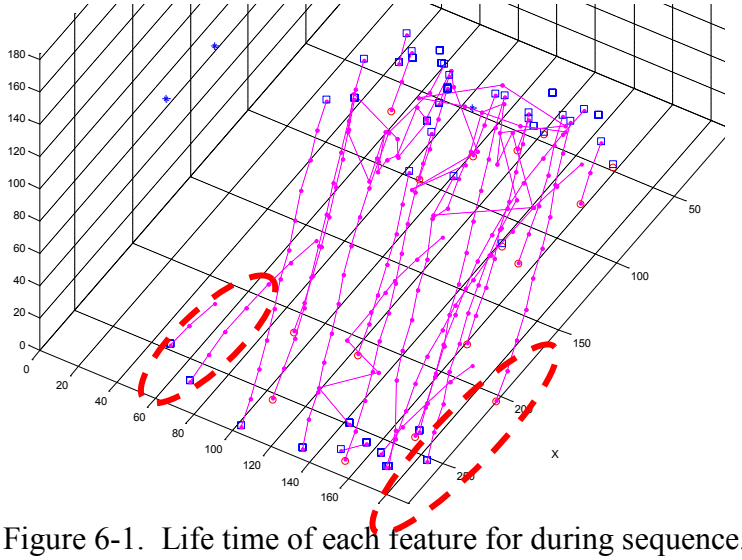


Figure 6-1. Life time of each feature for during sequence.

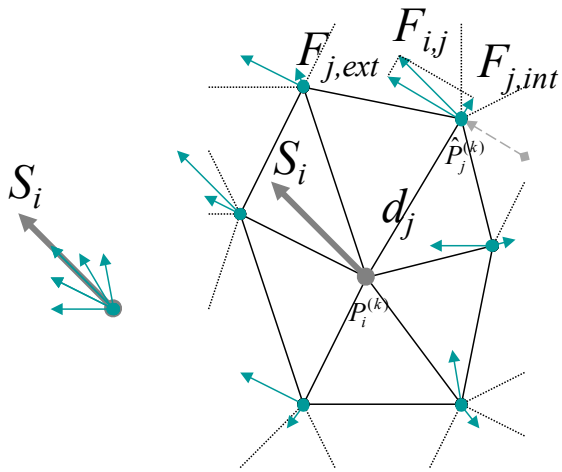


Figure 6-2. Sum of forces acting on each node.

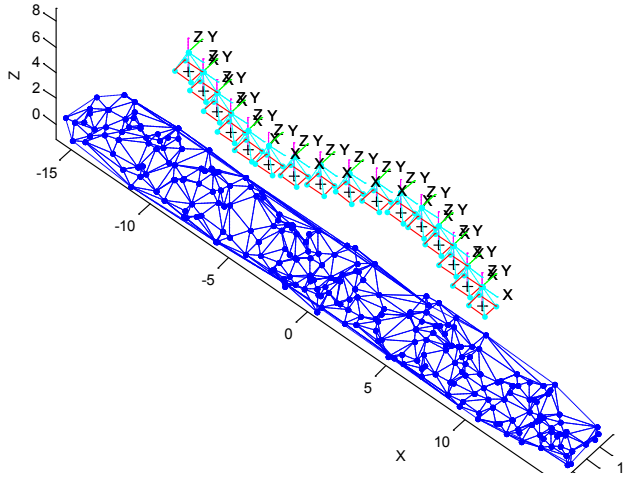


Figure 6-3. Virtual objects and 2D sinusoidal camera motion.

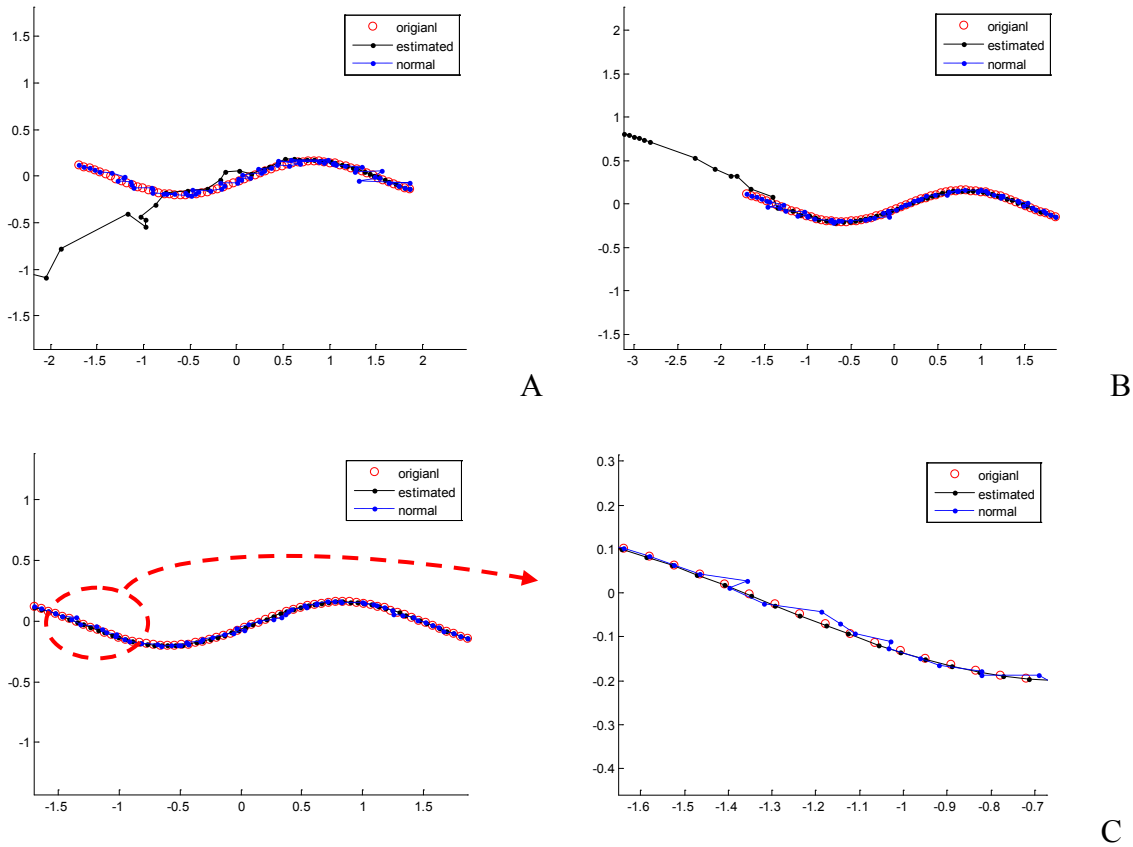


Figure 6-4. Optical flow estimation. A) 30.54% occlusion (5025/16384), B) 18.99% occlusion (3112/16384), C) 10.53% occlusion (1725/16384).

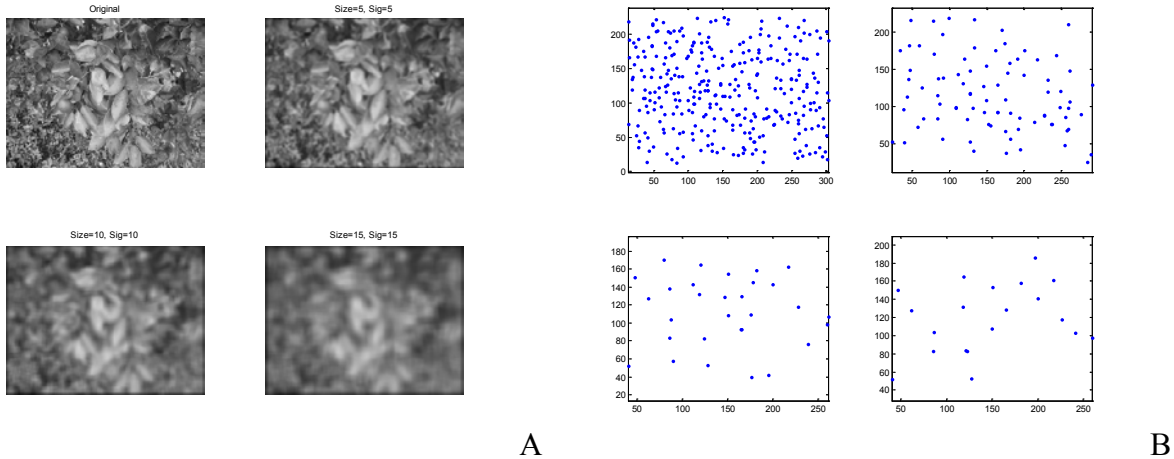


Figure 6-5. Nodes generation from multiple images. A) Generated coarse images using Gaussian filtering. B) Detected corners based on each image.

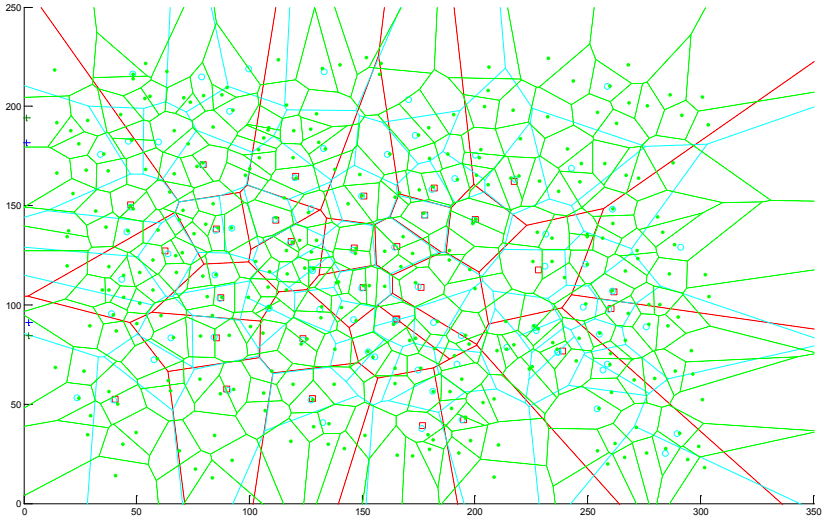


Figure 6-6. Voronoi segmentation applied to each image.

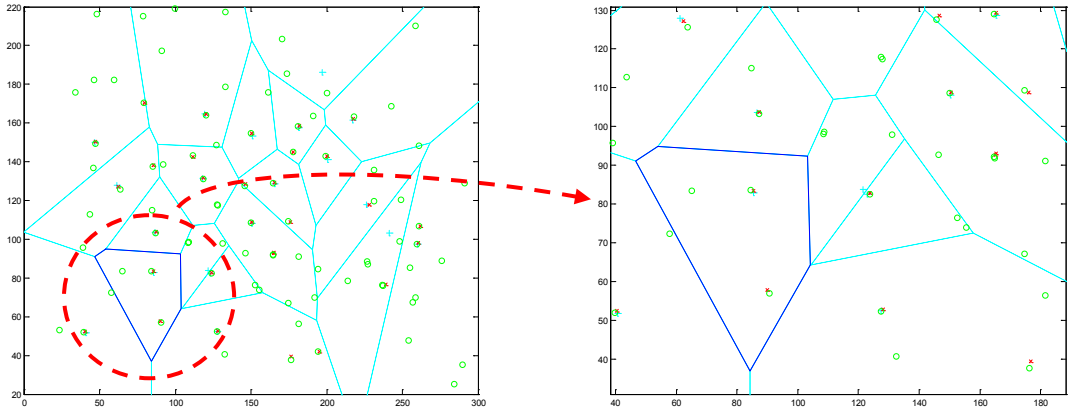


Figure 6-7. Grouped features in a segment of an upper layer.

```
function [ Nds_r ]= nds_vpts ( vPts_, lv_, pIdx_ )
if lv_<2 return; end;
vPt = vPts_{lv_}; vPt_1 = vPts_{lv_-1};
for i=1:size(pIdx_,2)
    Pt_ = vPt_1.Pt;
    polyPt_ = vPt.vPts{pIdx_(i)};
    pIdx  = inpolygon(Pt_(1,:),Pt_(2,:), polyPt_(1,:),polyPt_(2,:));
    pPt = Pt_(:,Idx_r);
    if isempty(pPt) continue; end;
    Idx = find(pIdx==1);
    Nds_r{i}.Idx = Idx;
    Nds_r{i}.Pt = pPt;
    Nds_r{i}.Nd = nds_vpts ( vPts_, lv_-1, Idx ); % recursive calling
End
```

Figure 6-8. MATLAB code of recursive grouping function.

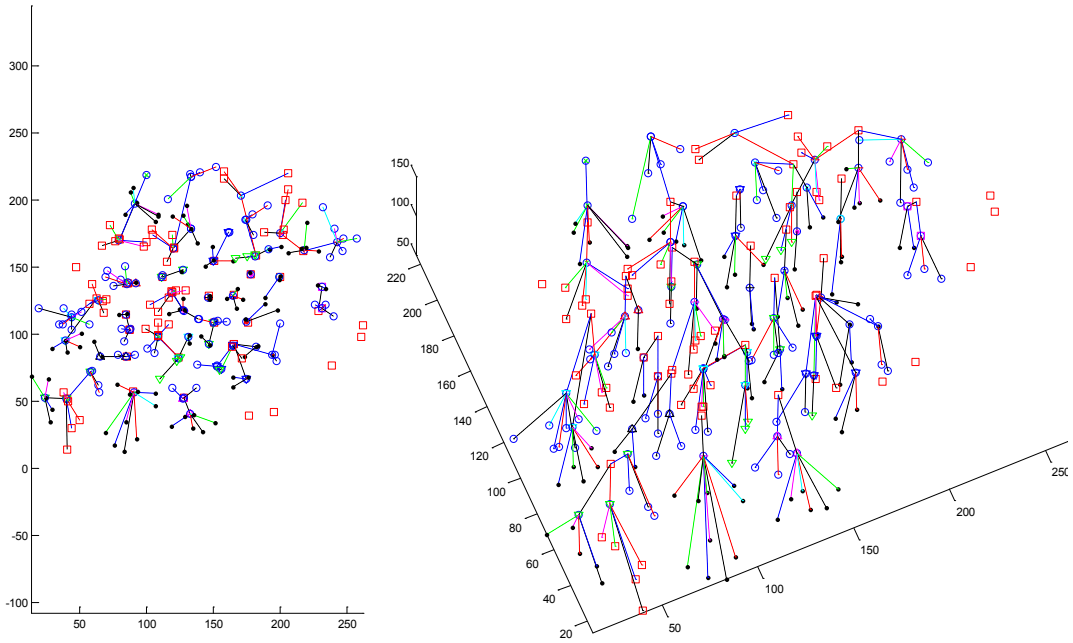


Figure 6-9. Automatically generated hierarchical connection based on an image.

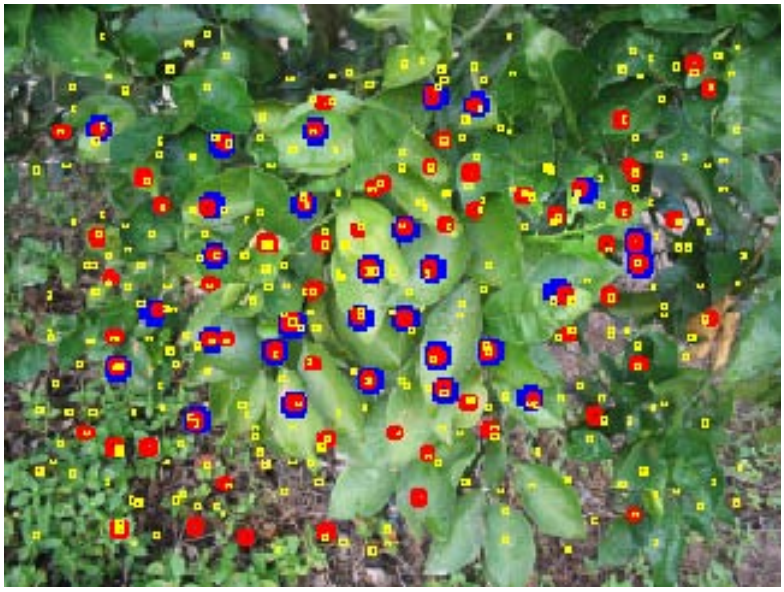


Figure 6-10. Determined dominant features.

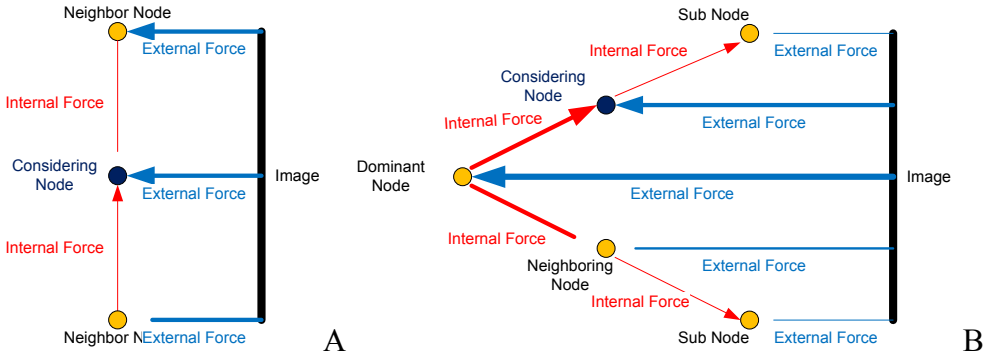


Figure 6-11. Forces acting on the considering node. A) Active mesh. B) Multilayered active tree.

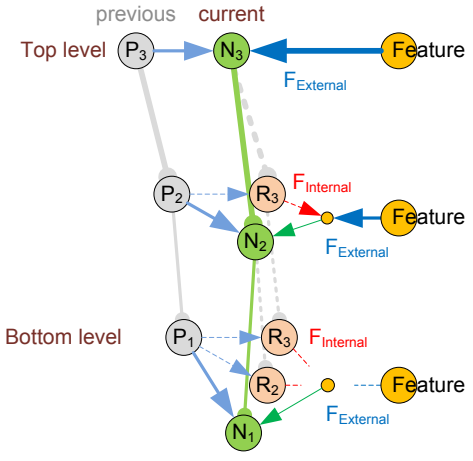


Figure 6-12. Procedure to determine subnodes.

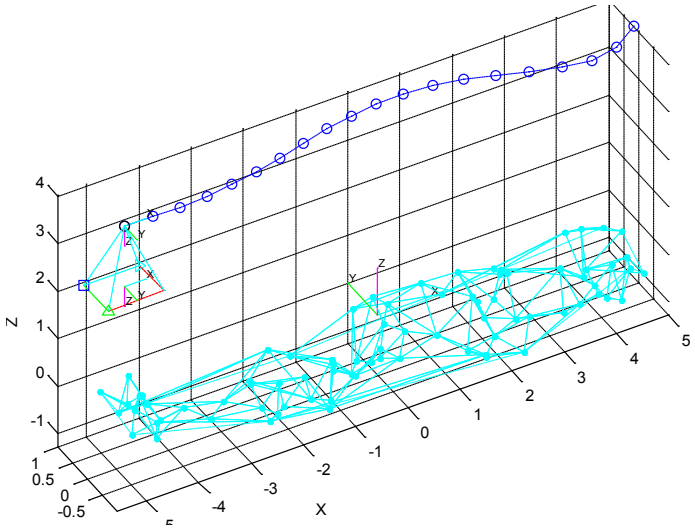


Figure 6-13. Camera motion model in the test for multilayered active tree.

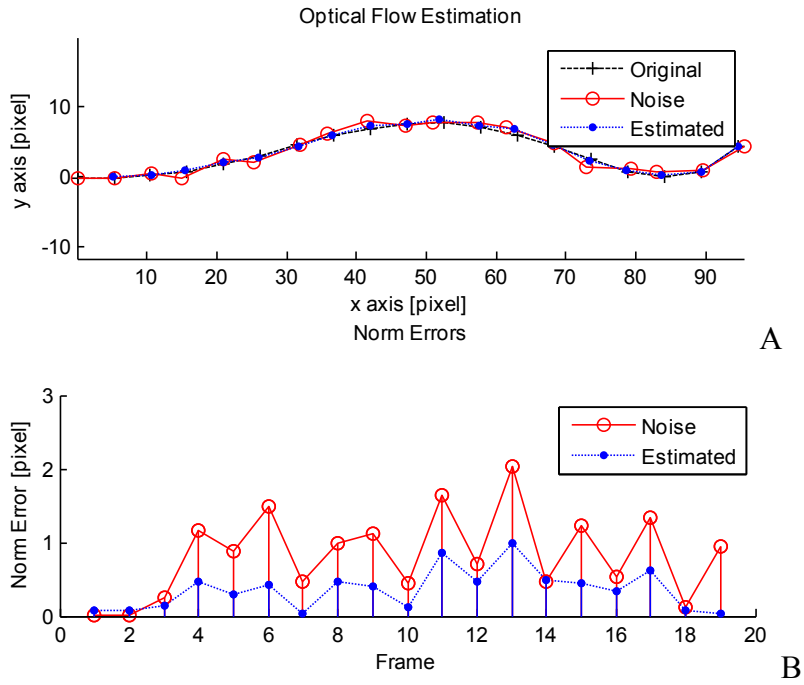


Figure 6-14. Simulation for robustness to noise. A) Optical flow esitmations. B) Norm errors between noisy optical flow and compensated optical flow.

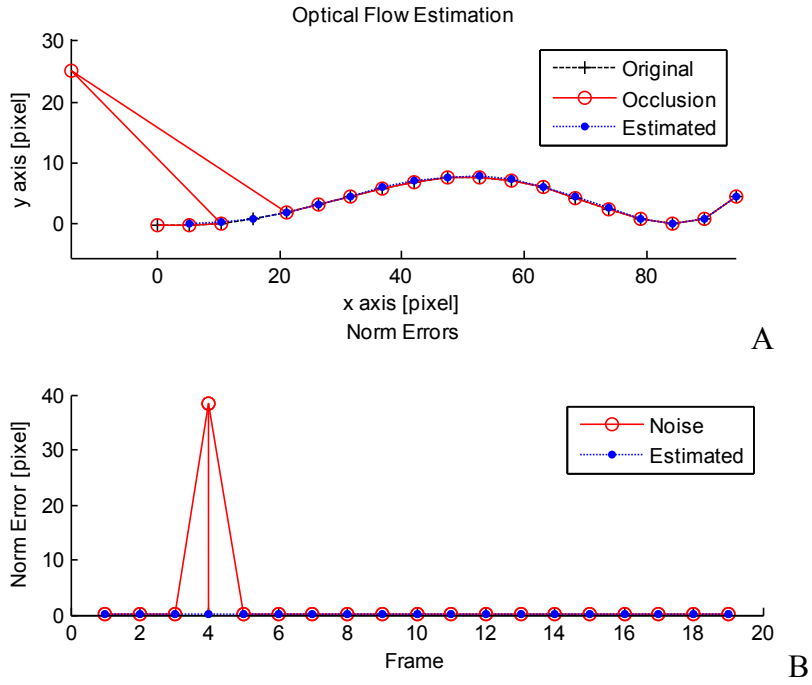


Figure 6-15. Simulation for robustness to occlusion. A) Optical flow esitmations. B) Norm errors between occluded optical flow and compensated optical flow.

CHAPTER 7
SEQUENTIAL IMAGE MOSAICING

The objective of this chapter is to create a panoramic image with sequential images.

Introduction

A ground-based scouting system which inspects growing states of plants or detects spreading diseases early can be one vision-based agricultural application associated with unmanned management. An aerial-based or satellite-based scouting system takes only the top view of canopies, whereas a ground-based scouting system can take the lateral view of canopies closer. Therefore, ground-based scouting systems are a more effective way to gather indispensable information of canopies.

Machine vision systems are also popular devices used in remote sensing. However, since cameras have limited viewing angles, a huge number of images may be needed to cover a whole grove. To reduce saving capacities and increase the effectiveness, image mosaicing techniques are commonly used to stitch images taken from different viewpoints.

Most literature dealing with image mosaicing has assumed a planar scene in which the shapes of even non-planar objects are not changed. Unchanged shapes make it easy to stitch images together. This assumption can be achieved only when a camera is purely panning or is far enough away from objects as a satellite. Otherwise the mosaicing is prone to failure because some features can be occluded or deformed, which is called a parallax problem. In two images taken from different positions as shown in Figure 7-1, for instance, the right side of an image at the left position and the left side of the other image at the right position of a camera could be different. Maintaining shapes in the scene is important to find out correspondences. Since the deformation of a feature point occurs less between video frames, losing the correspondences can be minimized by tracking the feature points.

Methods and Procedures

A mosaicing procedure is briefly divided into six stages: Image Acquisition, Feature Detection, Feature Matching, Estimation Homography, Warping and Blending. The basic process of mosaicing is to determine and align correspondences between two given images. Therefore, those two images should have enough area that overlaps. Once correspondences are determined, the extrinsic parameters of a camera can be also estimated. Then warping and aligning target images are achieved based on a homography between the images. Figure 7-2 shows a typical image mosaicing procedure. The structure of the algorithm consists of four essential functions: Initialization, Filtering, Tracking and mosaicing. Figure 7-3 shows a flow chart of the algorithm.

Projection Model

Most image mosaicing methods conform to either spherical or cylindrical projection models. Both projection models have a single center position of a camera. The image mosaicing using these models is carried out by the pure rotation of a camera. Since there is no translational motion, a homography becomes simply a rotation matrix.

$$\mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T \rightarrow \mathbf{H} = \mathbf{R}, (\mathbf{T} = 0). \tag{7-1}$$

In the pure rotational motion, no parallax occurs with a non-planar scene. Therefore, warping and matching are conducted simply. However, these projection models are limited because they cannot be applied any longer if a camera moves far enough away.

In a ground-based scouting system, the center of a camera moves along a grove alleyway. Assuming that cameras are placed on the rear wheel axis of the vehicle and the path of cameras is almost a straight line, only variations in the X and Y axis can be considered without

perspective warping computations. Since there are small variations in the z axis between successive frames, the optical flow indicates camera motions directly.

Alignment

An alignment is a process to register images with a distance along the motion vector between successive images. An alignment approach in my work was designed to remove the partially overlapped area of an image in order to register it along an edge perpendicular to a line which connects the center of two images. Since a parallax may cause incongruity while mosaicing, this cropping edge approach tries to minimize the parallax by leaving a partial image around the center. This approach assumes that a parallax increases as a camera moves farther away from the center of an image plane as shown in Figure 7-5 A). This assumption can be applied only when the camera direction is orthogonal to lengthy objects. Therefore, the alignment can be approximated by cropping into a part of the image. Figures 7-5 B) and C) show the process that determines the edge cropped overlapped area. Only case A) was used within my work.

To remove the overlapped area along the cropped edge, an alpha channel was used. An alpha channel is a gray mask image which controls transparency between a given image and background. First, you look for intersecting points between frame edges and a line perpendicular to another line between the centers of frames. Once the cropping edge is determined, it should be decided which side would be removed. The partial area to be removed should be located between the centers. Then, you can make boundaries based on these edges and apply a floodfill function to the area on the alpha channel. By controlling the alpha channel, the overlapped area can be blended smoothly. Regardless of which direction the camera moves, the partial area between centers would be removed.

Filename Numbering

If the registered image gets too big, it may be a problem to process. Thus big images should be cut with a fixed size whenever it surpasses twice the regular size. Precisely cropped images can be joined together whenever we want without any additional computation later. The file names are named with an index number similar to a matrix. Figure 7-6 A) shows an example of the numbering of divided mosaic images.

Experiment

Configuration

To be compatible with coordinates of VRML later, a negative focal length was employed like that of OpenGL. Configuration is also shown in Figure 7-7 B). A vehicle to which a camera is mounted on the real axis moves along the alleyway in a grove. Figure 7-7 A) shows a situation when camera moves in a sinusoidal path. The variation along the Y axis is due to vibrations of a vehicle while moving. Parameters assumed for simulation are shown in Table 1.

Simulation

Before applying the algorithm to an actual grove scene, the estimated camera path was compared with a known input sinusoidal path as shown in Figure 7-8. The current coordinates are based on the previous frame image. Since the image coordinates are discrete integers, round-off errors accumulate with respect to the camera path as the process continues. Cumulative errors can be corrected by using other positional sensors such as a DGPS receiver.

This algorithm was examined with prerecorded video clips, rather than real-time video input. To observe the performance of this algorithm, it was compared with a panorama demo program built in video-processing toolbox of MATLAB. This video clip was taken from a camera moving horizontally rightward. To make it easy to track feature points, salient points were scattered artificially and a tetrahedron was adhered in order to form a non-planar object.

Since the panorama demo program leaves behind the left part of the image, Figure 7-9 A) shows that the resulting image is skewed favoring the right side of the object, whereas the developed algorithm leaves a direct frontal view as shown in Figure 7-9 B). Figure 7-10 shows the result when the position A) and direction B) of the camera moves with a sinusoid pattern. Notice how images have been cropped and stitched.

Results

Real grove scene video clips used in this experiment were recorded in the UF grove on May 23rd 2007. A camera was fixed on a tripod and a cart that feigns a vehicle, and pulled by hands with a speed of less than 1mile/h. Mosaicing results for the grove scene are shown in Figure 7-11 through Figure 7-13.

First, video clips were recorded far enough away from the citrus grove. In this case, the vibration of the vehicle had little effect. Since this scene can be regarded as a planar object, the mosaicing is conducted quite well. Figure 7-12 shows a mosaicing result which was recorded closer to the grove so that leaves can be observed in detail. This result looks blurred because of winds. It is very difficult to solve this problem without information about the global location of the vehicle. In my work, the wind-waving problem was not considered.

Another problem found was a gradual deviation. The mosaicing image is very lengthy. If the horizontal axis of an image and the direction of a camera motion are not parallel, the mosaicing image will be out of the upper or lower boundary before long as shown in Figure 7-13. Therefore, the angle of camera should be aligned horizontally before recording. It was designed for large size images to be automatically divided into fixed sizes when the width exceeded the twice width of the fixed size as shown in Figure 7-13.

Conclusion

In my work, video image mosaicing for a non-planar object was studied. The developed algorithm estimates the motion vector of the camera. When a camera moves parallel to objects, cropping edge approach can approximate mosaicing of even non-planar objects such as a citrus grove. This video mosaicing was achieved in spite of moving irregularly. This result shows that the cropping edge approach can be appropriate to mosaicing.

Any well-designed image processing algorithm does not always yield successful results in every case. Since the algorithm developed in my work assumes that successive images are similar, that is optical flow can be measured, it makes an unexpected error when excessive changes of view occur. A few errors can affect the success of the entire mosaicing. Therefore, it is most important to track feature points robustly and capture video stably. The successful result of image mosaicing depends on how accurately feature points are tracked. Since the image coordinates are discrete integers, the estimation errors in terms of camera path may be accumulated as the process goes.

A noticeable problem from the results is a deviation from the horizontal image axis. Figure 7-3 shows that the camera motion and the horizontal axis of an image are not parallelized. Since the mosaicked image is very lengthy, it may not be within the margins of the upper or lower boundaries after long unless the horizontal axis of the image and the direction of the camera motion are substantially parallel or corrected using some appropriated sensors.

Table 7-1. Intrinsic parameters for simulation.

Video image	Mosaic image	Focal length	Metric Conversion	Number of frame
320x240	640x480	-460	300 dpi	64

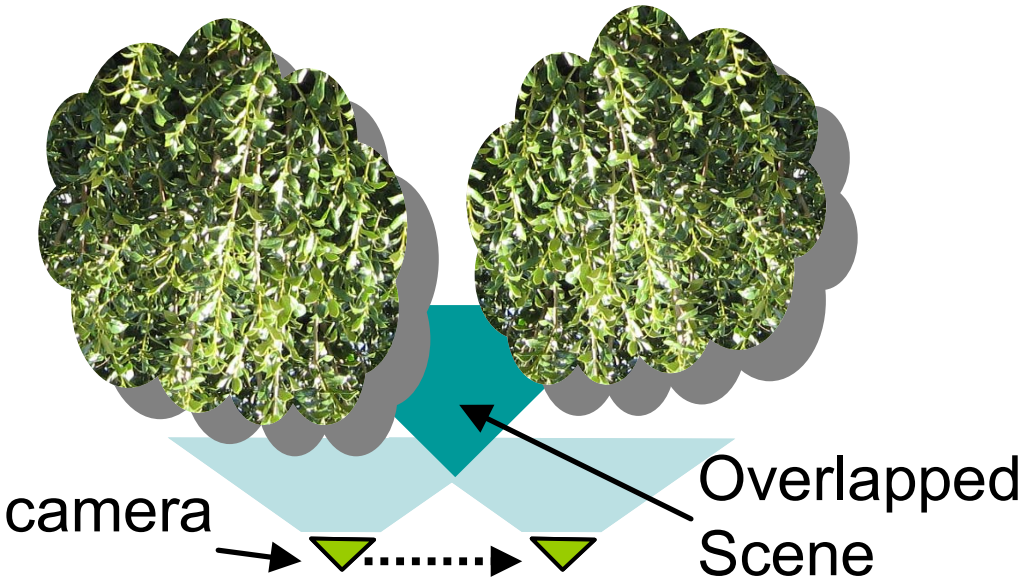


Figure 7-1. Parallax problem occurring with non-planar objects which have depth.

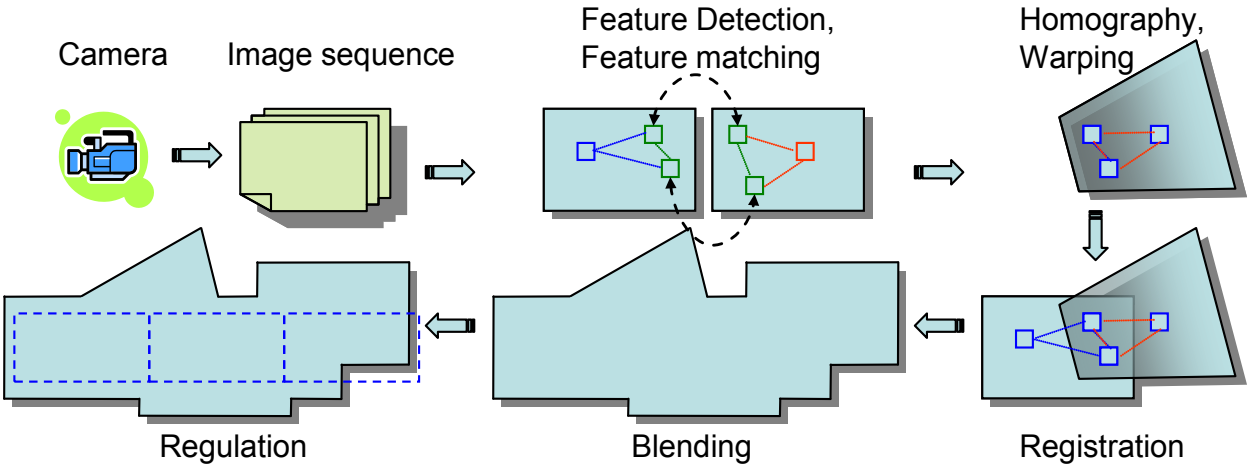


Figure 7-2. Basic steps of image mosaicing.

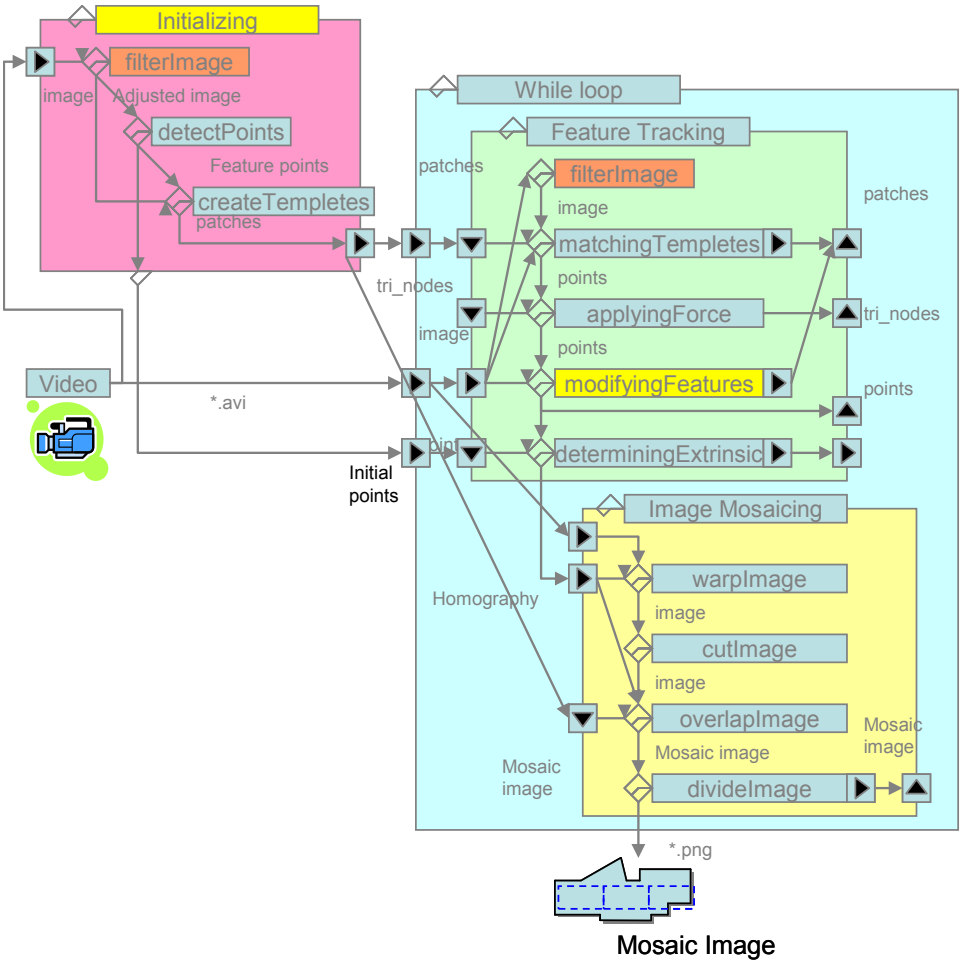


Figure 7-3. Diagram of algorithm structure.

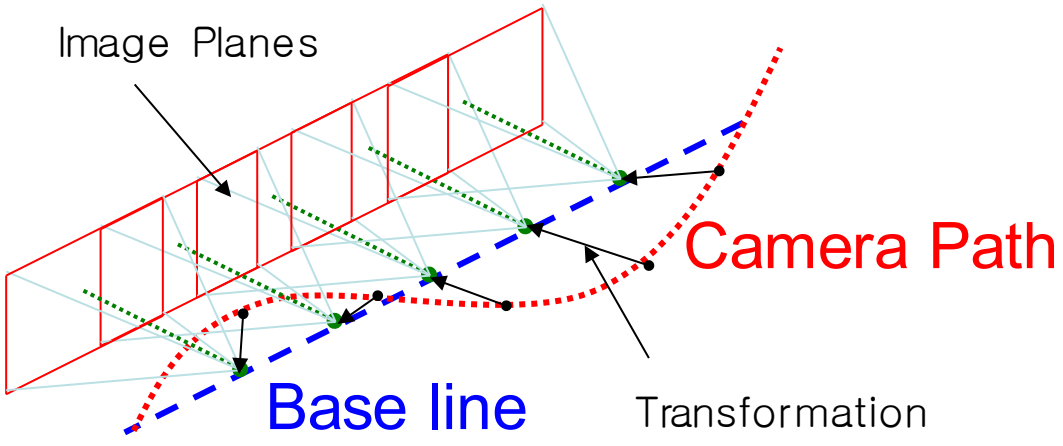


Figure 7-4. Translation projection model.

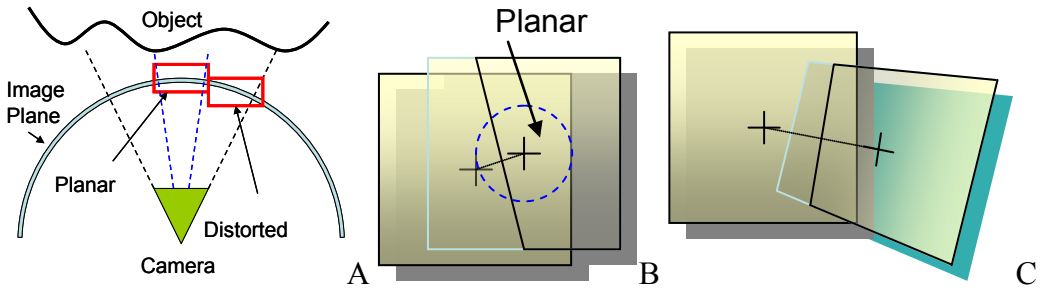


Figure 7-5. Alignment. A) Planar assumption. Registration with B) plane image and C) projective warped image.

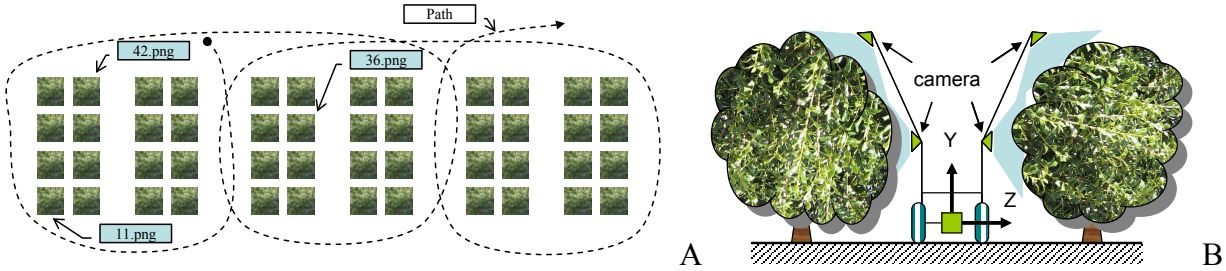


Figure 7-6. File numbering. A) Numbering of divided mosaic image. B) Ground-based scouting system.

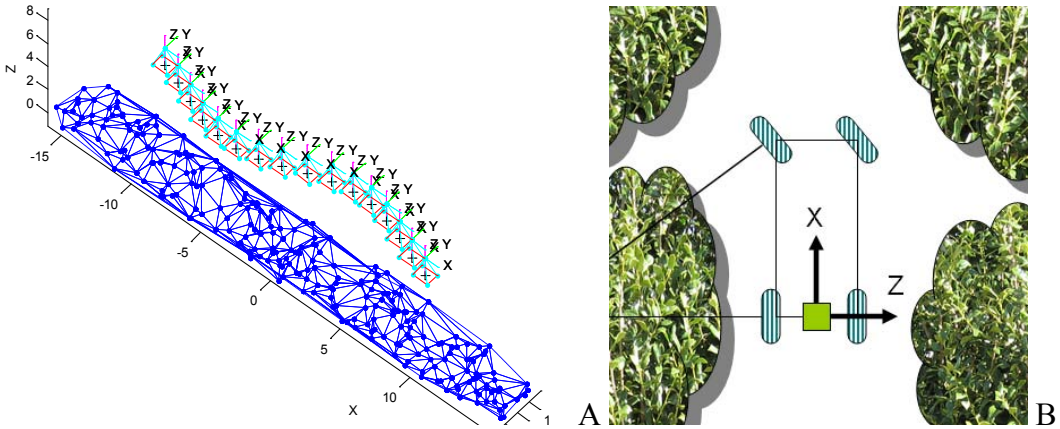


Figure 7-7. Configuation of image mosaicing simulation. A) Virtual non-planar model. B) Coordinates of camera.

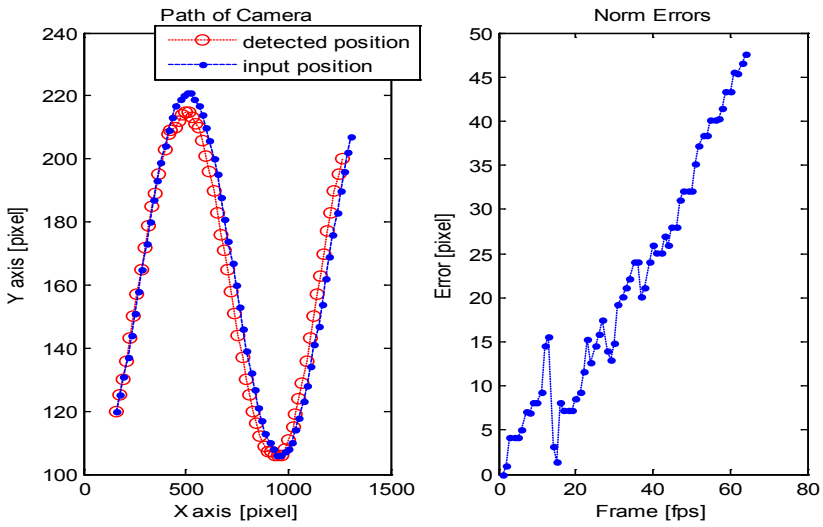


Figure 7-8. Evaluation of positioning error or camera.

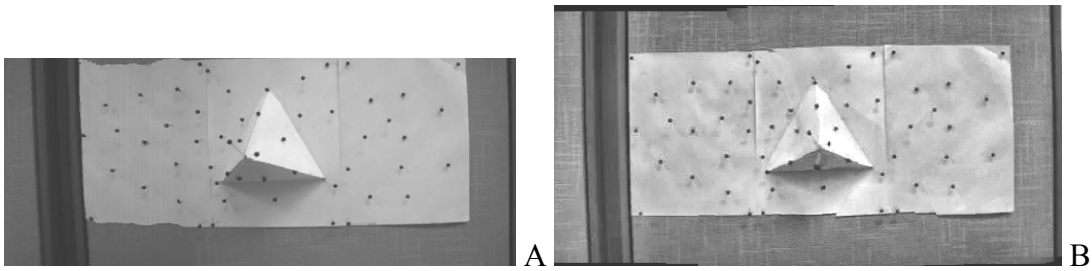


Figure 7-9. Cropping edge mosaicing effect. Comparison between mosaicing results from A) MATLAB and B) developed algorithm.

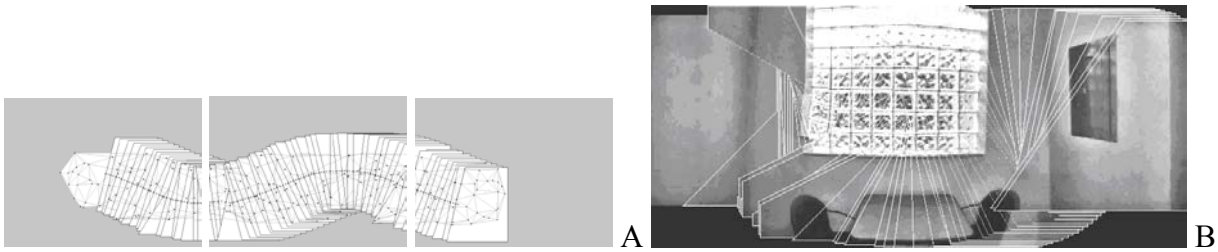


Figure 7-10. Mosaicing with sinusoid motion. A) Translational sinusoid motion and B) Rotational sinusoid motion.

Figure 7-11. Mosaicing result from video clip recorded far enough away from citrus grove.

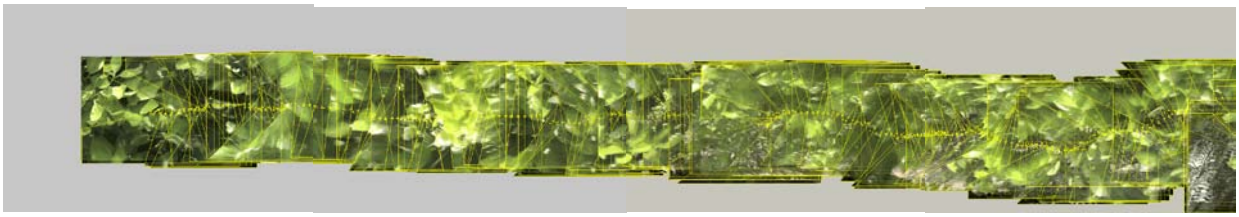


Figure 7-12. Mosaicing result from 5 times zoomed in video clip.

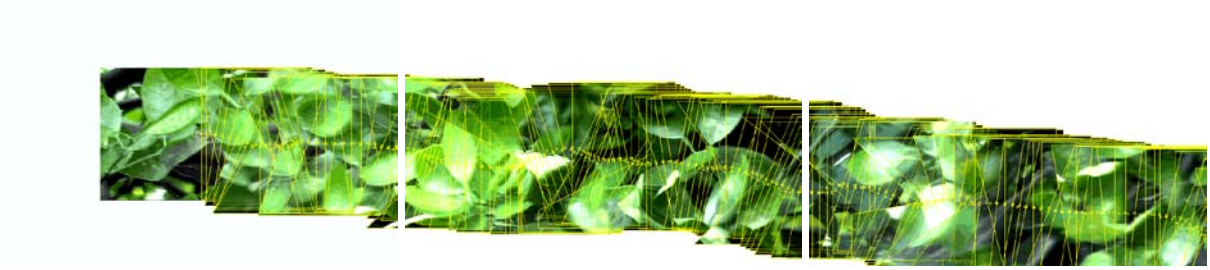


Figure 7-13. Mosaicing result from 10 times zoomed in video clip.

CHAPTER 8
VISION BASED 3D RECONSTRUCTION

The objective of this chapter is to reconstruct the surface of a citrus canopy so that the robot harvesting system can map and navigate fruit positions later.

Introduction

A 3D reconstruction of the canopy can be not only used to measure the volume of the canopy, but also to visualize the canopy in virtual space. To reconstruct the surface of the canopy, various sensors could be used. Vision systems are representative devices used for detecting the features of an object. Features extracted from the canopy become vertices of the surface of the canopy. The depth of a vertex is determined based on two-view geometry.

To reconstruct a structure, two sets of matching points and camera motion are needed, where matching points are projections of the actual 3D surface. When both the 3D position and 2D projection are given, camera motion can be estimated. The inputs of all steps require the others' outputs. Therefore, it forms a circular loop. Figure 8-1 shows this relationship and parameters among projection, motion estimation, and reconstruction. This loop is prone to diverge due to the accumulation of errors unless ground-truth is given.

An 8 point algorithm based on two-view geometry is widely used in reconstruction. The solution of two-view geometry is determined by Singular Value Decomposition (SVD) or inverse matrix operation, as introduced in Chapter 2. Since correspondences are coupled in the 8 point algorithm, errors that occur at some of the correspondences influence the existing reconstruction. Additionally, re-projection errors continue to increase.

Since the robot manipulator used in my study provides reliable motion of the end-effector, a Plücker coordinates system was applied to the 3D reconstruction. This approach calculates correspondences individually with a given camera motion. A pair of over 8 points is not required,

and local errors do not affect the existing reconstruction anymore. Because the Plücker coordinates system mainly uses the dot product and the cross product, its expressions about 3D computation is considerably simpler than ordinary triangular equations. It does not require any complicated matrix operations including SVD or inverse matrix. In addition, computation costs can be less expensive in certain computing systems which are specified for the matrix operations.

Methods and Procedures

This section shows how to manage the quality of interest points and discusses known problems with an 8 point algorithm. The alternative approach using a Plücker coordinates system is proposed as a counterproposal to the reconstruction. Simulation and indoor experiment are carried out to confirm results through re-projection error.

3D Reconstruction using an 8 Point Algorithm

The basic procedure of an 8 point algorithm is to estimate a relative camera motion, and then reconstruct the vertices based on the estimated camera motion. Since the estimated camera motion is relative, the origin coordinates must always be given. Therefore, base vertices are considered for the initial origin coordinates.

Markerless approach

The base vertex is a known basis, called ‘marker’, to estimate the origin coordinates. ‘Markerless’ means that 3D reconstruction is carried out without known markers. Since the harvesting robot motion model assumes that the frames get out of the initial frame, it loses the origin coordinates eventually. One possible way to achieve markerless, reconstruction is to continue adapting stable vertices as base vertices from reconstructed vertices.

Initial base vertex model

Since an icosahedron has many vertices which are not placed on a plane, and it is easy to make precisely, icosahedron base vertex models like Figure 8-2 were made to provide the initial known base vertex and compare the accuracy of reconstruction.

Interest point manager

The aim of interest point manager is to judge and manage the quality of Interest points. Two-view geometry assumes that given sets of correspondences are exact projections of identical vertices. In practice, input sets of correspondences are contaminated by a variety of noises or outliers. During the processes, even one outlier can cause the failure of the whole reconstruction eventually in an 8 point algorithm. Therefore, it is required that the interest point manager is capable of measuring and managing the quality of interest point quite strictly. Figure 8-3 shows a flow chart of the interest point manager. Since a list of interest points on each frame does not have an index, an additional index list is assigned where it adds and deletes elements of the list when events happen. There are three sets of lists for interest points.

- Interest point set (CF*) is used to track and reconstruct correspondences.
- Base point set (BM*) is used to estimate the camera motion.
- Candidate set (BE*) is used to judge which interest point is good or bad to be a base vertex.

When a camera moves, four events occur– interest point selection, interest point removal, base vertex selection and base vertex removal.

Interest point selection: The rule of adding a new interest point is to choose new interest points in empty area farthest from existing Interest points as show in Figure 8-4. This rule allows interest points to survive longer while the scene moves, as well as to avoid concentrating on one spot. The steps are as follows.

1. Extract an adequate number of candidate interest points.
2. Compute norm of difference between each new point and existing points.
3. Sort the list of norm.

4. Take as many new points as needed, which have the largest norm in the rest of the list.

Figure 8-4 A) illustrates a case that a camera moves leftward. The leftmost interest points will be picked up first, then the next middle points will be repeatedly picked up.

Interest point removal: As the camera moves, interest points are also supposed to change. When some of interest points are overlapped or go outside of frame edges, they must be removed in the index list. Since interest points not picked up as a base point have no information to track anymore, they should be discarded as soon as they disappear.

Once the Lucas-Kanade tracking function in OpenCV library, for example loses some of interest points, it places them into the nearest corners rather than discards them. This behavior is quite unnecessary for 3D reconstruction. This phenomenon always occurs around the frame edges. One solution is to force to discard them before they reach the frame edges. The optical flow's direction determines which edge is chosen, and its maximum speed establishes how thick the area will be. Eight pixels were selected for the edge thickness in this experiment.

Occasionally, some of the interest points are not placed on corners. In this case, they are prone to fluctuate around their initial position. Those altering movements lead to the wrong reconstruction. One of the solutions to these unstable interest points is to measure the difference between the input position and the re-projected position with respect to reconstructed vertices, which is called re-projection error. Figure 8-6 shows that some difference between input position (▫) and re-projected position (x) occurs when an interest point is placed on a non-corner. When a re-projection error of an interest point exceeds a threshold, it is discarded. Through repeated experiments, the threshold was set at 0.015 [in].

Base vertex selection and removal: Reconstructed vertices which have small re-projection errors or converge stably are picked up as new base vertices. When stable interest

points between frames converge into constant positions, the reconstructed vertices corresponding to them become candidates for the base vertices. A base frame provides the second view to compute reconstruction. When base vertices are chosen, the base frame is also captured and is held until the next selection event.

To estimate convergence, differentiation and standard deviation of re-projection error is observed between frames. If the reconstruction is adequate, differentiation and standard deviation of re-projection error will be almost zero. Figure 8-7 shows examples of a stable vertex and an unstable vertex. Unstable Interest points are prone to continue fluctuating as shown Figure 8-7 B). In a way, it is necessary to limit the number of samples to compute statistic quantity, thus minimizing the effect by momentarily wrong measurement. Figure 8-8 shows the statistical quantity of each vertex by using different sizes of circles. Large statistical quantity means instability, as a perfect measurement leads it to zero. The minimum error in this experiment was 0.01 [in]. The threshold for base vertices was set 0.02 [in] as shown in Figure 8-8 B).

Implementation

A real-time 8 point algorithm was implemented with OpenCV and C++Builder. C++Builder takes charge of the display and user interface, and OpenCV carries out image processing and connectivity to a camera. The implemented software was also designed to process video clips as well as real-time camera input. The farther the camera motion from the base position, the more accurately an estimation is calculated. Figure 8-9 shows successful reconstruction while base points show up on both the left base frame and the right current frame.

However, the base points used for motion estimation must continually be provided so that the camera keeps moving. Even though the re-projection error is small, accumulated errors eventually contribute to the failure of reconstruction. Figure 8-10 shows that the re-projection

error continues to increase, as the number of base vertices increases. The X-axis and Y-axis denote the number of frames and pixels, respectively. This effect is because it loses the global ground truth to maintain the origin coordinates as the scene moves away from the first initial view.

Sensitivity to round-off error

3D reconstruction through an 8 point algorithm is considerably more accurate with floating type interest points, and re-projection errors are virtually zero. In terms of pixel units of a real CCD image sensor, round-off error could occur because of integer type interest points. To examine the sensitivity of an 8 point algorithm, noises within 4 pixels, 0.0124 [in] (0.315 mm) on CCD were added purposely as shown in Figure 8-11 A). 4 pixels is not a big error for fluctuated interest points in the image, and is not noticeable as shown in Figure 8-11 A). Nevertheless, Figure 8-11 B) shows that the reconstruction deviates with big differences between the original and reconstructed vertices. The norm error of reconstruction was 6.3264 [in] when cameras were placed at 15 [in] away from the object in this test.

The effect of round-off error in an 8 point algorithm was simulated for two different types of vision systems as shown in Figure 8-13. The dashed red lines are reconstructed vertices. Figure 8-12 A) and C) show that the reconstruction was perfectly achieved in both systems when floating type interest points are used. However, in the case of integer type interest points, the reconstruction was unacceptable in both systems. Since stereo vision could secure many more correspondences, the reconstruction errors were relatively smaller as shown in Figure 8-12 D).

Discord index

Since interest points are just a collection gathered, indices are assigned to manage (add / delete) interest points, as well as to provide correspondent information. Even though there is nothing wrong with the index manager, mismatching error happens on occasion while running

the software. Figure 8-13 shows a mismatching case of interest points between frames. The black solid lines represent appropriate matching, and the green dotted lines represent mismatching. Since this reconstruction software assumes that correspondences are always correct, and the program is not designed to examine the adequacy of matching again, the whole reconstruction is prone to collapse, once mismatching happens.

3D Reconstruction using a Plücker Coordinates System

An 8 point algorithm is widely referred to as a 3D reconstruction. Because correspondences are coupled in an 8 point algorithm, even a few outliers can influence the whole reconstruction. In addition, accumulated errors keep increasing. Therefore, many methods such as RANSAC, bundle adjustment and Marque-Levenberg are employed to discard outliers. However, according to the circular relationship shown in Figure 8-1, no method without ground truth is likely to resolve the accumulation of error.

The robot manipulator used in my study provides measurable and repeatable motion of the end-effector. If the motion information is given as a ground-truth, a Plücker coordinates systems can be simply applied to a 3D reconstruction instead of an 8 point algorithm. This approach is intuitive and independently computes correspondences with given camera motions. Computation is conducted about a single correspondence; therefore, it is not necessary to take a set of over 8 points. Furthermore, since any individual error does not influence the existing reconstruction any longer, it is not necessary to check outliers during reconstruction. This approach starts with defining a line in a space (source: Crane III, 2006, screw theory lecture note).

A line in Plücker coordinates is defined as follows.

$$\mathbf{L} = \{\mathbf{S}; \mathbf{S}_{0L}\}, \tag{8-1}$$

where $\mathbf{S}$ is a normalized vector between two points, and $\mathbf{S}_{0L}$ is a moment with respect to an origin as shown in Figure 8-14 A). $\mathbf{S}$ and $\mathbf{S}_{0L}$ are determined by given points $\mathbf{P}$ and $\mathbf{T}$.

$$\mathbf{S} = \frac{\mathbf{P} - \mathbf{T}}{|\mathbf{P} - \mathbf{T}|}, \mathbf{S}_{0L} = \mathbf{P} \times \mathbf{S}, \tag{8-2}$$

where $\mathbf{T} = [x_c \quad y_c \quad z_c]^T$ is the center of the camera, and $\mathbf{P} = [x_w \quad y_w \quad z_w]^T$ is the correspondent point on the image plane with respect to world coordinates. The $\mathbf{P}$ is determined by the extrinsic camera parameters. Figure 8-14 B) represents

$$\mathbf{P} = \mathbf{R}\mathbf{p} + \mathbf{T}, \tag{8-3}$$

where $\mathbf{R}$ is the orientation of the camera which has a 3x3 rotation matrix, $\mathbf{p} = [x_{im} \quad y_{im}]^T$ is a correspondent point with respect to image coordinates. Assuming the correspondent points $\mathbf{P}_1$ and $\mathbf{P}_2$ are projections of a vertex $\mathbf{Q}$ of an object, the lines $\mathbf{L}_1$ and $\mathbf{L}_2$ are on the same plane as shown in Figure 8-15 A). In other words, two non-parallel lines on the plane must intersect at $\mathbf{Q}$.

Given the two lines $\mathbf{L}_1$ and $\mathbf{L}_2$, the intersection point $\mathbf{Q}$ can be determined by using the following equation.

$$\mathbf{Q} = \frac{\mathbf{S}_{12} \cdot \mathbf{S}_{0L2}}{\sin\theta}\mathbf{S}_1 + \frac{\mathbf{S}_{12} \cdot \mathbf{S}_{0L1}}{\sin\theta}\mathbf{S}_2, \tag{8-4}$$

where θ is an angle between lines, $\mathbf{S}_{12}$ denotes normalized cross product between $\mathbf{S}_1$ and $\mathbf{S}_2$.

$$\theta = \cos^{-1}(\mathbf{S}_1 \cdot \mathbf{S}_2), \ \mathbf{S}_{12} = \frac{\mathbf{S}_1 \times \mathbf{S}_2}{|\mathbf{S}_1 \times \mathbf{S}_2|}. \tag{8-5}$$

If lines $\mathbf{L}_1$ and $\mathbf{L}_2$ are not on the same plane, there can still exist another line $\mathbf{L}_{12}$ orthogonal to both $\mathbf{L}_1$ and $\mathbf{L}_2$ as shown in Figure 8-15 B).

Given the non-coplanar lines $\mathbf{L}_1$ and $\mathbf{L}_2$, the intersection point $\mathbf{Q}_1$ and $\mathbf{Q}_2$ can be determined by using the following equation.

$$\mathbf{Q}_1 = \frac{\mathbf{S}_{12} \times (\mathbf{S}_{0L2} - d(\mathbf{S}_{12} \times \mathbf{S}_2))}{\mathbf{S}_{12} \cdot \mathbf{S}_{0L1}},\ \mathbf{Q}_2 = \mathbf{Q}_1 + d\mathbf{S}_{12}, \tag{8-6}$$

where d is the distance between $\mathbf{Q}_1$ and $\mathbf{Q}_2$.

$$d = -\frac{\mathbf{S}_2 \cdot \mathbf{S}_{0L1} + \mathbf{S}_1 \cdot \mathbf{S}_{0L2}}{\sin\theta}. \tag{8-7}$$

Due to interference, interest points on the image may not be an accurate enough projection. Therefore, the two lines are unlikely to be on the same plane in reality. In this case, $\mathbf{Q}$ can be approximated by taking the average between $\mathbf{Q}_1$ and $\mathbf{Q}_2$. (Refer to Appendix A to see more inducing procedures.)

$$\mathbf{Q} = \frac{\mathbf{Q}_1 + \mathbf{Q}_2}{2}. \tag{8-8}$$

To validate the equations, a reconstruction simulation using a Plücker coordinates system was conducted with a virtual tetrahedron model as shown Figure 8-16. The red crosses are input correspondences and the cyan dots are re-projections. The re-projection errors were almost zero in this simulation. Therefore, the idea that the equations work precisely was confirmed.

Experiment

Configuration

Figure 8-17 shows that a camera is attached on the end-effector of a robot manipulator. A camera does not need to be in the center of the end-effector. The robot manipulator used is Robotics Research 1207 which has 7 DOF, and provides the position and orientation of the end-effector.

Robot motion generation

Since the inverse kinematics formula of the robot manipulator used in this experiment was unknown, the robot was moved manually to create particular motions. To save resources, the motion simulation software was implemented as shown in Figure 8-18 A). This software allows humans to jog the virtual robot manipulator in virtual space with respect to global coordinates. Once base motions are determined, intermediate motions are generated by fitting methods such as Spline. Figure 8-18 B) shows each joint angle was generated automatically.

Robot motion control and capture image

To make it easy to capture images, fully automated and integrated robot control software was implemented as shown in Figure 8-19 A). This software was implemented to load and display motion data, send commands to the robot manipulator and capture and save images from a camera. Since it is unlikely to take clear shots due to the vibrations caused by inertia, it was designed to stop at each position for 3 seconds. Distance between motions was set within 10 [in] to secure a sufficiently overlapped area between sequential images. Figure 8-19 B) shows the captured images along the path of motion.

Calibrations for Camera Parameters

Every real camera has its own intrinsic parameters such as lens distortion, focal length, principal point, scaling factors and skew angels. One of cameras used in my study has a very small lens. The smaller focal length a camera has, the more lens distortion occurs. Since two-view geometry assumes an ideal flat image plane, it is necessary to compensate for lens distortion effect.

Rectification for lens distortion effect

Two types of lens distortion are typically considered. A tangential distortion represents the degree of skewing between axes. A radial distortion illustrates either barrel or pincushion effect.

For objects close to a camera, a barrel distortion occurs. These distortions can be compensated for with the following equations, which are introduced by Camera Calibration Toolbox for MATLAB of Caltech Vision lab (2008).

Tangential distortion (skew effect):

$$\mathbf{d}_{\text{tangential}} = \begin{bmatrix} 2xy & \left(r^2 + 2x^2\right) \\ \left(r^2 + 2y^2\right) & 2xy \end{bmatrix} \begin{pmatrix} k_3 \\ k_4 \end{pmatrix} = \begin{bmatrix} 2k_3xy + k_4\left(r^2 + 2x^2\right) \\ k_3\left(r^2 + 2y^2\right) + k_4 2xy \end{bmatrix}. \tag{8-9}$$

Radial distortion (barrel or pincushion effect):

$$\begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = KK \begin{pmatrix} x_d \\ x_d \\ 1 \end{pmatrix}, \text{ where } \begin{pmatrix} x_d \\ y_d \end{pmatrix} = \left(1 + k_1r^2 + k_2r^4 + k_5r^6\right) \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{d}_{\text{tangential}}, \tag{8-10}$$

where $r^2 = x^2 + y^2$. Coefficient k's are determined through the camera calibration. Since these equations are reversible, it is possible to generate virtual correspondences for simulation. Figure 8-20 B) shows the undistorted image through camera calibration. The intrinsic parameters are shown in Table 8-1 and the coefficient k determined through the calibration was $k = \begin{bmatrix} -0.6717 & 0.8389 & -0.0047 & 0.0148 & 0 \end{bmatrix}$.

Accuracy of camera motion estimation

Figure 8-21 A) shows the extrinsic camera parameters estimated by camera calibration. The re-projection errors were within 2 pixels. That means the estimated camera motions were acceptably accurate. When these camera motions were transformed into the motion information from the robot manipulator, the target checkerboard is supposed to coincide if the camera motions are accurate. However, the target checkerboard did not coincide as shown in Figure 8-21 B). Therefore, it is regarded that the motion information from the robot manipulator has some errors. The maximum difference between samples was about 3 [in].

Feature Matching

For matching correspondences, the performance of SIFT and SURF feature matching methods were examined with several pairs of indoor scenes. Figure 8-22 shows matching results from several pairs of images which are taken with about 2 [in] distance. The dashed lines indicate optical flow vector. The similar optical flow vectors are prone to denote similar depth of vertices. Figure 8-22 A) shows several mismatched correspondences whose vectors are twisted. Optical flow vectors that differ greatly from the dominant optical flow vector are excluded. This rule was applied to discard mismatched correspondences. Figure 8-22 B) shows most of the outliers were discarded. Through the test, it is regarded that SURF shows adequate matching performance and high speed relatively more than SIFT. Average computation time for a pair was 3 times faster than SIFT.

Indoor Experiments

The 3D reconstruction algorithm was examined on an indoor scene before applying it to an actual canopy. Since it is difficult and inaccurate to measure the size of an actual canopy, it is necessary to confirm the algorithm with known sizes of objects. To confirm the performance of the algorithm, the reconstruction of two icosahedrons was conducted at the two pairs of correspondences as shown in Figure 8-23 A). Two sets of interest points were selected manually. The difference between input correspondences and re-projected points are adequately the same as shown in Figure 8-23 B), which means reconstruction is locally correct. However, since the motion information from the robot manipulator is not quite accurate, reconstructed icosahedrons at each motion did not coincide globally as shown in Figure 8-23 C). This discordant result represents errors between two different matching sets. This offset can be compensated by calibrations between motion information and physical position.

Figure 8-24 shows a result in reconstruction of another object, using the same procedure. Interest points were selected manually (by hand) and automatically (by SURF), and both results were plotted in Figure 8-24 B). Even though there are inevitable errors during automatic selection, the entire reconstruction maintained a considerable deal of approximation.

To confirm the effect of input errors, the result of the Plücker coordinates system was compared with that of the 8 point algorithm. Figure 8-26 A) is the same plot with Figure 8-24 C). The reconstruction by an 8 point algorithm shown in Figure 8-25 B) looks impractical. Therefore, it is regarded that a Plücker coordinates system using a single point to compute, yields better performance than an 8 point algorithm using a set of coupled points for 3D reconstruction.

Outdoor Experiment

The algorithm was eventually examined with an actual canopy in Ocala, Florida. The experiment was carried out at Pine Acre grove in Florida. Figure 8-26 A) shows the operation on an actual canopy. Since canopies are much bigger than the workspace of the robot manipulator in general, the reconstructed surface of a canopy is just part of it. Figure 8-27 shows 75 sequence images sampled with a zigzag camera motion shown in Figure 8-26 B).

Results

Since the image quality was comparatively low and the overlapping area was not wide enough, many of the interest points were discarded unlike the indoor experiment. In addition, interest points were not distributed uniformly, and only a few correspondences were obtained at each pair of images. Figure 8-28 shows one of matching results. There were still about 10% of mismatching correspondences; however, those mismatching correspondences are unlikely to influence the final reconstruction. Feature tracking could contribute to holding an unbiased distribution of interest points, even though it is not in bad condition.

Figure 8-29 A) shows the final reconstruction of the canopy based on empirical measurement. This part of canopy surface shown in Figure 8-29 B) was approximated using 2D grid interpolation. The 2D grid interpolation can not only reduce the entire number of vertices, but also regulate biased vertices. In addition, .it can compensate wrong vertices too. If adequately many vertices were reconstructed, estimated vertices the surface could look more realistic. Figure 8-29 B) shows reconstructed vertices and camera motions. Since fruit detection was not applied to my study, some of the fruit positions were manually picked up, so that we can confirm the reconstruction was appropriately carried out. The circles in Figure 8-29 B) indicate fruit positions selected manually. The red dots in Figure 8-30 A) indicate re-projection from reconstruction. The re-projection errors between input fruit positions and re-projections were adequately small. Therefore, as we have seen in the indoor experiment, it is regarded that the surface reconstruction of a real canopy is conducted well. The reconstructed surface can be used for visualization and measurement of the volume.

Conclusion

In my study, 3D reconstruction of the citrus canopy was studied. An 8 point algorithm is an adequate approach for the reconstruction of objects which are entirely covered in the frame of view. However, it is hypersensitive to minor errors due to coupled input points. Furthermore, when objects are larger than the view frame like the canopy case, reconstruction errors are likely to accumulate. To match correspondences between sequential images, the SURF matching method was employed. A Plücker coordinates approach to reconstruct the 3D surface of canopy worked out adequately. The Plücker coordinates system's expression for reconstruction is intuitive and computed individually. Therefore, any individual error does not destroy the existing reconstruction. Since the camera motion obtained from the robot manipulator could be

inconsistent with the physical pose, it is necessary to take as many samples as possible. The successful result of reconstruction depends on how accurately feature points can be matched.

Future Work

Since the objective of my study was the reconstruction of the canopy, some of the automation processes were not implemented in my study. In future work, fruit detection will be included. Once fruit position was mapped on the surface of the canopy, the actual fruit position will be proved by positioning the end-effector of the robot manipulator to the mapped position from the reconstruction.

Table 8-1. Calibrated Intrinsic camera parameters.

Coordinates	Image Resolution	Forcal Length	Image Center	α
x-axis	640	747.01	276.89	0
y-axis	480	742.74	240.18	0

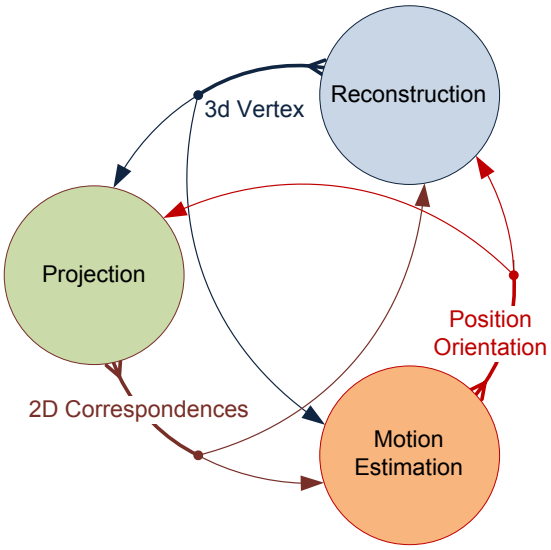


Figure 8-1. Loop among projection, motion estimation and reconstruction.

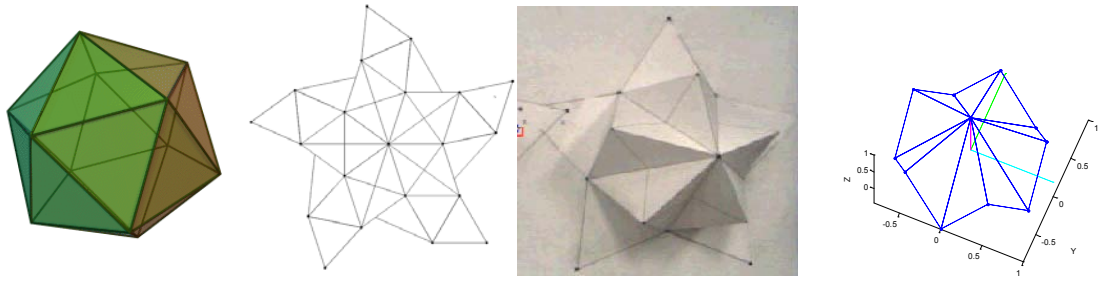


Figure 8-2. Icosahedron base vertex model.

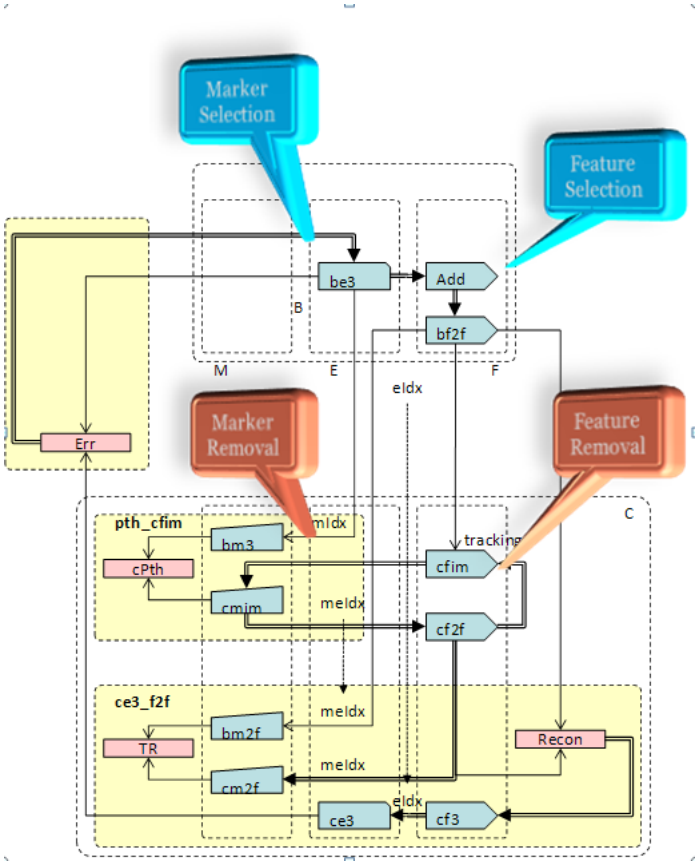


Figure 8-3. Flow chart of an 8 point algorithm.

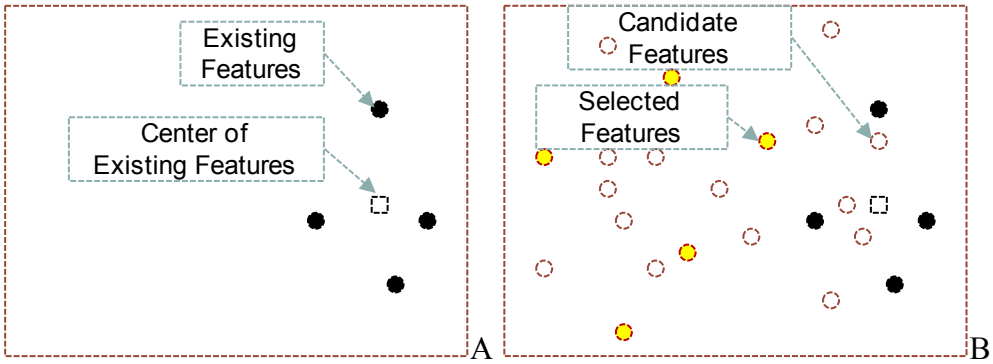


Figure 8-4. Interest point selection. Basis to pick select new Interest points at A) previous frame and B) next frame.

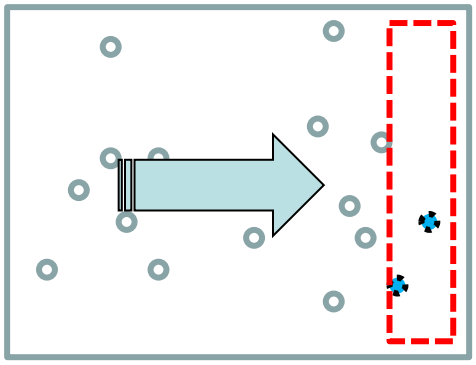


Figure 8-5. Interest points around edge are discarded.

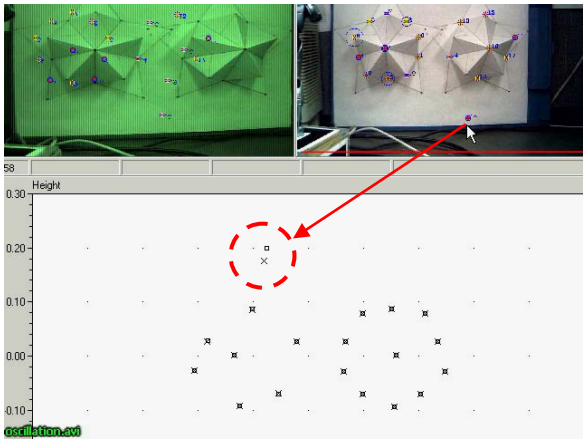


Figure 8-6. Re-projection error of unstable Interest points.

A B

Figure 8-7. Stability of vertex. A) Converging stable vertex. B) Fluctuating unstable vertex.

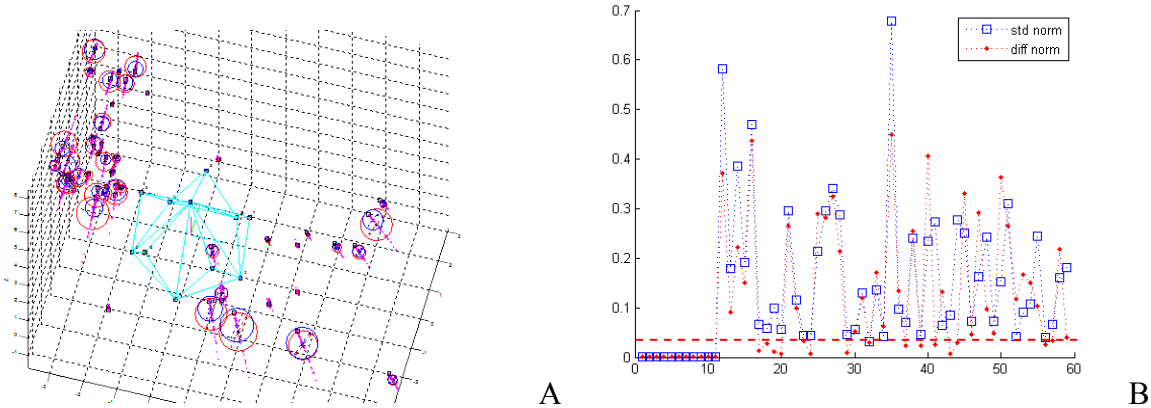


Figure 8-8. Average statistical quantity. A) Plot in 3D view. B) 2D plot with a threshold.

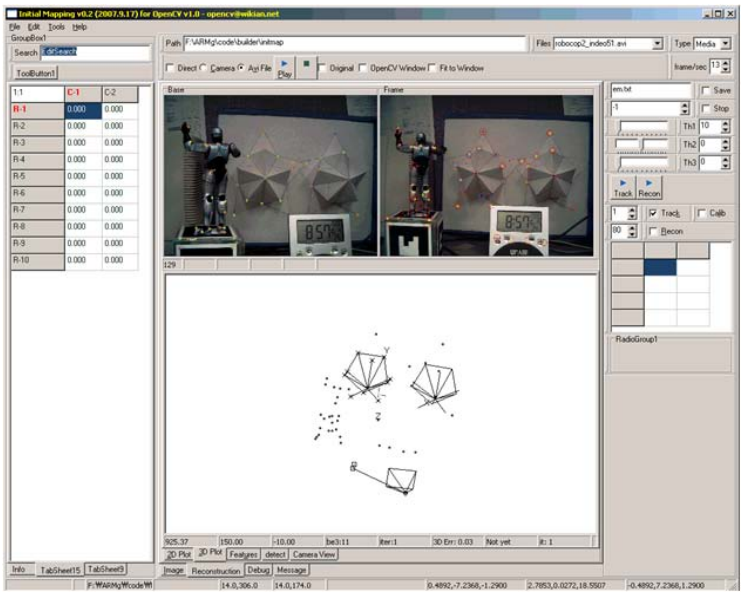


Figure 8-9. 3D reconstruction program using an 8 point algorithm in real-time.

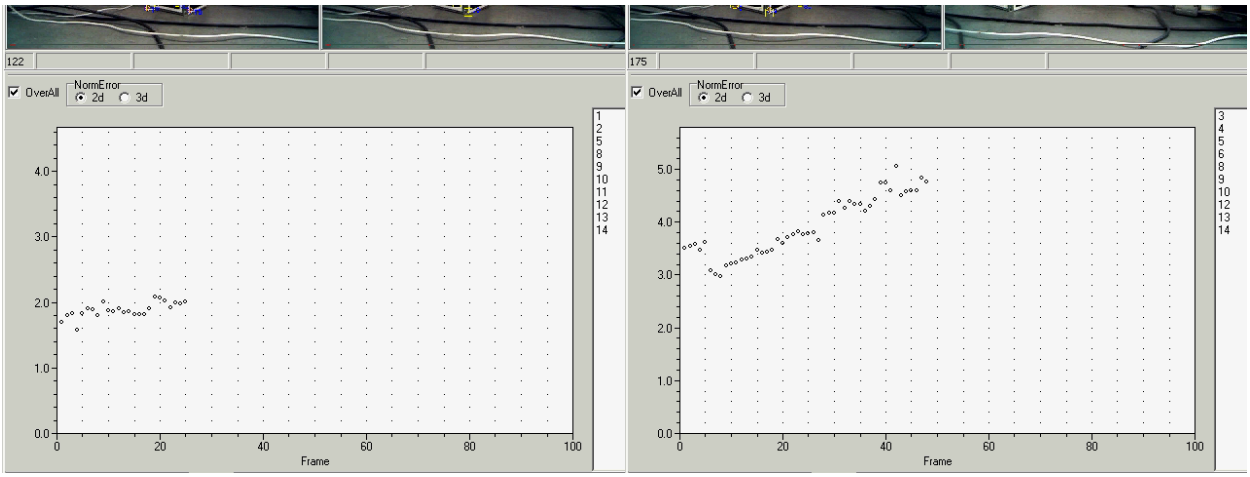


Figure 8-10. Increment of accumulated re-projection error.

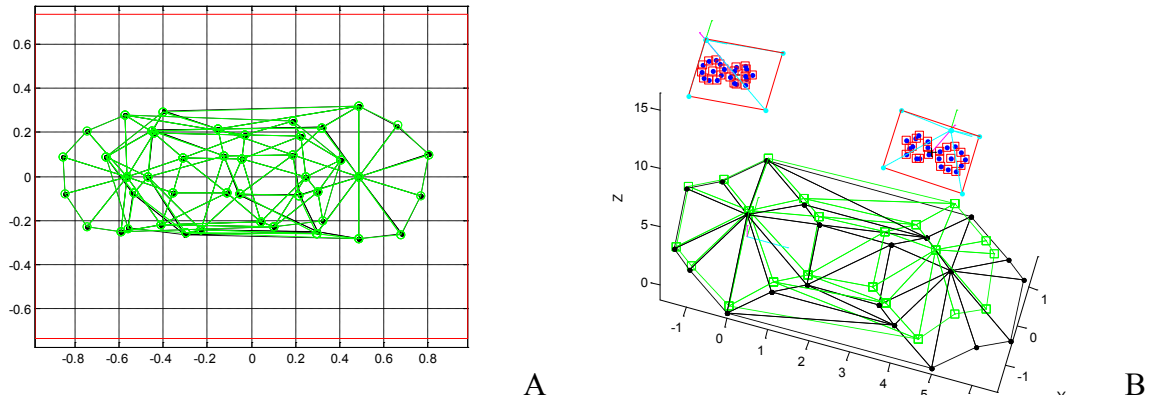


Figure 8-11. Sensitivity test of an 8 point algorithm with 4 pixel round-off error.

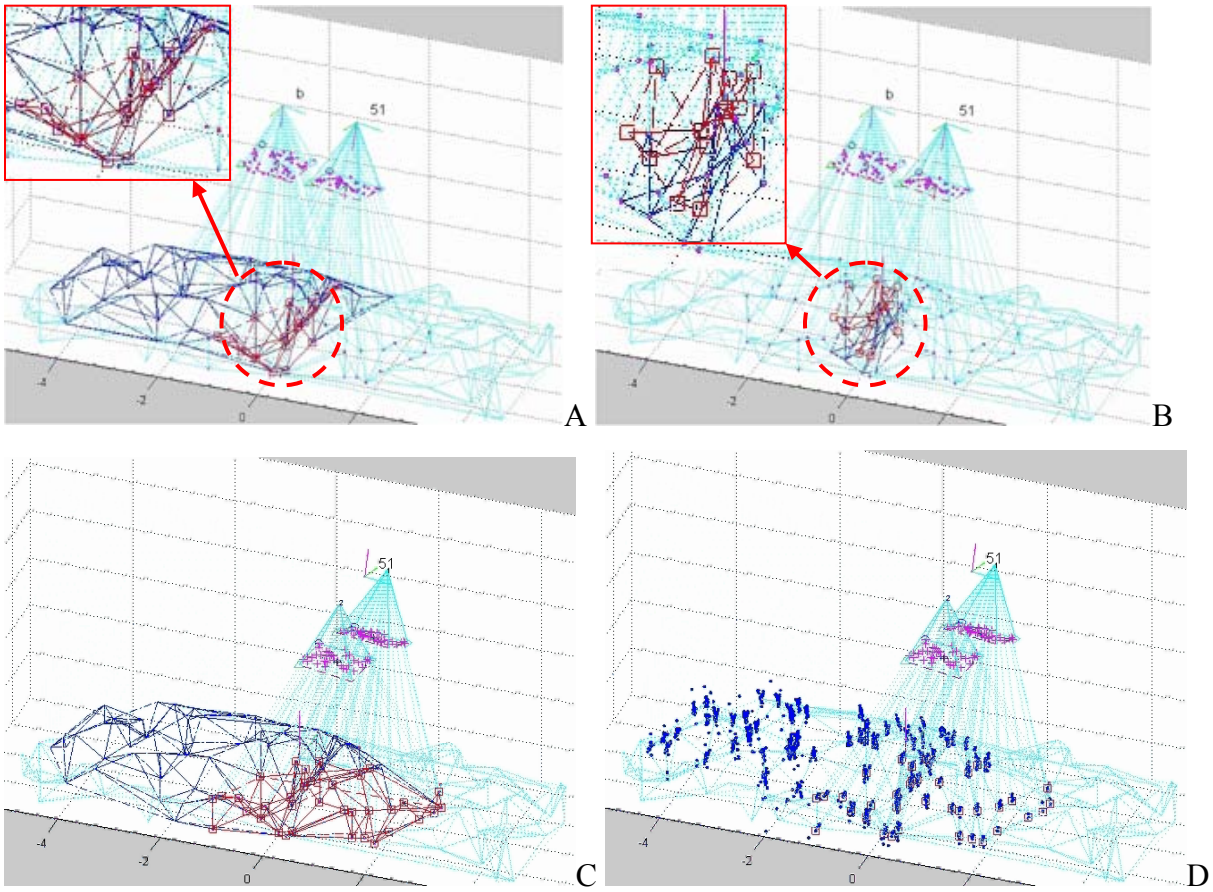


Figure 8-12. Reconstruction simulation. Simulation with A) floating type pixels in single vision, B) integer type pixels in single vision, C) floating type pixel in stereo vision and D) integer type pixels in stereo vision.

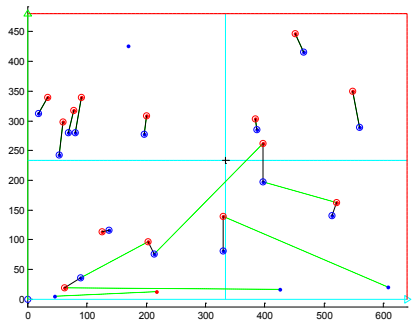


Figure 8-13. Mismatched index problem.

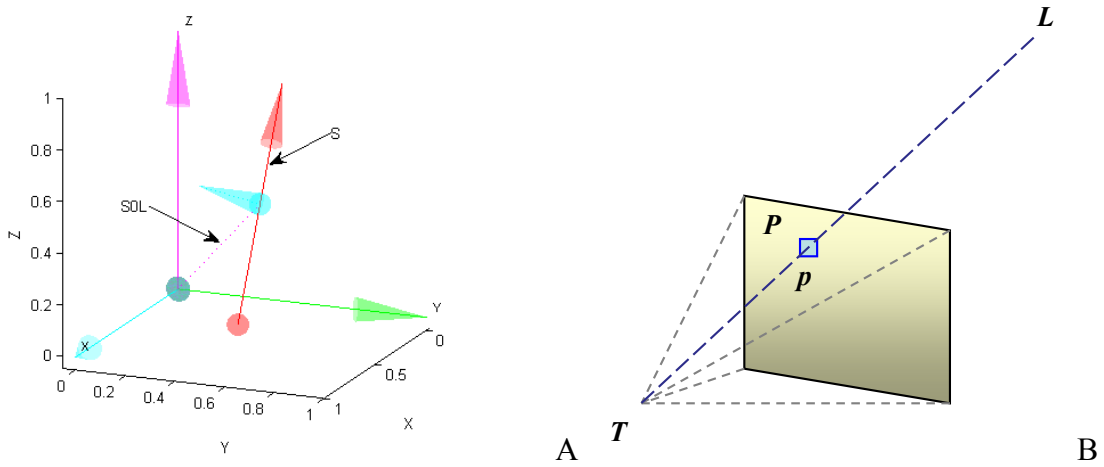


Figure 8-14. Plücker coordinates system. A) A line definition. B) Intersection between a line and a plan.

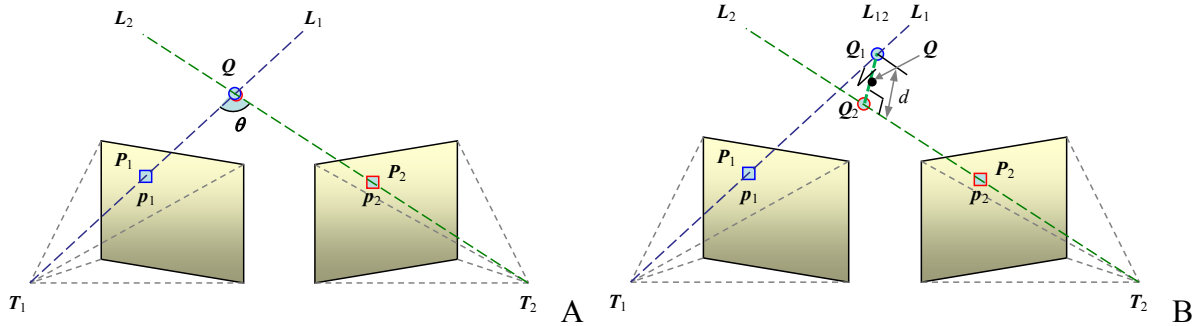


Figure 8-15. Determination of an object vertex. A) Two coplanar lines. B) Common line orthogonal with two lines.

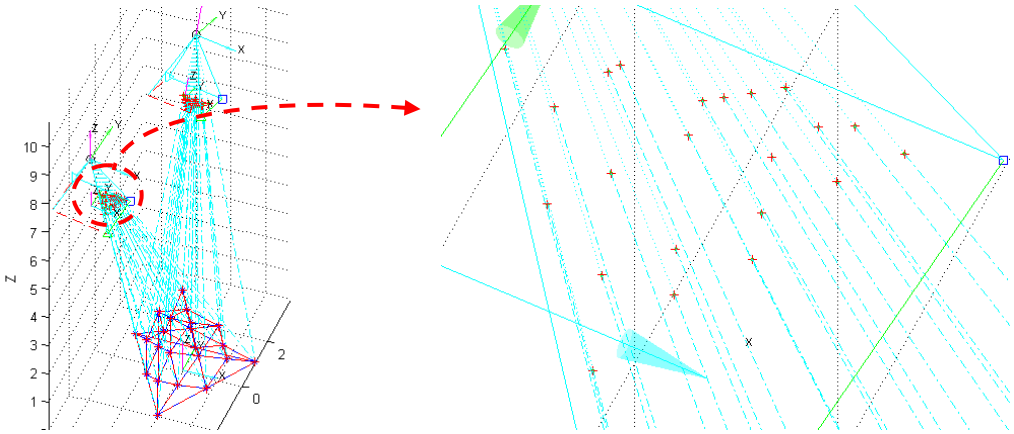


Figure 8-16. Simulation of reconstruction using a Plücker coordinate system.

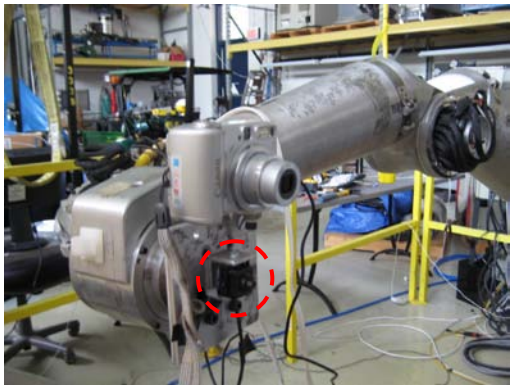
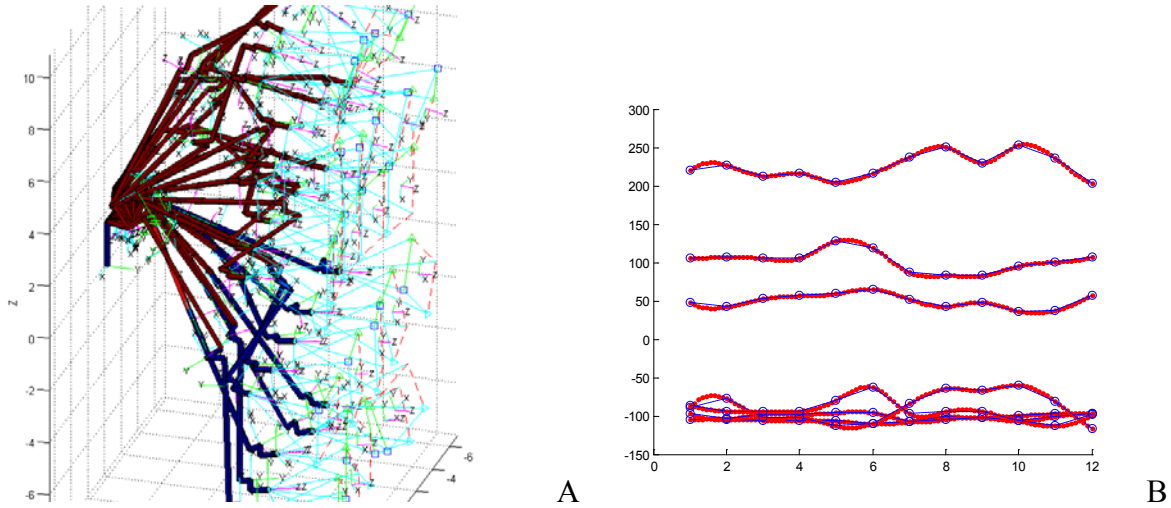


Figure 8-17. Configuration of camera on R1207 robot manipulator.



A B

Figure 8-18. Robot motion simulation. A) Base robot motion. B) Joint angle generation based on the base motion.

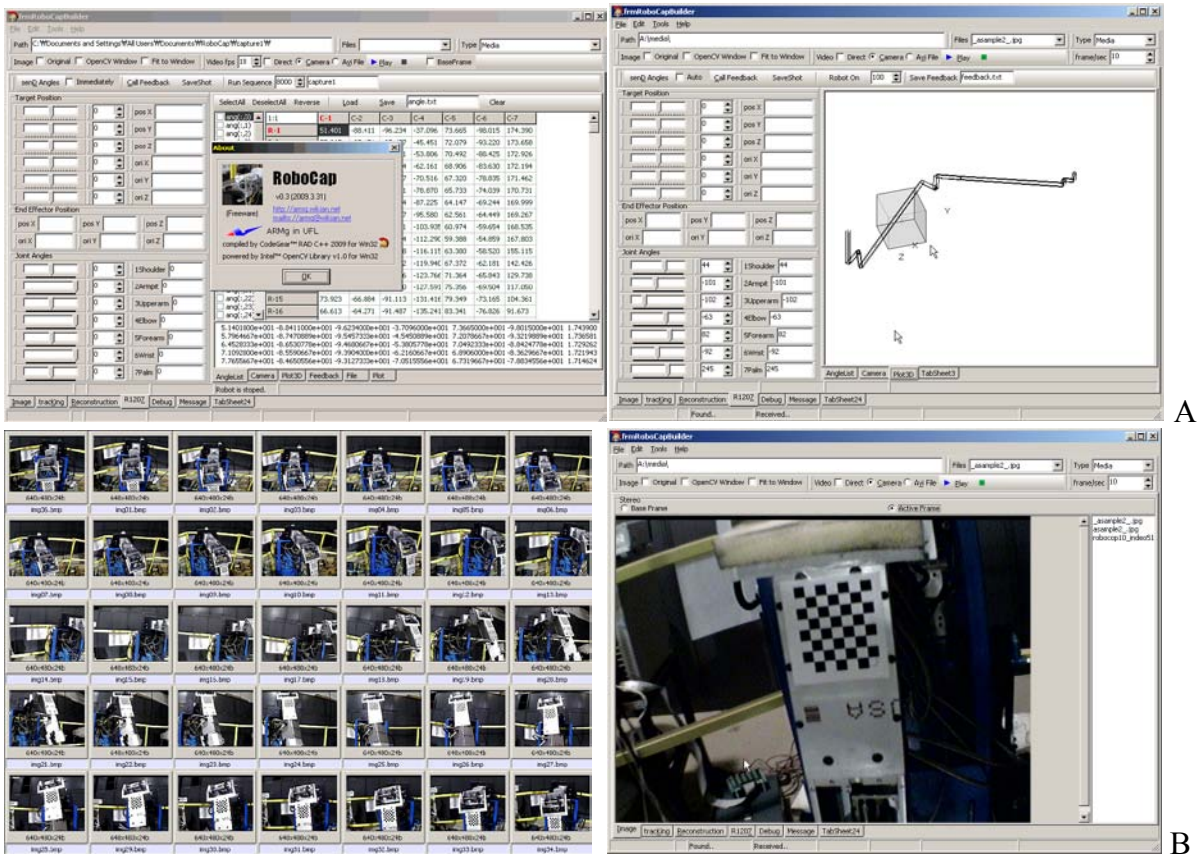


Figure 8-19. Robot control software. A) Robot motion simulation. B) Image acquisition.

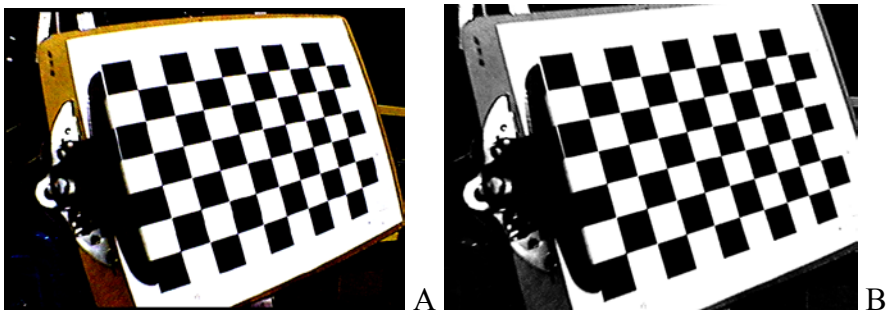


Figure 8-20. Camera calibration. A) Original image. B) Undistorted image.

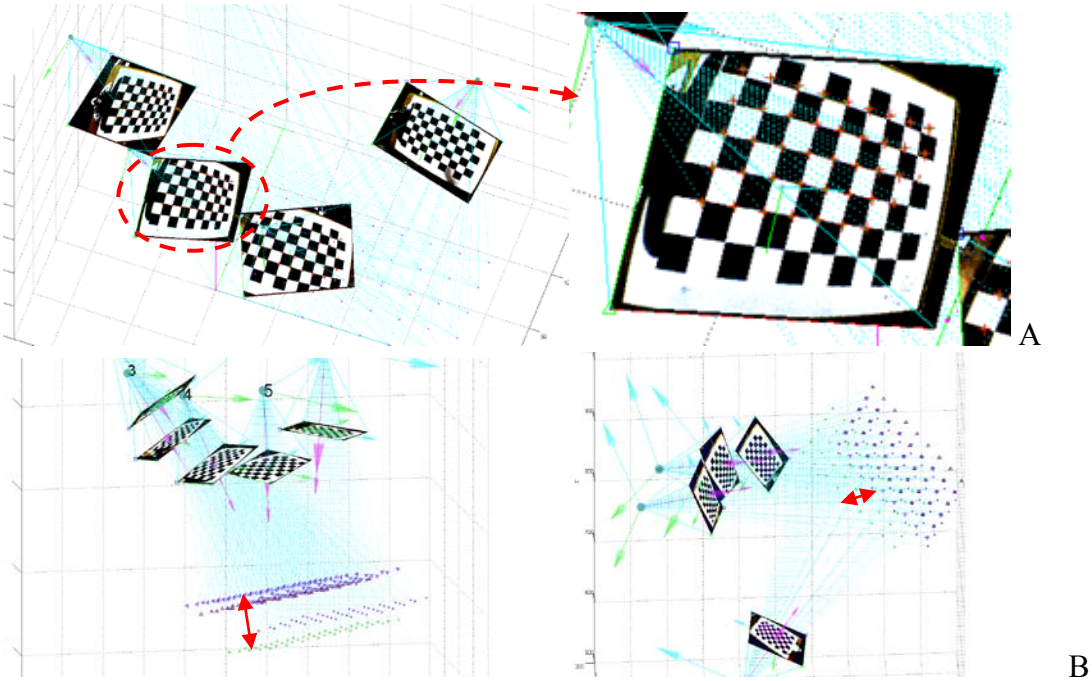


Figure 8-21. Detemination of camera motion. A) Camera motions estimated by camera calibration. B) Transformed camera motion based on motion information from robot manipulator.

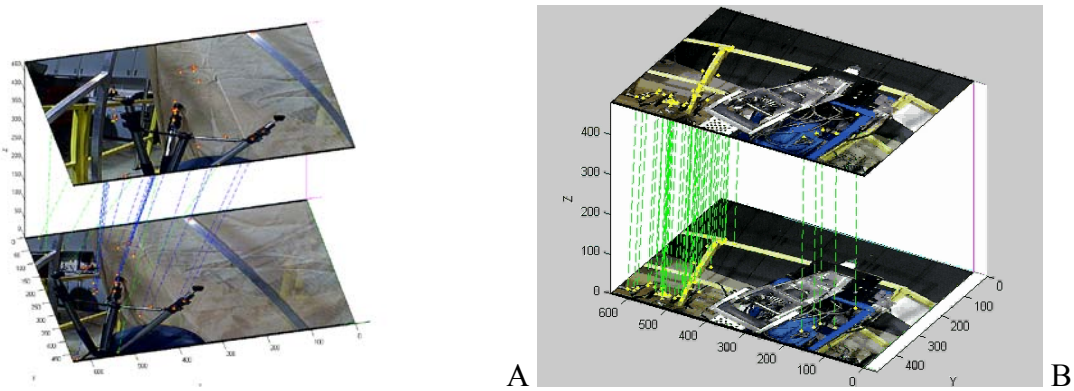


Figure 8-22. Feature matching. A) Matching by SIFT. B) Matching by SURF.

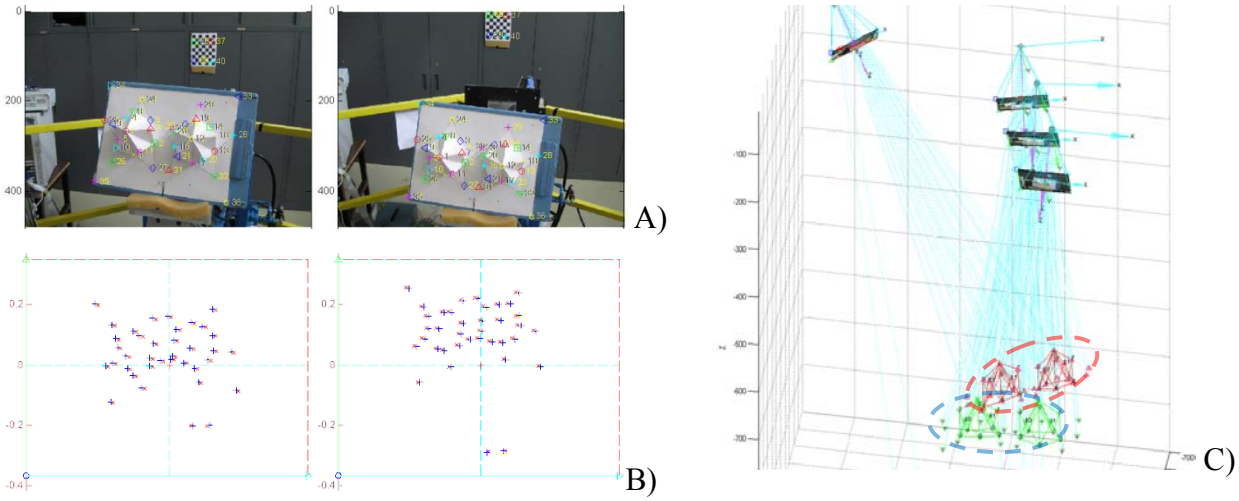


Figure 8-23. Reconstruction of icosahedrons. A) Icosahedron objects. B) Difference between input points and re-projected point. C) Reconstructed icosahedrons.

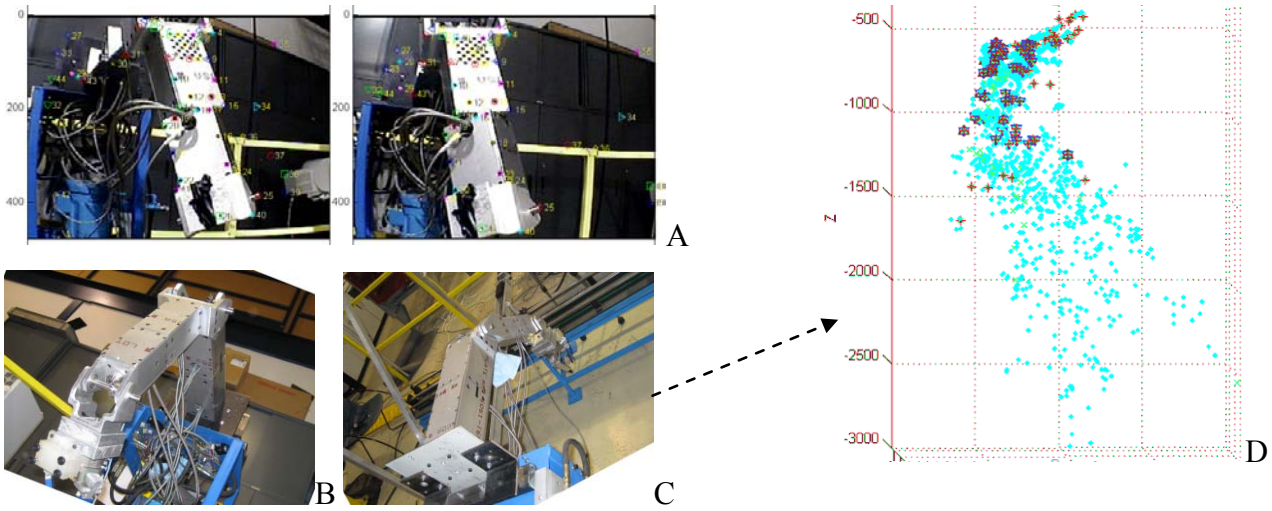


Figure 8-24. Reconstruction of a robot manipulator. A) Interest points picked up manually. B) Viewpoint of the object. C) Viewpoint of the object. D) Reconstructed result.

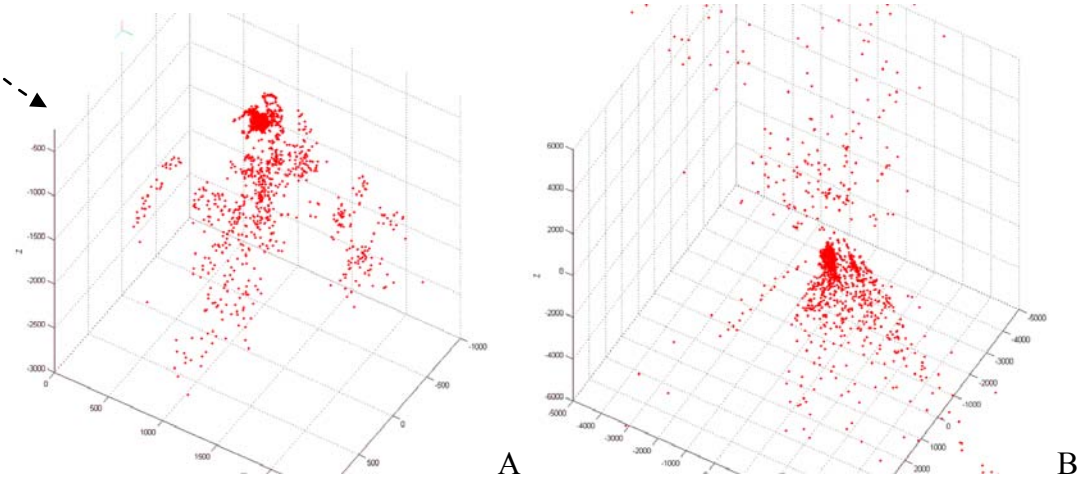


Figure 8-25. Reconstruction comparison. Reconstruction by A) a Plücker coordinates and B) an 8 point algorithm.

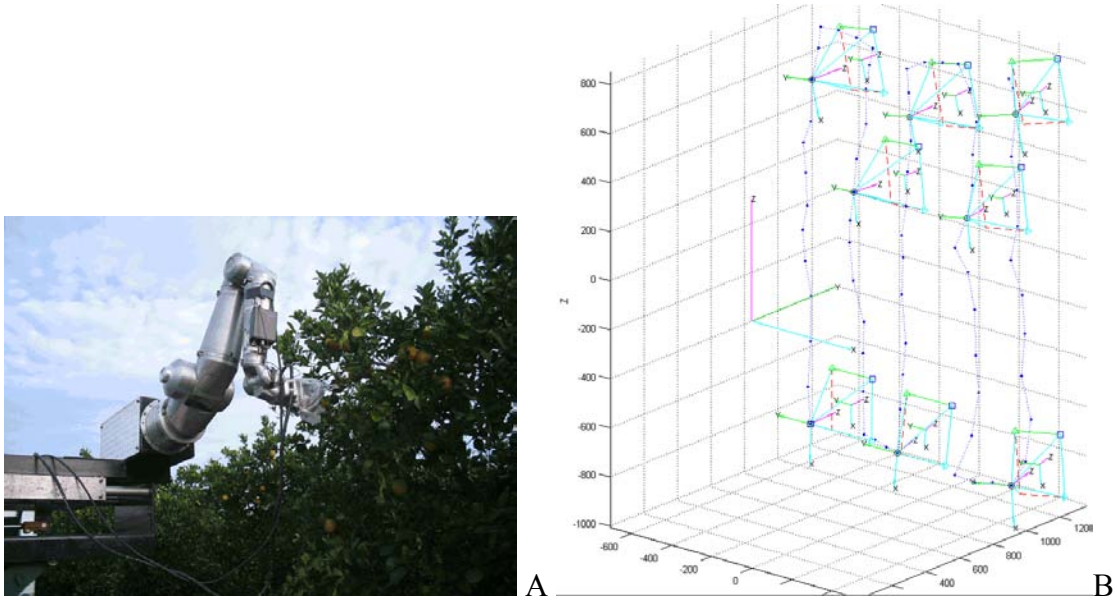


Figure 8-26. Outdoor experiment. A) Scanning using robot manipulator with B) zigzag camera motions.

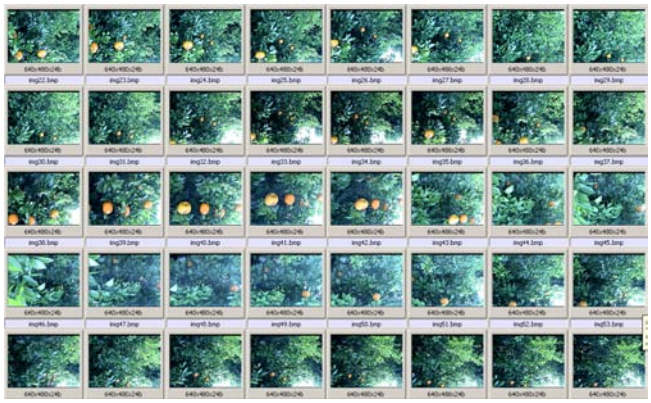


Figure 8-27. Captured sequential images.

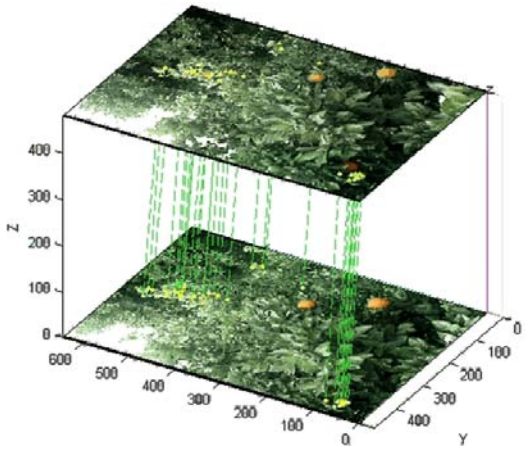


Figure 8-28. Matched correspondences between sequence images.

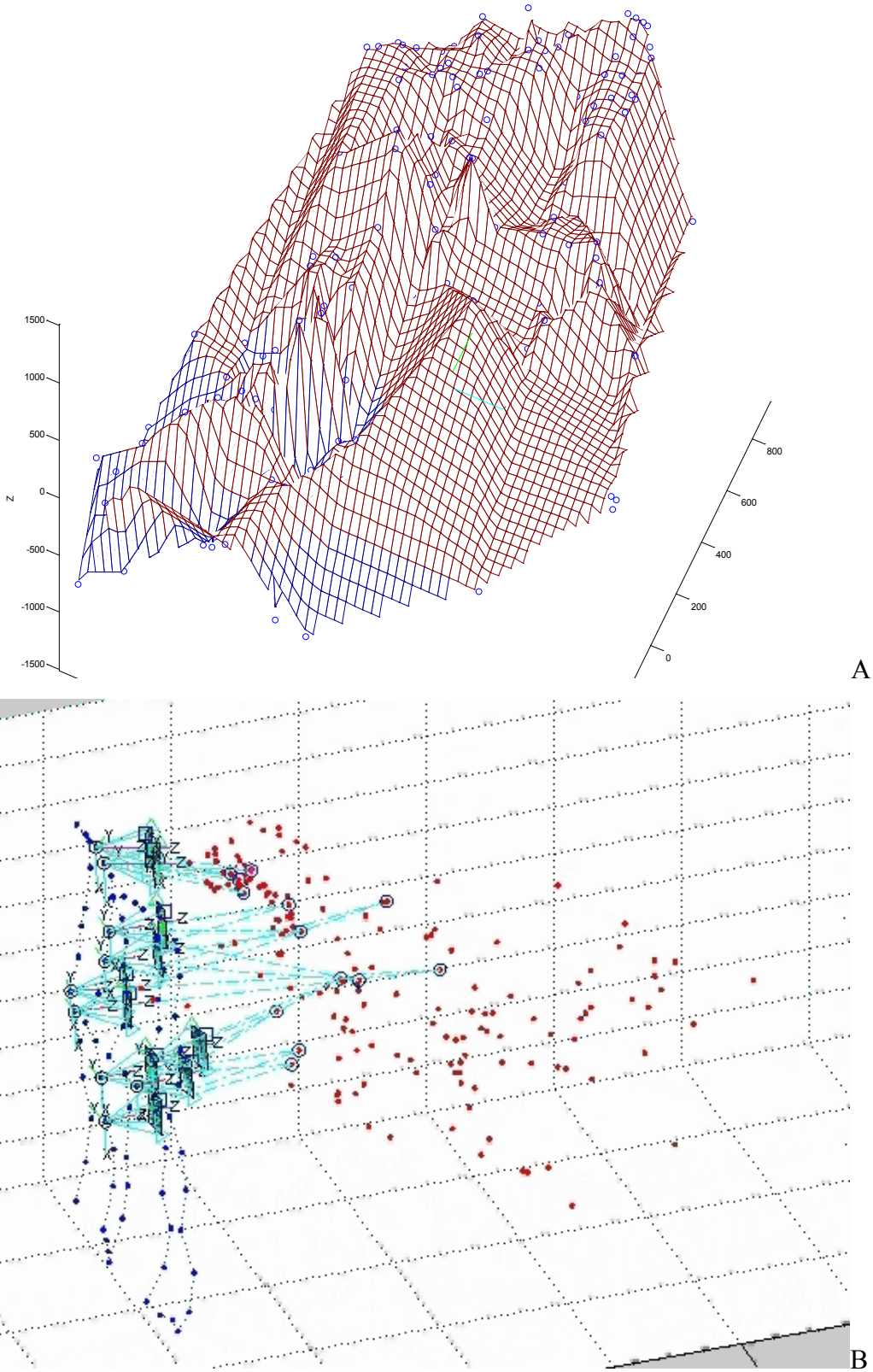


Figure 8-29. Reconstruction results. A) Approximated surface of citrus canopy. B) Reconstructed vertices.

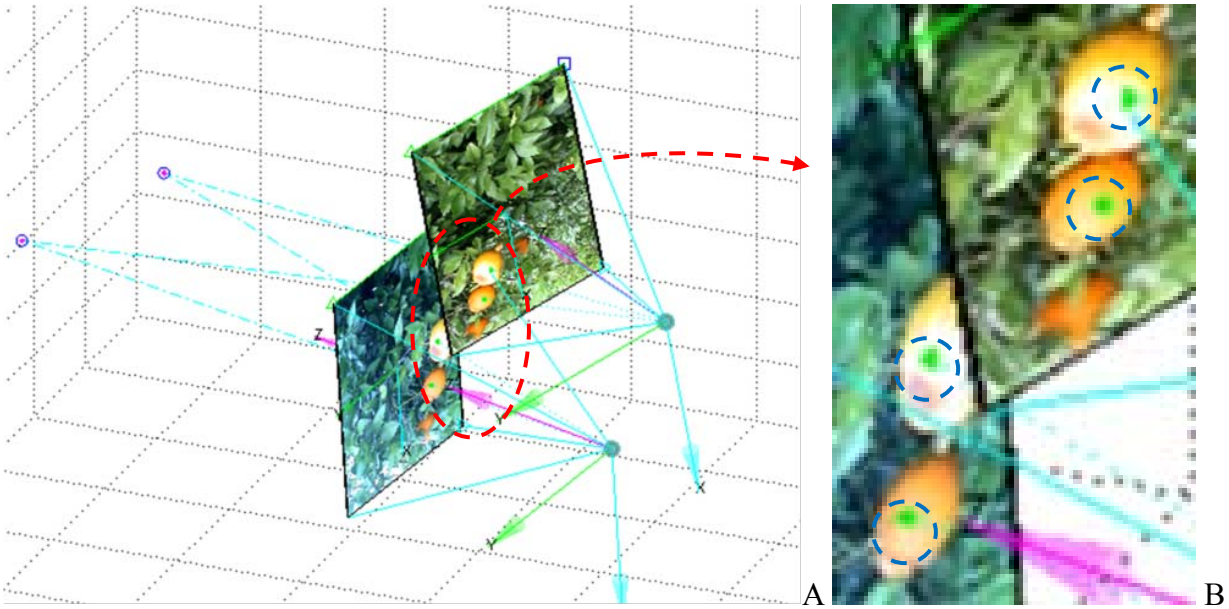


Figure 8-30. Reconstruction confirmation. A) Fruit positioning. B) Re-projection error.

CHAPTER 9
RANGE SENSOR BASED 3D RECONSTRUCTION

The objective of this chapter is to globally reconstruct canopies by combining an active range sensor (Laser Detection And Ranging, LADAR) and a Global Positioning System (GPS) receiver.

Introduction

In precision agriculture, to treat fields specifically, it is important to observe the variability. Remote-sensing is one monitoring method used for wide areas. There are a couple of approaches in remote-sensing. As compared with aerial or satellite-based remote-sensing systems, ground-based remote-sensing systems give more effective benefits such as detailed vision-based information about distributions of status like diseases or vigor observed closer at hand. However, this chapter discusses a technique that approximately measures the volume of canopies. A range sensor (LADAR) and a GPS receiver is used for measurement, and estimated data can be mapped on a global map of the geographic information system (GIS).

Methods and Procedures

Before conducting experiments, simulation software was developed. The simulation step is not only necessary to verify the algorithm, but helpful to analyze errors with actual field data. The entire procedure is shown in Figure 9-1.

Coordinates of LADAR and Vehicle

Scanning model of LADAR

A LADAR is an active range sensor which measures distances to objects by emitting laser beam pulses and scanning for reflected pulses in a counter-clockwise planar semicircle (180°) as shown in Figure 9-2 A) and B). Therefore, scanned data do not indicate 3D space information without manipulation. Furthermore, if the position or orientation of a laser source is consistently

altering dynamically, it makes estimation more difficult. A GPS receiver can provide only position information of a LADAR. To determine the orientation of a LADAR, it is assumed that both the position and the orientation of a LADAR are fixed on the rear axle of a vehicle, and it scans in the lateral direction with respect to the forward direction of a vehicle. Since the Y axis of a LADAR can be regarded as the normal vector of the path curve as shown in Figure 9-2 C), the orientation can be determined based on the path obtained from a GPS receiver.

Motion model of vehicle

Since a LADAR is capable of scanning the canopy on both sides of a vehicle, the vehicle moves along every two rows in spiral as shown in Figure 9-3. Since the speed of a vehicle is not very fast, the resolution of position information taken from a GPS receiver unit is too underpopulated compared with a LADAR. Therefore, the intermediate motions of a LADAR are interpolated by a spline curve fitting method with the assumption the position information received from a GPS receiver is accurate.

Simulation

On the assumption that measured data are ideally accurate, a simulation for range sensor-based reconstruction was conducted. This simulation has several steps. A path generation and distance measurement simulate GPS and LADAR information, respectively. A 3D reconstruction is achieved based on these data.

Virtual canopy model

For simulation, a virtual canopy was modeled using a mixed Gaussian. The mesh of the model was generated by Delaunay triangulation. Virtual canopies having different height are uniformly aligned in a grid. The pitch size between canopies is 4 [m]. The created virtual canopy model is illustrated in Figure 9-4.

LADAR simulation

A virtual canopy model consists of a number of triangular facets. When a laser beam passes through a facet, the distance between the beam source and intersecting point on the facet can be measured. According to Plücker coordinates system (see Appendix A), an intersection point $\mathbf{p}$ that a line $\{\mathbf{S}_1 \;;\; \mathbf{S}_{0L1}\}$ passes through a plane $\{\mathbf{S}_2 \;;\; d_2\}$ is determined by

$$\mathbf{p} = \frac{(\mathbf{S}_2 \times \mathbf{S}_{0L1}) - d_2\mathbf{S}_1}{\mathbf{S}_1 \cdot \mathbf{S}_2}. \tag{9-1}$$

Therefore the distance between the laser beam source and facet is $|\mathbf{p}|$. The intersecting point can also be placed on the outside of a given facet as shown in Figure 9-5. To discard these invalid points, the summation among these three angles between the intersecting point and three vertices of a facet is computed. When the intersecting point is located inside the facet, the summation of angles is equal to 360°. An angle between two lines $\{\mathbf{S}_1 \;;\; \mathbf{S}_{0L1}\}$ and $\{\mathbf{S}_2 \;;\; \mathbf{S}_{0L2}\}$ is solved by

$$\theta = \cos^{-1}\left(\frac{\mathbf{S}_1 \cdot \mathbf{S}_2}{|\mathbf{S}_1| \cdot |\mathbf{S}_2|}\right). \tag{9-2}$$

There could be several facets which a laser beam passes through. The closest facet from the laser source is regarded as an actual reflecting one. Since it is time-consuming to compute all of the triangles every time, facet which are placed within 25° upper and lower bounds with respect to scanning plane are selected as candidates to check in order to reduce the number of checking facets during iteration.

Motion simulation

Since a laser beam source is a moving part, its coordinates must be transformed. To set this configuration, the YXZ rotation transformation is used.

$$\mathbf{R} = \mathbf{R}_y \mathbf{R}_x \mathbf{R}_z , \tag{9-3}$$

where $\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\beta & \cos\alpha \end{bmatrix}$, $\mathbf{R}_y = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$, $\mathbf{R}_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

The direction of the beam $\mathbf{P}_{Path}$ at each position is determined by the following transformation.

$$\mathbf{P}_{Path} = \mathbf{RTP}_{LADAR} , \tag{9-4}$$

where $\mathbf{T}$ is a translation vector and $\mathbf{P}_{LADAR}$ is a laser beam vector. Figure 9-6 shows the coordinates at a scanning.

A real LADAR considered has 180° scanning angle, with 180 individual laser beams emitting from the source distributed evenly over the 180°. The LADAR has the capability of 5 scans within a second, however, this frequency is too high to reasonably calculate. A more manageable resolution and frequency for computation in this simulation was set to have a scanning angle of 60° and frequency of 12 scans in 0.5 [s].

To simplify the calculation in this simulation, a path equivalent to only one row is tested. Figure 9-7 shows a fully simulated scene. Notice that the intersecting points along the surface of virtual grove model are observed in Figure 9-7 B).

Simulated distances

The distance resulted in simulation is shown in Figure 9-8. X axis and Y axis represent the scanning angle and distance, respectively. Z axis in Figure 9-8 B) means time. The distance toward discarded points which is placed at no facet is set 0.

Results in reconstruction

The process of simulation is similar to the inverse process of the reconstruction algorithm. The model reconstructed by using the distance and motion information is shown in Figure 9-9, as

well as another figure combined with an original model is shown so as to observe how well it was reconstructed. The visual result shows that the algorithm works well. However, this result is obtained under the assumption that motion information is accurate.

Experiment

Configuration of equipments

The configuration of equipment used in the experiment is shown in Figure 9-10 A). As mentioned before, the Y axis of a LADAR was placed parallel to the real axle. Since the power generator sets up excessive vibration, it was place on the other cart to protect other devices from vibration.

Data acquisition software

To obtain data from equipments, a data-acquisition software was developed by referring to the previous simulation software. Time and position information from a GPS receiver is recorded at the first column to the third column, and then 180 number of LADAR data go after every second in order to synchronize data with respect to time. This software shown in Figure 9-10 displays receiving data and allows users to change several settings. This software was compiled by MFC 6.0.

Calibration of LADAR

Raw data obtained from LADAR does not represent physical range as it is, a range calibration was conducted with six sample lengths from one foot to six feet. X axis in Figure 9-11 B) denotes sample lengths and y axis denotes values taken from LADAR. Data from a LADAR showed almost linear variation. The metric unit was used in this experiment for compatibility with that of a GPS receiver.

Obtained raw data

A LADAR used in this experiment was able to measure up to 81 [m]. Since the maximum value means just empty space, those were discarded. Figure 9-12 B) shows position data obtained from a GPS receiver. Even though the vehicle was moved straightly, there are large errors as shown in the figure. x axis in the plot comes under Longitude and y axis comes under Latitude. In the computation, the origin position was reset right and bottom side of data set.

Adjustment of position data

Since these position data contains large errors, I was not able to make use of them as it is. So I manually reset the control points which fit to the curve smoothly. Regeneration of path and reconstruction was conducted at each separated row. The most rightward row in Figure 9-13 denotes the 1st row in this experiment. Red dots in are regenerated positions of the 1st row by using a Spline curve fitting method.

Results

Reconstruction 3D space data

Figure 9-14 shows the result in reconstruction of 4th row. Acquiring time was 110 [s] for the 4th row. It was necessarily to separate vertices in a range in order to divide between the canopy and ground. Points lower than 20 [cm] were colored with green regarding as the ground. Therefore, two canopies become distinctive.

Vertices around canopies are manually collected, and tetrahedrons are created by using a 3D delaunay triangular function as shown in Figure 9-15. A tetrahedron consists of four vertices or four facets. Figure 9-16 shows the entire reconstruction of a grove. The second canopy in the 3rd row looks like two trees. But there was no basis to separate adjacent canopies with obtained data. Even though the sampled grove was chosen for the purpose of salient result, there is little

gap between canopies in the general citrus grove. For automatic collection, additional information such as stem positions must be provided.

Calculation of volume of canopy

Once a tetrahedron is determined from the vertices, then the volume of each tetrahedron can be solved by

$$V = \frac{1}{3}Ah = \frac{1}{3}\frac{|\vec{a}\cdot\vec{b}|}{2}h = \frac{|\vec{a}\cdot\vec{b}\cdot\vec{c}|}{6}, \quad (8\text{-}10)$$

where $\vec{a},\vec{b},\vec{c}$ are adjacent edge vectors of a vertex. The entire volume of a canopy is summation of each volume of tetrahedron, and the height of a canopy is just the maximum Z axis value among vertices. The results of each canopy are shown in Table 8-1.

Representation on GIS software

To show an application about collected data, the estimated variabilities were mapped using an Arcview GIS software as shown in Figure 9-17. Figure 9-17 A) shows the location and path on satellite image and B) shows the distribution of the estimated variability map by choosing an inverse distance interpolation option. If more samples are used, more practical distribution will be obtained.

Conclusion

A remote-sensing technique associated with a ground-based scouting system was implemented. This system is designed to measure variability such as volume and height of citrus canopy using an active range sensor and a GPS. A simulation software contributed to saving the time to develop and verify the algorithm before experiment. A vertical-lateral scanning approach can successfully reconstruct 3D canopies and solve their volume and height. A LADAR is an effective range sensor to measure object's size. However, since the precision of result is sensitive

to motions, a more accurate positioning system is needed. If the center of a canopy was known on the map, the grouping vertices for volumetry could be done automatically. This system can also be applied to unmanned vehicle systems to automate the whole process of a ground-based remote-sensing.

Future Work

Additional devices may be needed when the boundary between canopies is ambiguous. For example, since range data do not tell between a canopy and other objects such as ground or empty space, vision information can be a reference to separate them. Basically, range information can be projected on the vision image through a calibration. If the boundary between a canopy and the other part is divided through the segmentation, range data can be grouped based on the segments. Then vertices associated with a canopy can be collected automatically.

Table 9-1. Estimated volume and height of canopy from reconstructed 3D space data.

4th row		3rd row		2nd row		1st row	
Volume[m^3]	Height[m]	Volume[m^3]	Height[m]	Volume[m^3]	Height[m]	Volume[m^3]	Height[m]
11.661	1.9476	11.079	1.5466	16.866	1.6399	14.288	2.0056
14.394	1.7248	64.874	2.73	13.279	2.5763	20.552	2.0231
				16.504	2.7476	26.576	2.705
				14.524	2.143		
				17.004	2.9033		

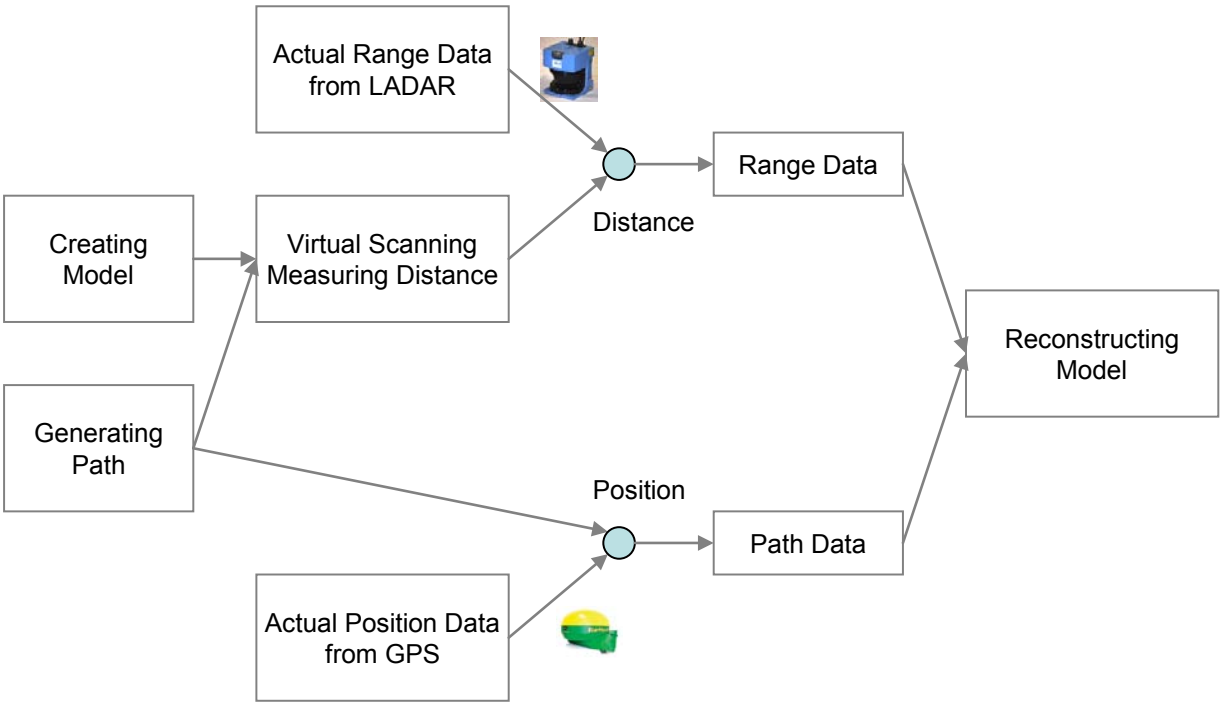


Figure 9-1. Procedure diagram.

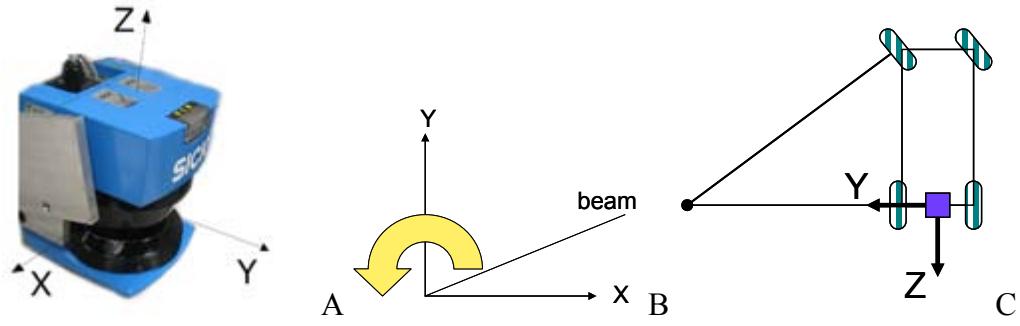


Figure 9-2. Coordinates of LADAR system. A) Coordinates of LADAR. B) Scanning direction. C) Coordinates of LADAR on a vehicle.

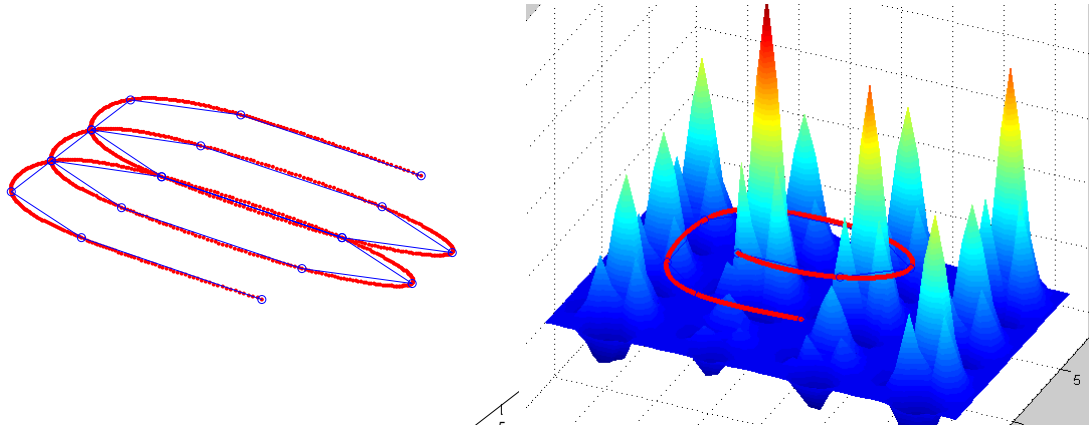


Figure 9-3. Path model of a vehicle in a virtual grove.

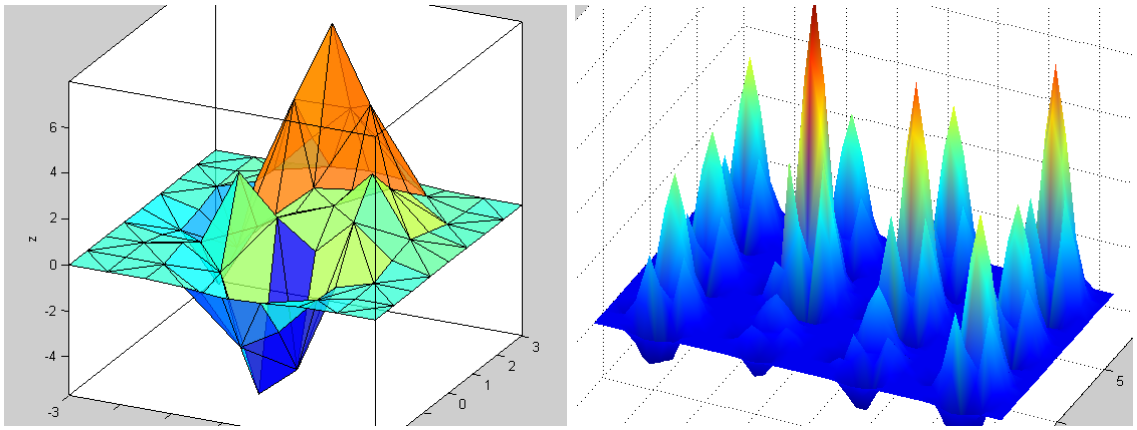


Figure 9-4. Virtual canopy model. A) Unit virtual canopy. B) Virtual grove.

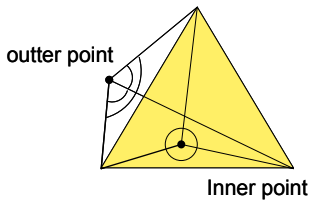


Figure 9-5. Search for a facet which a laser beam passes.

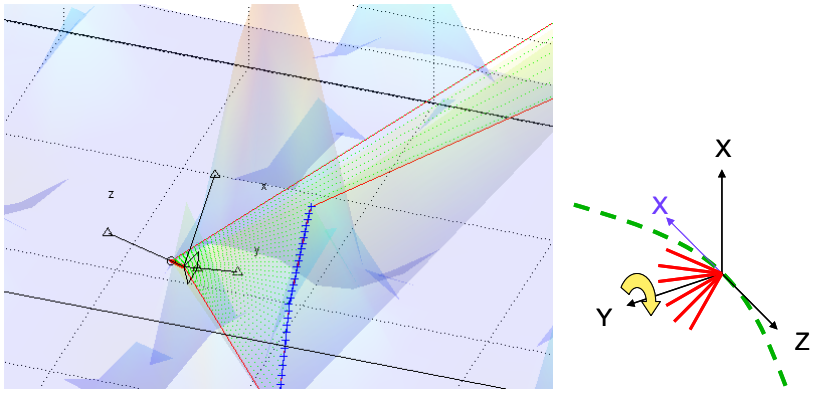


Figure 9-6. Coordinates of LADAR during simulation.

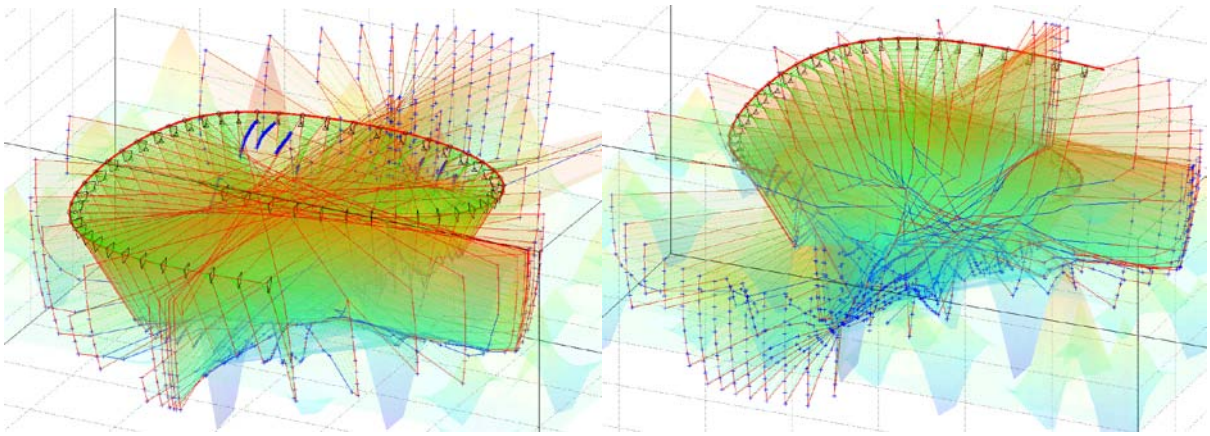


Figure 9-7. Fully scanned visual simulation.

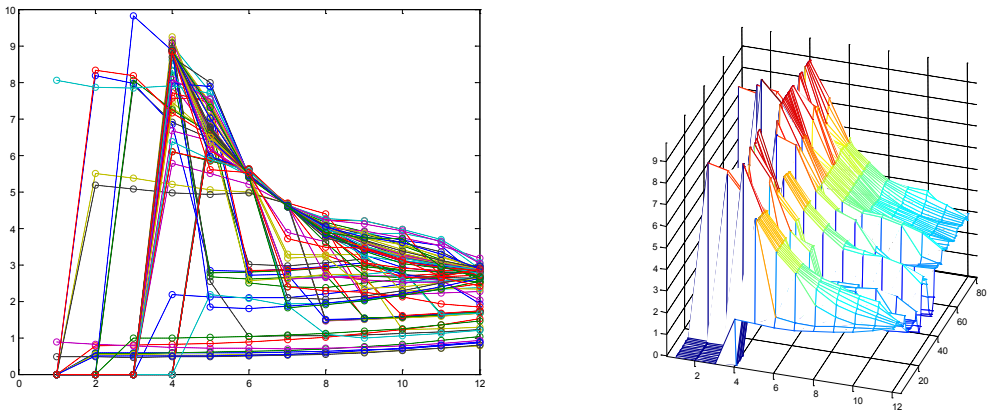


Figure 9-8. Distances measured from simulation.

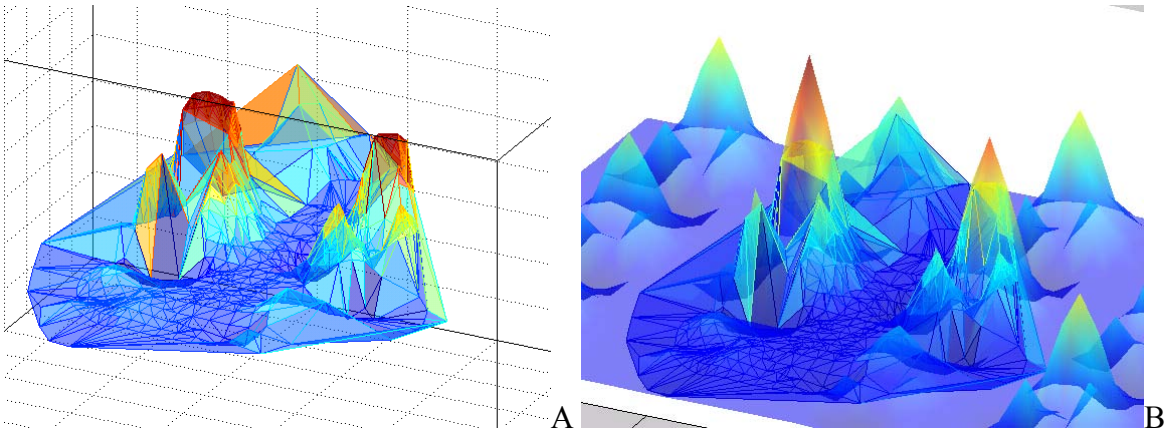
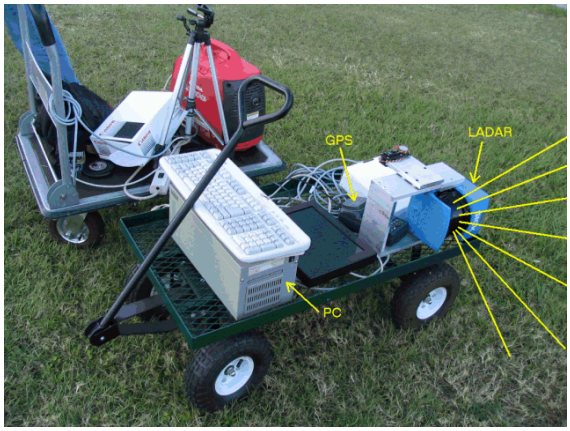
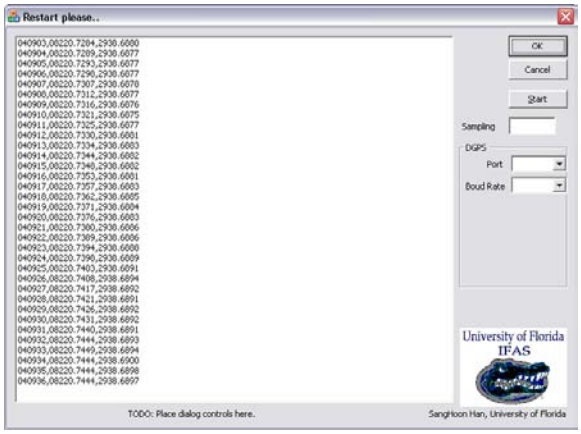


Figure 9-9. Reconstruction results. A) Reconstructed model. B) reconstructed model fitting to the original virtual grove.

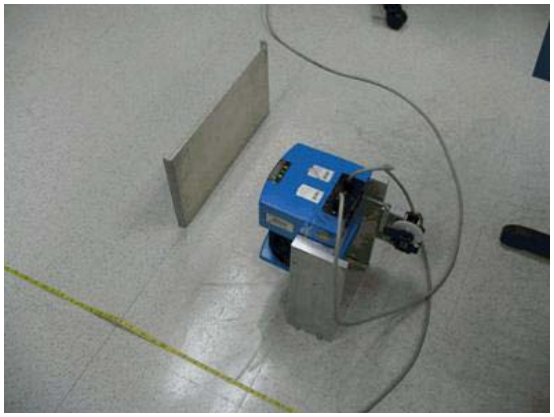


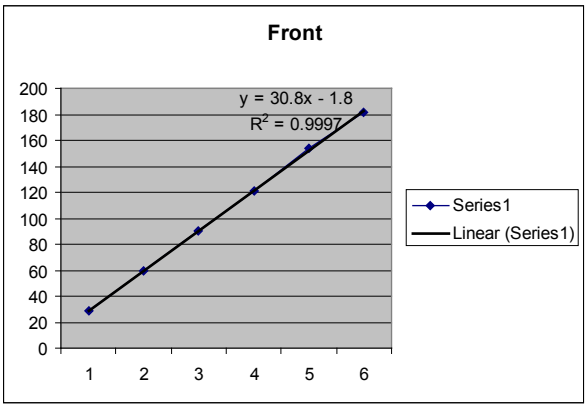
A



B

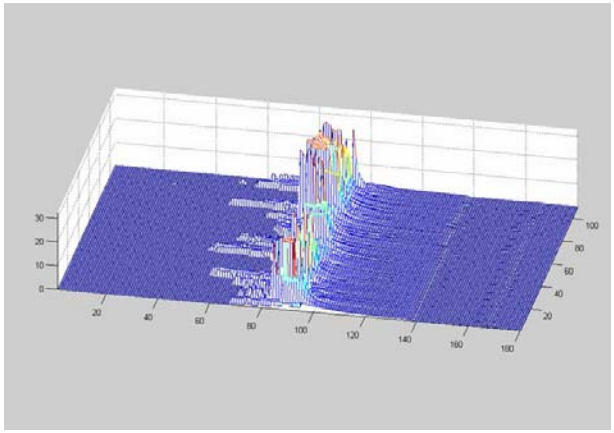
Figure 9-10. Configuration of LADAR and GPS system. A) System layout. B) Data acquisition software.

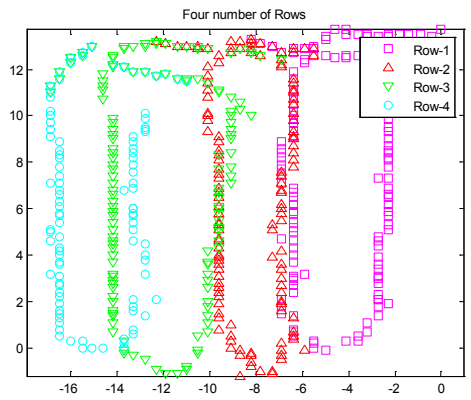

A



B

Figure 9-11. LADAR calibration. A) Calibration of LADAR. B) Linearity of range.


A



B

Figure 9-12. Raw data. A) Raw range data of 4th row acquired from LADAR. B) Row position data acquired from GPS.

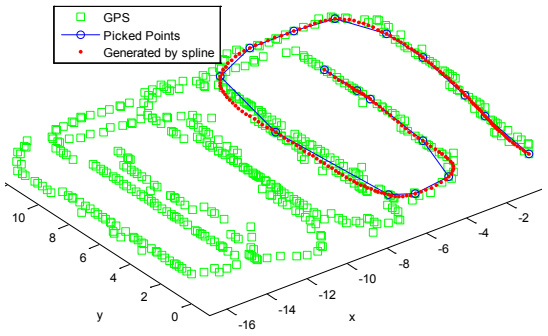


Figure 9-13. Adjusted control points for path.

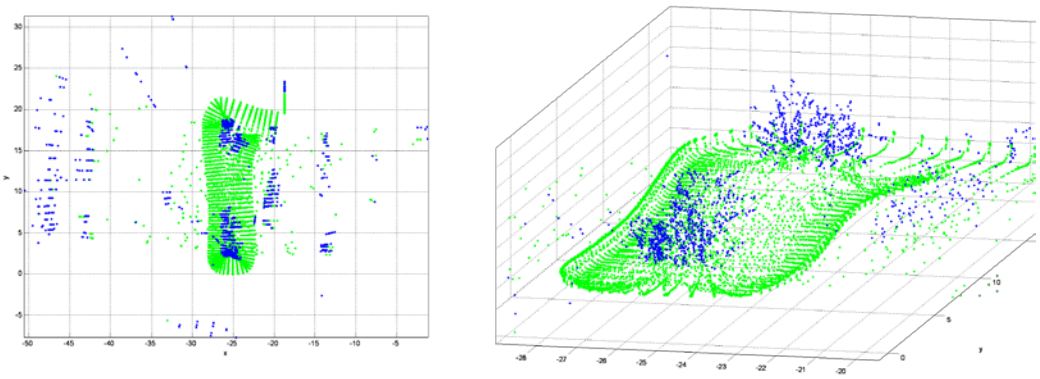


Figure 9-14. Reconstructed 3D space data in the 4th row.

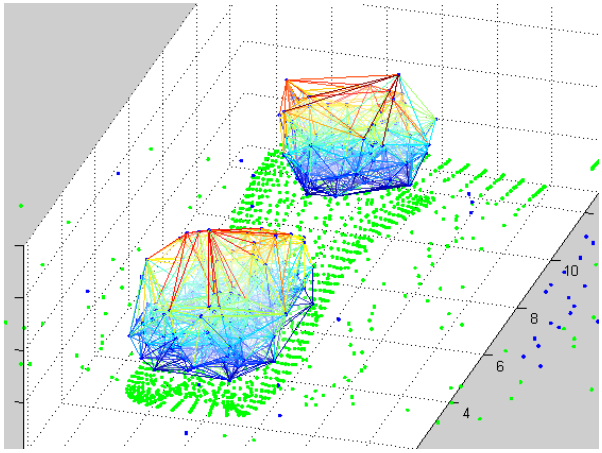


Figure 9-15. Created tetrahedrons corresponding to canopy.

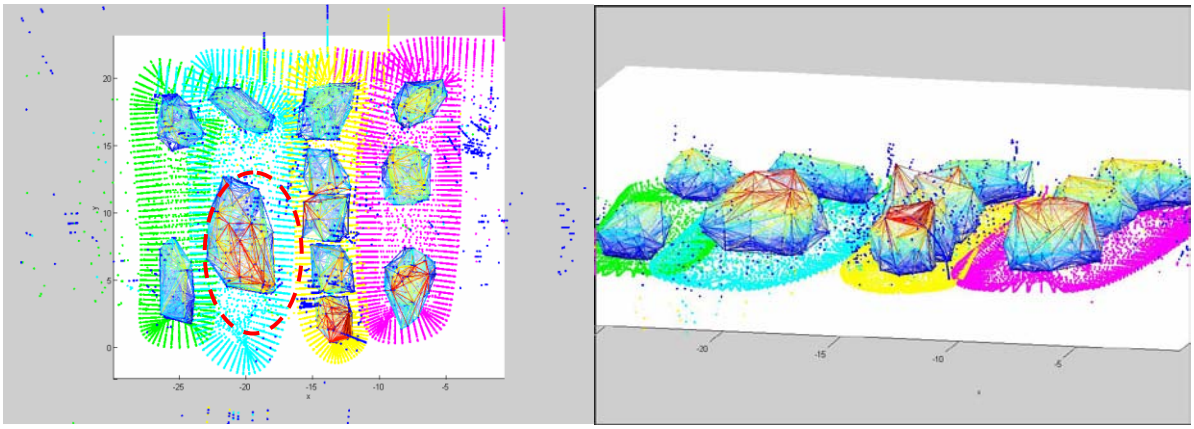
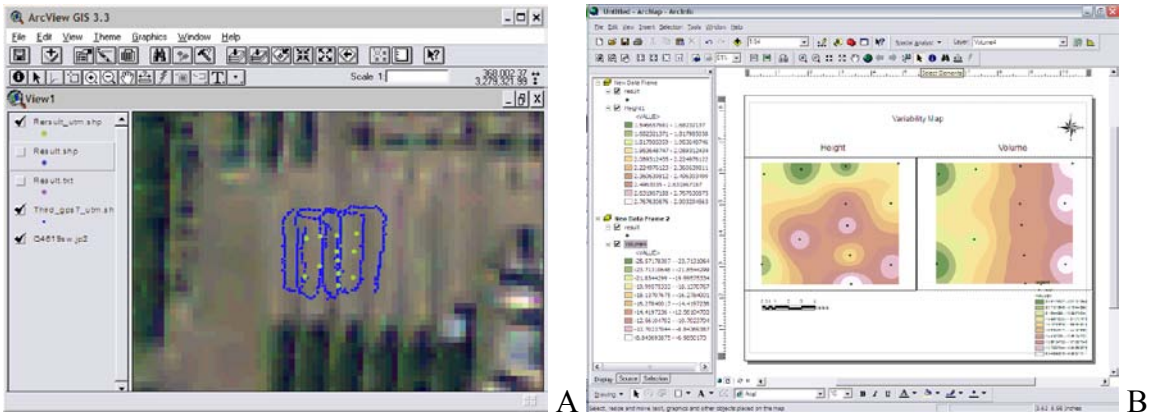


Figure 9-16. Reconstructed entire grove.



A B

Figure 9-17. Results applied to GIS software. A) Location. B) Variability map.

CHAPTER 10
CONCLUSIONS AND FUTURE WORK

The objective of this chapter is to summarize and conclude my study.

Summary and Conclusions

Corners extracted from canopy scenes which have no artificial landmarks are unstable when adding a variety of disturbances such as backlight or illumination. Thus the longevity of features is prone to become short when the camera moves. To strengthen the role of features, a leaf detection and a multilayered tracking method were invented, which are designed specially for canopy scenes.

The key point of leaf detection is to segment leaf areas through the combination of morphological operations. Once segmented, the center of a leaf in each segment is set as a feature point. Since shadowed leaves are not sustainable, leaf detection was designed to apply discriminating thresholds to each segment to secure as many features as possible.

In cases where the intensities of some leaves are similar, segment boundaries become ambiguous. Thus choosing the center of leaf varies among frames. To be insensitive to these sudden changes of the feature point, an active mesh feature tracking method was employed. Even though an active mesh is insensitive to sudden changes, it requires exhaustive computation and it is possible for wrong features to disturb normal features. Therefore, a multilayered active tree method was invented which expands 2D meshes into 3D meshes. The creation of multiple layers is based on coarse-to-fine images by means of Gaussian filtering. Features extracted from each image recursively form hierarchical structures by connecting sub-features within an area which is segmented by a Voronoi diagram. Since a multilayered active tree can evaluate the strength of a feature, it can control the disturbance from unnecessary features. Subsistence of features was

evaluated with the comparison of optical flow. The optical flow through these methods results in insensitivity to noise.

A parallax can be seen around the edges of an image. Furthermore, a parallax occurs in the horizontal movement model. Thus a cropping edge mosaicing method was suggested. It was designed to automatically determine the cropping area based on the direction of optical flow. Cropping area is blended by controlling an alpha channel. A cropping edge mosaicing method showed that it can minimize the discordance caused by the parallax effect. One critical point that must be considered in the lengthy mosaic is that a camera alignment must be parallel to the direction of movement. Otherwise, the mosaic will go past the upper or lower boundaries.

For a 3D canopy model, local reconstruction was conducted by robot vision and global reconstruction was conducted by a range sensor vehicle. Since it is difficult to accurately measure dimensions of an unknown structure, the accuracy of reconstruction is evaluated with re-projection error. In the vision-based reconstruction, an 8 point algorithm was tried first, which is widely known. However, since an 8 point algorithm requires over 8 input points, it was concluded that those coupled input points are quite sensitive to noise. In addition, since a reconstructing object is larger than the view in the local reconstruction, accumulation of error is inevitable without a ground truth. Instead, a Plücker coordinates system is employed for reconstruction, which employs a simpler computation method than an 8 point algorithm. Since a Plücker coordinates system independently conducts the reconstruction with one pair of correspondences with respect to given motion, error input does not influence the existing reconstruction. The robot used for reconstruction provides repeated constant motion information of the end-effector. Thus a Plücker coordinates system was valid. According to experiments, even though the motion information seems to have some errors, the re-projection error was

adequately small. Therefore, reconstruction by a Plücker coordinates system was successful with the given motions. For reconstruction with a range sensor, a combination with LADAR and GPS was used. Since a LADAR scans only 2D, additional information is needed to achieve 3D space. A GPS is used to provide vertical direction information orthogonal to the scan plane of the LADAR. Since the resolution of GPS information is underpopulated compared to that of LADAR, the sparse positions and orientations are interpolated with a Spline fitting method. The volume of reconstructed canopies were measured and represented on a global map. Contributions within my study about each topic are shown in Table 10-1.

Future Work

Leaf detection was carried out with an image. This approach will be applicable to multilayered successive images. Watershed segmentation could be also used to segregate fruits from leaves. For active meshes, fixed parameters were used. If parameters can be adjusted automatically, it will be a more intelligent tracking method. In the estimation of camera motion, accumulating errors could not be converged. For a motion model in the open system, since there is no ground truth, accumulation of error is essentially inevitable. To be within an allowable range, a high performance camera, a fast computing system and a limited camera motion must be considered. A LADAR range sensor is quite accurate, but the range information does not tell which is ground or empty space. If range data are combined with vision data, it will yield more effective results. When applying a range sensor, precise positioning and calibration are needed. A vehicle system simulation will be able to test in the implemented virtual space. All topics in my study are tested independently. In the next work, all procedures will be designed to work at once automatically in real-time. Collected and processed information about the surface of the canopy and fruit positions can be represented in the virtual environment as shown in Figure 10-1 so that farmers can observe and control the real situation remotely.

Table 10-1. Contribution techniques.

Topic	Main Technique	Contribution - Level
Feature detection	Leaf detection	novel invention – normal
Feature tracking	Multilayered active tree	novel invention – complex
Image mosaicing	Crop-off composition	novel trial – normal
3D reconstruction (Vision)	Plücker coordinates system	novel trial – normal
3D reconstruction (LADAR)	Vertical scanning	trial – simple

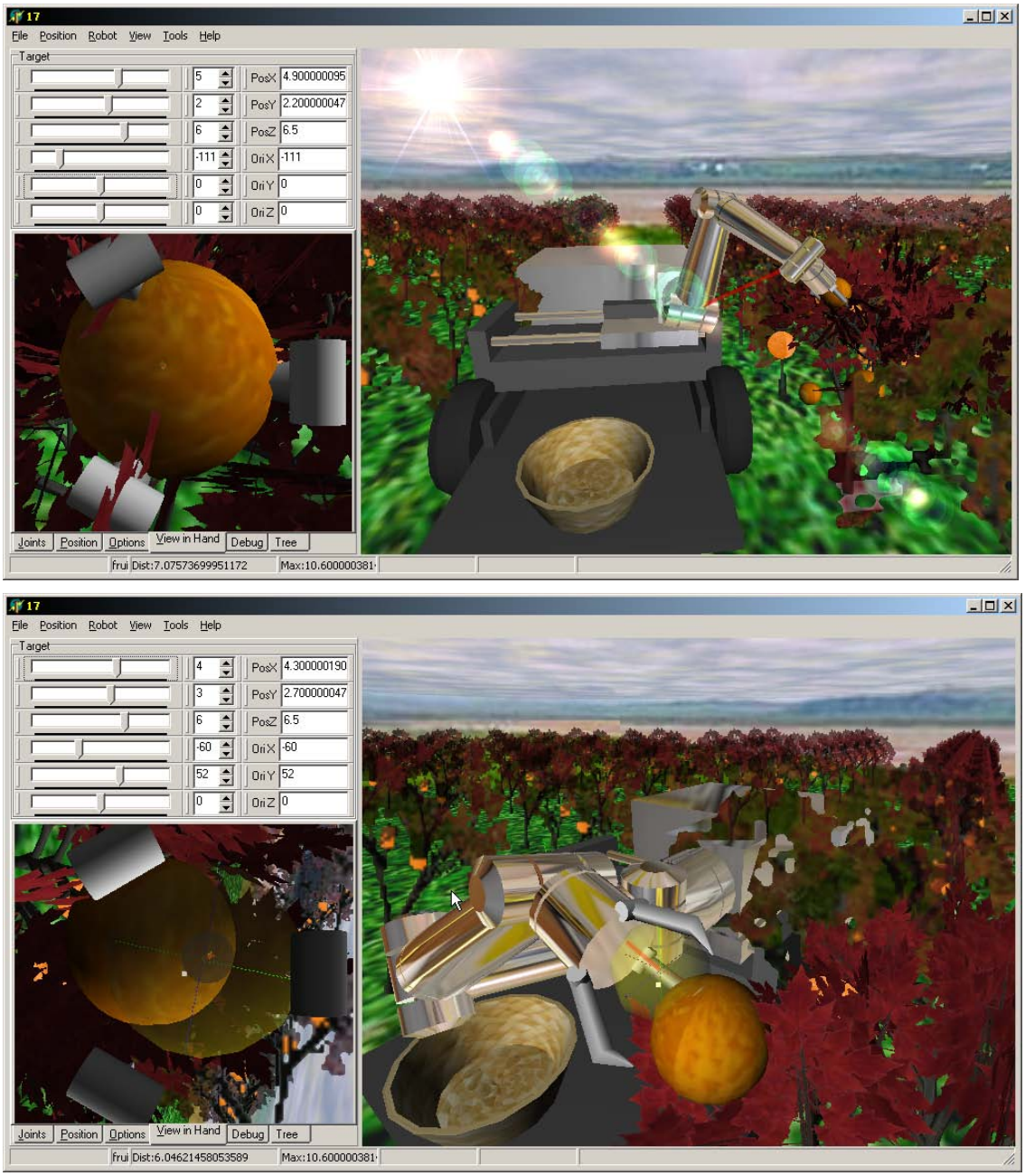


Figure 10-1. Virtual robot harvesting simulation.

APPENDIX A
GEOMETRY

Plücker Coordinates System

Definitions

Point

A point in homogeneous expression is

$$\mathbf{p} = (x, y, z \quad ; \quad w) \tag{A-1}$$

In this dissertation, a point is regarded as $\mathbf{p} = (x, y, z)$, i.e. $\mathbf{p} = (x, y, z \quad ; \quad 1)$.

Line

A line in Plücker coordinates is defined as follows.

$$\mathbf{L} = \{\mathbf{S} \quad ; \quad \mathbf{S}_{0\mathrm{L}}\} \tag{A-2}$$

where $\mathbf{S}$ is a normalized vertor between two points, and $\mathbf{S}_{0L}$ is a moment with respect to an origin. $\mathbf{S}$ and $\mathbf{S}_{0L}$ are determined by given points $\mathbf{p}_1$ and $\mathbf{p}_2$.

$$\mathbf{S} = \frac{\mathbf{p}_2 - \mathbf{p}_1}{|\mathbf{p}_2 - \mathbf{p}_1|}, \ \mathbf{S}_{0L} = \mathbf{p}_2 \times \mathbf{S} \tag{A-3}$$

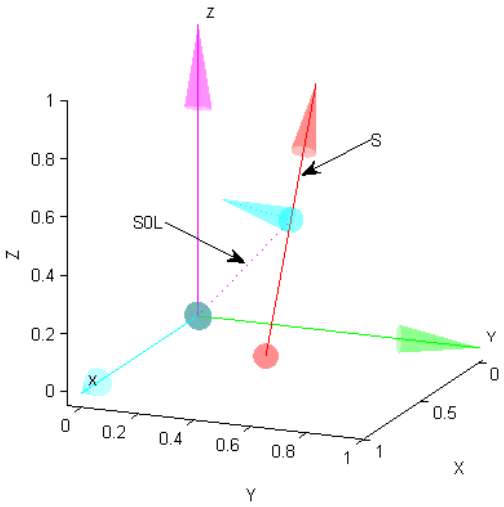


Figure A-1. A line definition in Plück coordinates system.

Plane

A plane is defined by

$$\mathbf{N} = [\mathbf{S} \quad ; \quad d] \tag{A-4}$$

Given three points $\mathbf{p}_1 = (x_1, y_1, z_1), \mathbf{p}_2 = (x_2, y_2, z_2), \mathbf{p}_3 = (x_3, y_3, z_3)$, a plane which a point and a line makes is

$$\mathbf{S} = \mathbf{S}_{0l12} - (\mathbf{p}_3 \times \mathbf{S}_{12}), \; d = -\mathbf{p}_3 \cdot \mathbf{S}_{0l12} \tag{A-5}$$

Angle

A sign between two lines which meet each other

Given two lines $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$ and $\mathbf{L}_2 = \{\mathbf{S}_2; \mathbf{S}_{0L2}\}$, a sign of the angle between them is

$$sign(\theta) = sign(\mathbf{S}_1 \times \mathbf{S}_{0L2} + \mathbf{S}_2 \times \mathbf{S}_{0L1}) \tag{A-6}$$

An angle between two lines which meet each other

Given two lines $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$ and $\mathbf{L}_2 = \{\mathbf{S}_2; \mathbf{S}_{0L2}\}$, an angle between them is

$$\theta = sign(\theta) \cos^{-1}(\mathbf{S}_1 \cdot \mathbf{S}_2) \tag{A-7}$$

An angle between two planes

Given two planes $\mathbf{N}_1 = [\mathbf{S}_1; d_1]$ and $\mathbf{N}_2 = [\mathbf{S}_2; d_2]$, an angle between them is

$$\theta = \cos^{-1}\left(\frac{\mathbf{S}_1 \cdot \mathbf{S}_2}{|\mathbf{S}_1| \cdot |\mathbf{S}_2|}\right) \tag{A-8}$$

Point

A point where a line passes though a plane

Given a line $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$ and a plane $\mathbf{N}_2 = [\mathbf{S}_2; d_2]$, a point where a line passes though a plane is

$$\mathbf{p} = \frac{(\mathbf{S}_2 \times \mathbf{S}_{0L1}) - d_2 \mathbf{S}_1}{\mathbf{S}_1 \cdot \mathbf{S}_2} \tag{A-9}$$

The angle between the line and the plane is.

$$\theta = \cos^{-1}\left(\frac{\mathbf{S}_1 \cdot \mathbf{S}_2}{|\mathbf{S}_1| \cdot |\mathbf{S}_2|}\right) - \frac{\pi}{2} \tag{A-10}$$

A point on a plane which is closest to the point

Given a point $\mathbf{p}_1 = (x, y, z)$ and a plane $\mathbf{N}_2 = [\mathbf{S}_2; d_2]$, a point on the plane closest to the point is

$$\mathbf{p} = \frac{-(d_2 + (\mathbf{p}_1 \cdot \mathbf{S}_2))\mathbf{S}_2}{\mathbf{S}_2 \cdot \mathbf{S}_2} + \mathbf{p}_1 \tag{A-11}$$

A point on a line which is closest to the point

Given a point $\mathbf{p}_1 = (x, y, z)$ and a line $\mathbf{L}_2 = \{\mathbf{S}_2; \mathbf{S}_{0L2}\}$, a point on the plane closest to the point is

$$\mathbf{p} = \frac{\mathbf{S}_2 \times (\mathbf{S}_{0L2} - (\mathbf{p}_1 \times \mathbf{S}_2))}{\mathbf{S}_2 \cdot \mathbf{S}_2} + \mathbf{p}_1 \tag{A-12}$$

A point on a line which is closest to a line

Given two lines $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$ and $\mathbf{L}_2 = \{\mathbf{S}_2; \mathbf{S}_{0L2}\}$, a point on each line which is closest to the other line is

$$\begin{aligned} d &= -\frac{\mathbf{S}_1 \cdot \mathbf{S}_{0L2} + \mathbf{S}_2 \cdot \mathbf{S}_{0L1}}{\sin(\cos^{-1}(\mathbf{S}_1 \cdot \mathbf{S}_2))} \\ \mathbf{p}_1 &= \frac{\mathbf{S}_{0L1} \times (\mathbf{S}_{0L2} - d((\mathbf{S}_1 \times \mathbf{S}_2) \times \mathbf{S}_2))}{\mathbf{S}_2 \cdot \mathbf{S}_{0L1}} \\ \mathbf{p}_2 &= d(\mathbf{S}_1 \times \mathbf{S}_2) + \mathbf{p}_1 \end{aligned} \tag{A-13}$$

Where d is a distance between two points.

Line

A line passing through two lines with the shortest distance

Given two lines $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$ and $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$, a line passing through two lines with the shortest distance is

$$\mathbf{L} = \left\{ \frac{\mathbf{S}_1 \times \mathbf{S}_2}{|\mathbf{S}_1 \times \mathbf{S}_2|} \quad ; \quad \mathbf{p}_1 \times (\mathbf{S}_1 \times \mathbf{S}_2) \right\} \tag{A-14}$$

Where $\mathbf{p}_1$ is a point on line $\mathbf{L}_1$ which is closest to the other line.

A line made by two planes

Given two point $\mathbf{N}_1 = [\mathbf{S}_1; d_1]$ and $\mathbf{N}_2 = [\mathbf{S}_2; d_2]$, a line made by two planes is

$$\mathbf{L} = \{\mathbf{S}_1 \cdot \mathbf{S}_2 \quad ; \quad d_{01}\mathbf{S}_2 - d_{02}\mathbf{S}_1\} \tag{A-15}$$

Plane

A plane made by a point and a line

Given a point $\mathbf{p}_1 = (x, y, z)$ and a line $\mathbf{L}_2 = \{\mathbf{S}_2; \mathbf{S}_{0L2}\}$, a plain made by a point and a line is

$$\mathbf{N} = [\mathbf{S}_{0L2} - (\mathbf{p}_1 \times \mathbf{S}_2) \quad ; \quad -\mathbf{p}_1 \cdot \mathbf{S}_{0L2}] \tag{A-16}$$

A plane made by two lines

Given a point two lines $\mathbf{L}_1 = \{\mathbf{S}_1; \mathbf{S}_{0L1}\}$ and $\mathbf{L}_2 = \{\mathbf{S}_2; \mathbf{S}_{0L2}\}$, a plane made by two lines is

$$\mathbf{N} = [\mathbf{S}_1 \times \mathbf{S}_2 \quad ; \quad -\mathbf{S}_{0L1} \cdot \mathbf{S}_2] \tag{A-17}$$

Basic Functions

Triangle Functions

Angles in triangle when edges are known

Given edges l_i, angles are

$$S = \frac{1}{2}\sum_{k=1}^{3} l_k$$

$$\theta_i = \sin^{-1}\left(\frac{2\sqrt{S\prod_{k=1}^{3}(S-l_k)}}{\prod_{k=1}^{3} l_k}\right). \qquad \text{(A-18)}$$

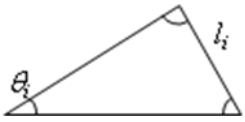


Figure A-2. Angles in triangle when edges are known.

Volume of Tetrahedron

Given edges l_i, the volume is

$$V = \frac{1}{3}Ah = \frac{1}{3}\frac{|\vec{a}\cdot\vec{b}|}{2}h = \frac{|\vec{a}\cdot\vec{b}\cdot\vec{c}|}{6}. \qquad \text{(A-19)}$$

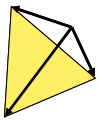


Figure A-3. Volume of a tetrahedron.

APPENDIX B
CRITICAL SOURCE CODES OF ALGORITHM

Leaf Detection

```
function [ Im_r ] = segment_detect( Im_ )

%>> [ Im_r ] = segment_detect( Im_ )
%
%(c) copyright SangHoon Han, ARMg UFL 3/26/2008

%% ------------------------------------------------------------------------------------
if nargin<1, return;  end, [ h, w, c ] = size(Im_); Im_r = Im_; Im = Im_;
%% ------------------------------------------------------------------------------------
Im1 = im1_im(Im_);
opIm = imnorm(imopen(Im_, strel('disk',8))); % imshow(opIm); oIm = imnorm(imshow_open( Im1,8 ));
flIm = opIm;
% flIm = imfilter(opIm,  fspecial('disk',8), 'replicate'); % [ wIm, bIm, Ims ] = imshow_watershed(-oIm); %
imshow(Im_r);
[ wIm, Ims ] = watershed_(-flIm); bIm = double(wIm == 0);
sgArs = [];
for i=1:length(Ims),
   sgAr = regionprops(Ims{i},'Area');
   sgArs = [sgArs sgAr.Area];
end
msgAr = mean(sgArs);
%%
imPnt_r = []; i=55; th_r = []; Im_r = bIm;
opIm1 = imnorm(imopen(Im_, strel('disk',4)));
for i=1:length(Ims), %disp(i);
   sgImi = Ims{i}; % imshow(Ims{55});
   sgAr = regionprops(sgImi,'Area');
%     if sgAr.Area<(msgAr*0.2)  , continue; end;
%     if ~tf_ratio(sgImi,  0.2), continue; end;

   mlIm = immultiply(opIm, sgImi); % imshow(mlIm);  mlIm = imshow_multiply(oIm, sgImi); %

   th = 0; sm = 0;
   while th<1
      bwIm = double(im2bw(mlIm, th)); % imshow(bwIm);
      sm = sum(sum(bwIm))/sum(sum(sgImi));
      if sm<.25
         bwCnt = regionprops(bwIm,'Centroid'); % imshow(sgImi);
         if ~isempty(bwCnt),
            imPnt_r = [imPnt_r floor(bwCnt.Centroid')];
         end;
         %           [ bPnts, bdIm ] = bwboundaries(bwIm,'noholes');
         %           [L,num] = bwlabel(opIm1);
         %           lb = Label2rgb(L);
         %           if ~isempty(bPnts)
         %              bPnt1 = bPnts{1};
         %              mnPnt = floor(mean(bPnt1));
         break;
         %
         % %              lbIm = double(label2rgb(bdIm, 'jet', 'w', 'shuffle'));
```

```
            % %                for j=1:length(lbIm)
            % %                   bPnti = bPnts{i};
            % %                end
            %               %            cPnt = floor(sgAr.Centroid');
            %            end
        end
        th = th+0.01;
    end
    Im_r = imadd(Im_r, bwIm);
end
Im_r(:,:,3) = imnorm(bIm);
Im_r(:,:,2) = im_plot(bIm, imPnt_r, 'y+',1);
% imshow(Im_r);
% Im_r = im_plot(oIm,imPnt_r,'+',2);


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function segment_detect_demo() % clear
%%
imf_ = 'garden1_cinepak_filtered0000.bmp';
Im_ = imshow_file(imf_); Im_ = im1_im(Im_);
%%
clc; clf; [ Im_r ] = segment_detect( Im_ );
imshow(Im_r);
```

Multilayered Active Tree Tracking

Active Mesh

```
function [ ePnt2_r ] = mesh_track( oPnt2s_r )

% match images.
% > Im
%>> [ crPnt_r ] = track_mesh( prPnt_, crPnt_, Fct_ )
%
%(c) copyright SangHoon Han, ARMg UFL 4/7/2007

%% ---------------------------------------------------------------------------------------
if nargin<1, return; end
%% ---------------------------------------------------------------------------------------
Pnt21 = oPnt2s_r{1};
tri = delaunay(Pnt21(1,:), Pnt21(2,:));
trii = [tri tri(:,1)];
ngPnts = mesh_neighbor(tri);
%% wrong input
wPnt2s_r = oPnt2s_r;

%% combine forces
clc;
ngPnts = mesh_neighbor(tri);
F_ = []; F_tot =[];
Pnt2_1=wPnt2s_r{1};
Pnt2=wPnt2s_r{1};
pTime=now;
sz2 = size(oPnt2s_r,2);
for k=2:sz2, pTime = datestr_(pTime, k, sz2);
```

```
    Pnt2_k_1 = wPnt2s_r{k-1};
    Lprev = [];
    for i=1:size(Pnt2,2) % for each node
       ngPnt = ngPnts{i};
       for j=1:size(ngPnt,2),
          nj = ngPnt(j); %disp([k,i,nj]);
          Pnt2_k_1nj = Pnt2_k_1(:,nj);
          if norm(Pnt2_k_1nj)==inf,
             Pnt2_k_1nj = Pnt2_1(:,nj); end
          Pnt2_k_1i = Pnt2_k_1(:,i);
          if norm(Pnt2_k_1i)==inf,
             Pnt2_k_1i = Pnt2_1(:,i); end
          dlen = Pnt2_k_1nj-Pnt2_k_1i;
          Lprev(i,nj) = norm(dlen);
       end
    end
    Pnt2_k = wPnt2s_r{k};
    for i=1:size(Pnt2,2) % for each node
       Pnt2_k_i = Pnt2_k(:,i);
       ngPnt = ngPnts{i};
       slen = 0;
       for j=1:size(ngPnt,2),
          nj = ngPnt(j);
          slen = slen+Lprev(i,nj);
       end
       F_ = [];
       for j=1:size(ngPnt,2),
          nj = ngPnt(j); %disp([k,i,nj]);
          Pnt2_k_nj = Pnt2_k(:,nj);
          if norm(Pnt2_k_nj)==inf,
%               Pnt2_k_nj = F_tot{k-1}(:,nj); end
             Pnt2_k_nj = Pnt2(:,nj); end
          Pnt2_k_1nj = Pnt2_k_1(:,nj);
          if norm(Pnt2_k_1nj)==inf,
             Pnt2_k_1nj = Pnt2_1(:,nj); end
          Fext = Pnt2_k_nj-Pnt2_k_1nj; % external force

          if norm(Pnt2_k_i)==inf,
             Pnt2_k_i = Pnt2(:,i); end
          dPnt2 = Pnt2_k_nj-Pnt2_k_i; % internal length
          Lcurr = norm(dPnt2);
          if Lcurr==0,
             Lcurr = 1; end
          w = (Lprev(i,nj)-Lcurr)/Lcurr;
          Fint = w*dPnt2;

          if slen==0,
             slen = 1; end
          beta = (1-Lprev(i,nj)/slen)/(size(ngPnt,2)-1);
          F_{k-1,i}(:,nj) = beta*(0.95*Fext-0.05*Fint);
       end
       sumF = sum(F_{k-1,i},2);
       Pnt2_k_i = Pnt2_k(:,i);
       if norm(Pnt2_k_i)==inf,
          Pnt2_k_i = Pnt2(:,i); end
       Pnt2_k_1i = Pnt2_k_1(:,i);
```

```
        if norm(Pnt2_k_1i)==inf,
           Pnt2_k_1i = F_tot{k-2}(:,i); end
        alph = 1;
        alph = norm(Pnt2_k_i-Pnt2_k_1i)/norm(sumF);
        if alph<=1e-5, alph=1; end
        F_tot{k-1}(:,i) = alph*sumF+Pnt2(:,i);
    end
    Pnt2_1 = Pnt2;
    Pnt2 = F_tot{k-1};
end
%% F_ext{1,1}
% clf; plot_(oPnt2s_r{3}(:,1),'b+'); hold on;
% plot_(F_tot{1}(:,1),'ro'); hold off
%% plot combine force
nf = size(F_tot{1},2); np = size(F_tot,2);
tw = []; twp = []; err = [];
c = 'kgrbc'; s= '.+so*'; figure(1); clf; hold on;
pTime = now; owPnt2_r = []; ePnt2_r = []; wPnt2_r = [];
for i=1:np, pTime = datestr_(pTime, i, np);
    tw = []; ow = []; er= []; ww = [];
    for j=1:nf
        tw = [tw [F_tot{i}(:,j); i]];
        op2 = oPnt2s_r{i+1}(:,j);
        ow = [ow [op2; i]];
        er = [er norm(F_tot{i}(:,j)-op2)/norm(oPnt2s_r{i}(:,j)-op2)];
        wp2 = wPnt2s_r{i+1}(:,j);
        if wp2==inf
%             plot_(oPnt2s_r{i}(:,j),'s');
%             plot_([oPnt2s_r{i}(:,j) ow(1:2,j)],'b:');
%             plot_([ow(1:2,j) tw(1:2,j)],'.k-');
        else
           ww = [ww [wp2; i]];
        end
    end
    Pnt2s_r{i} = tw;
    err = [err; er];
%     plot_(tw(1:2,:),[c(2) s(2)]);
%     plot_(ow(1:2,:),[c(5) s(5)]);
    owPnt2_r = [owPnt2_r mean(ow(1:2,:),2)];
    ePnt2_r  = [ePnt2_r mean(tw(1:2,:),2)];
    wPnt2_r  = [wPnt2_r mean(ww(1:2,:),2)];
end
figure(3); clf; hold on;
plot_(owPnt2_r,'ks');
plot_(ePnt2_r,'.-');
plot_(wPnt2_r,'ro:');
legend('origianal','Active-mesh','Mean');
axis normal
%%
figure(2); clf; hold on;
for j=1:nf
    k = mod(j,4)+1;
    plot(1:size(err,1),err(:,j)',[':' c(k) s(k)]);
end
```

Hierarchical Structure Generation

```
[ imPnt, desc1 ] = surfpoints( Im_ );
imPts_r{1} = imPnt;
% subplot(2,2,1); imshow(Im); title(['Original']);
bIm=[];
for i=2:n_
   Ims_r{i} = imfilter(Im_, fspecial('gaussian',5*i,5*i) );
   [ imPnt, desc1 ] = surfpoints( Ims_r{i} );
   imPts_r{i} = imPnt;
%    subplot(2,2,i); imshow(bIm{i}); title(['Size=' int2str(5*i) ', Sig=' int2str(5*i)]);
End

for i=1:size(Pts_,2)
   vPts_r{i}.Pt=Pts_{i};
   [ vIdx, vPts ]= idx_voronoi ( Pts_{i} );
   vPts_r{i}.vPts=vPts;
end

if lv_<2 return; end;
vPt = vPts_{lv_}; vPt_1 = vPts_{lv_-1};
for i=1:size(pIdx_,2)
   [ pIdx, pPt ]= idx_inpolygon( vPt_1.Pt, vPt.vPts{pIdx_(i)} );
   if isempty(pPt) continue; end;
   Idx = find(pIdx==1); Nds_r{i}.Idx = Idx; Nds_r{i}.Pt = pPt;
   Nds_r{i}.Nd = nds_vpts ( vPts_, lv_-1, Idx );
end
```

Multilayered Active Tree

```
function [ Ns_r ]= multimesh_mv ( Ns, imPts_, Pth_r )
% ------------------------------------------------------------------------------------------
%%
Ns_r=Ns;
if nargin<1|isempty(Ns), multimesh_mv_demo; return; end
%%

% ------------------------------------------------------------------------------------------
k=2;
for k=2:size(Pth_r,2); disp(['k: ' int2str(k)]); %size(Pth_r,2)
   l3=3; l2=2; l1=1;
   dexFrc3=[];
   for n3=1:size(Ns,1), disp(['n3: ' int2str(n3)]);
      Nd3=Ns_r{n3,k-1};
      if isempty(Nd3)|isempty(Nd3.Pt), continue; end;
      imPtk3 = imPts_{l3}.Pts{k}(:,n3);
      if isinf(imPtk3)
         dexFrc31 = Pt_ref3;
      else
         dexFrc31 = imPtk3-Nd3.Pt; % currentExternal-previous
      end
      dexFrc3 = [dexFrc3 dexFrc31];
      Ns_r{n3,k}.Pt = Nd3.Pt+dexFrc31;
   end;
   Pt_ref3 = mean(dexFrc3,2);
```

```
    for n3=1:size(Ns,1), disp(['n3: ' int2str(n3)]);
      Nd3=Ns_r{n3,k-1};
      if isempty(Nd3)|isempty(Nd3.Pt), continue; end;
      imPtk3 = imPts_{l3}.Pts{k}(:,n3);
      if isinf(imPtk3)
        dexFrc31 = Pt_ref3;
      else
        dexFrc31 = imPtk3-Nd3.Pt; % currentExternal-previous
      end
      Ns_r{n3,k}.Pt = Nd3.Pt+dexFrc31;
      imPtk2 = imPts_{l2}.Pts{k}(:,Nd3.Idx);
      dexFrc2 = imPtk2-Nd3.subPt;
       while sum(sum(isinf(dexFrc2)))>0,
         sn = round(size(dexFrc2(dexFrc2==Inf),1)/2);
         if sn>1
           dexFrc2(dexFrc2==Inf) = Pt_ref2*ones(1,sn);
         else
           dexFrc2(dexFrc2==Inf) = Pt_ref2;
         end
       end
      Pt_ref2 = mean(dexFrc2,2);
      Nd2=Nd3.Nd;
      if isempty(Nd2), continue; end;
      for n2=1:size(Nd3.subPt,2),
        Pt2 = Nd3.subPt(:,n2);
        b2 = .1;
        [ang len] = ang_pt2( Pt2, Pt_ref3 ); b2 = abs(ang)/pi;
%         [ang len] = ang_pt2( Pt2, dexFrc31 ); b2 = abs(ang)/pi;
        a2=1-b2;
        dexFrc23 = a2*dexFrc2(:,n2) + b2*dexFrc31;
%         dexFrc23 = dexFrc2(:,n2);

        Ns_r{n3,k}.subPt(:,n2) = Pt2 + dexFrc23;
      end
%       Ns_r{n3,k}.subPt = Nd3.subPt + (a2*dexFrc2 + b2*dexFrc31*ncol(dexFrc2));
      for n1=1:size(Nd2,2), disp(['n1: ' int2str(n1)]);
        Nd1=Nd2{n1};
        if isempty(Nd1), continue; end;
        imPtk1 = imPts_{l1}.Pts{k}(:,Nd1.Idx);
        dexFrc1 = imPtk1-Nd1.subPt; % currentExternal-previous
        while sum(sum(isinf(dexFrc1)))>0,
          sn = round(size(dexFrc1(dexFrc1==Inf),1)/2);
          if sn>1
            dexFrc1(dexFrc1==Inf) = Pt_ref1*ones(1,sn);
          else
            dexFrc1(dexFrc1==Inf) = Pt_ref1;
          end
        end
        Pt_ref1 = mean(dexFrc1,2);
        a1=.9; b1=.075; c1=.015;
        for n11=1:size(Nd1.subPt,2),
          Pt1 = Nd1.subPt(:,n11);
          [ang len] = ang_pt2( Pt_ref2, Pt_ref3 ); c1 = (abs(ang)/pi)*1/4;
          [ang len] = ang_pt2( Pt1    , Pt_ref3 ); b1 = (abs(ang)/pi)*3/4;
          b23 = b1+c1;
          a1  = 1-b23;
```

```
            dexFrc23 = a1*dexFrc1(:,n11) + b1*dexFrc2(:,n1) + c1*dexFrc31;
%               dexFrc23 = dexFrc1(:,n11);

            Ns_r{n3,k}.Nd{n1}.subPt(:,n11) = Pt1 + dexFrc23;
        end
%           Ns_r{n3,k}.Nd{n1}.subPt = Nd1.subPt + ( a1*dexFrc1 + b1*dexFrc2(:,n1)*ncol(dexFrc1) +
c1*dexFrc31*ncol(dexFrc1));
      end
   end
end
```

Sequential Image Mosaicing

```
function [ Img_r ] = imgcut( Img_, offPnt_, c_, thk_ )

%plot 3D points on 2D image with viewpoint.
%>> Img_r = imcreate_( CData_, bAlpha_ )
%
%(c) copyright SangHoon Han, ARMg UFL 2/8/2007

%% ------------------------------------------------------------------------------------
if nargin<1,       return; end
h = Img_.h; w = Img_.w; c = Img_.c;
cp =round([w h]'/2);
if nargin<2, offPnt_ =      cp; end
if nargin<3,      c_ = [1 1 0]; end
if nargin<4,    thk_ =       0; end
%% ------------------------------------------------------------------------------------
xy = [1 w; 1 h];
Pnts = [xy(:,1) [xy(1,1) xy(2,2)]' xy(:,2) [xy(1,2) xy(2,1)]' xy(:,1)];
Img_r = Img_;
if thk_>0, Img_r = imglines(Img_r, Pnts, c_); end
Img_.gIm = zeros(h, w);

%% get angle
%
dp = cp-offPnt_;
Ang = -pi/2+atan2(dp(2),dp(1)); %Ang = -90*pi/180;

%% get sp, ep
% Ang=10*pi/180;
a = tan(Ang);
x = offPnt_(1); %y = Img_r.h-np(2);
y = offPnt_(2);

sy = 1;
if a==0
   sx = x;
   sy = y;
else
   sx = x+(sy-y)/a;
   if sx<1,
      sx = 1;
      sy = y+(sx-x)*a;
   elseif sx>w
      sx = w;
      sy = y+(sx-x)*a;
```

```
    end
end

sp = round([sx sy]');

ey = h;
if a==0
    ex = x;
    ey = y;
else
    ex = x+(ey-y)/a;
    if ex<1,
        ex = 1;
        ey = y+(ex-x)*a;
    elseif ex>w
        ex = w;
        ey = y+(ex-x)*a;
    end
end

ep = round([ex ey]');
Img_r = imgline(Img_r, [sp, ep], c_, thk_);

if Ang<0
    sPnt = ep; ePnt = sp;
else
    sPnt = sp; ePnt = ep;
end

% [sPnt, ePnt]
%% pntlist
% sPnt=[1,100]'; ePnt=[320,50]';
sx = sPnt(1); ex = ePnt(1);
sy = sPnt(2); ey = ePnt(2);
eg = [];
switch sx
    case 1, eg = 1;
        if (1<sy & sy<h), Pnts(:,2) = sPnt; eg = 1; end;
    case w, eg = 3;
        if (1<sy & sy<h), Pnts(:,4) = sPnt; eg = 3; end;
    otherwise
        switch sy
            case 1, eg = 4; Pnts(:,1) = sPnt; Pnts(:,5) = sPnt;
            case h, eg = 2; Pnts(:,3) = sPnt;
        end
end

switch ex
    case 1
        if (1<ey & ey<h),
            switch eg
                case 2
                    Pnts(:,1) = ePnt; Pnts(:,5) = ePnt; Pnts(:,4) = [];
                case 3
                    Pnts(:,1) = ePnt; Pnts(:,5) = ePnt;
                case 4
```

```
            Pnts = [ePnt Pnts];
        end
    end
  case w
    if (1<ey & ey<h),
        switch eg
          case 4
            Pnts(:,3) = ePnt; Pnts(:,2) = [];
          case 1
            Pnts(:,3) = ePnt;
          case 2
            Pnts = [Pnts(:,1:3) ePnt Pnts(:,4:5)];
        end
    end
  otherwise
    switch ey
        case 1          ,
          switch eg
            case 1
              Pnts(:,4) = ePnt; Pnts(:,3) = [];
            case 2
              Pnts(:,4) = ePnt;
            case 3
              Pnts = [Pnts(:,1:4) ePnt Pnts(:,5)];
          end
        case h,
          switch eg
            case 3
              Pnts(:,2) = ePnt; Pnts(:,5) = ePnt; Pnts(:,1) = [];
            case 4
              Pnts(:,2) = ePnt;
            case 1
              Pnts = [Pnts(:,1:2) ePnt Pnts(:,3:5)];
          end
    end
end
Img = imglines(Img_, Pnts, c_);
%% imfill %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fIm  = imfill(Img.gIm);
gIm = 1-fIm./max(max(fIm));
for i=1:Img_r.c
  Img_r.Im(:,:,i) = Img_r.Im(:,:,i).*gIm;
end
Img_r.gIm = Img_r.gIm.*gIm;


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function imgcut_demo() % clear
%%
Img_ = imgread('test1.png'); c_= [1 1 0];
offPnt_=[150 100]';
%% ===========================================================================
clc; Img_r = imgcut( Img_, offPnt_, c_ );
Img_r = imgpoint(Img_r,[160 120]',[1 1 0],'+');
Img_r = imgpoint(Img_r,offPnt_,[1 1 0],'s');
clf; imgshow(Img_r);
```

3D Reconstruction with Images

```
function [ Ln_r ]= ln_pttr ( Pt_, Extr_,Intr_ )
Pth = cpth_extr(Extr_);
Pnt1 = Pth(1:3);
[ cfrm3_r, Pnt2 ] = cfrm3tr( Extr_,Intr_,[], Pt_ );
if isempty(Pnt2), return; end;
Ln_r = ln_pnt2(Pnt1, Pnt2);

% ================================================================================
function [ Pnt_r1, Pnt_r2, len_r, Ln_r, Ang_r ]= pnt2_ln2 ( Ln_1, Ln_2 )
 [S1, S0L1] = s_ln(Ln_1); [S2, S0L2] = s_ln(Ln_2); % S1,S2 is unit vector.

dtS12 = dot(S1,S2);  crS12 = cross(S1,S2);
Ang_r = acos(dtS12); sinAng = sin(Ang_r);
if sinAng==0, sinAng=1; end;
S12 = crS12/norm(crS12);
dtS2L1 = dot(S2,S0L1); dtS1L2 = dot(S1,S0L2);
len_r  = -(dtS1L2+dtS2L1) /sinAng; % length
if (dtS2L1==0), % parallel or coplanar
  Pnt1 = dot(S0L2,S12)/sinAng;
  Pnt2 = dot(S0L1,S12)/sinAng;
  Pnt_r1 = Pnt1*S1+Pnt2*S2;
else %
  SL2_cS12S2 = S0L2-len_r*cross(S12,S2);
  Pnt_r1 = cross(S0L1,SL2_cS12S2)/dtS2L1;
end
Pnt_r2 = Pnt_r1+len_r*S12;
S0L12 = cross(Pnt_r1, S12);
Ln_r = ln_s(S12, S0L12);

% ================================================================================
function [ Pnt_r ]= pnt_pttr2 ( Pt_1,Pt_2, Extr_1,Extr_2, Intr_ )
Ln1 = ln_pttr( Pt_1, Extr_1, Intr_ ); if isempty(Ln1), return; end;
Ln2 = ln_pttr( Pt_2, Extr_2, Intr_ ); if isempty(Ln2), return; end;
[ Pnt1, Pnt2 ] = pnt2_ln2( Ln1, Ln2 );
Pnt_r = mean([Pnt1 Pnt2],2);
```

3D Reconstruction with Range Data

```
function [ rVrt ] = reconstructObject(scn_, Path_, Dst_, viewp_);
%scanObject Summary of this function goes here
%  Detailed explanation goes here
%02.scan object
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
if nargin<1,
    scn_.rng = 90; scn_.smpNum = 12; scn_.t = 1; scn_.scnNum = 41; scn_.len =10;
end
if nargin<2, paths_ = load('paths.txt'); end
if nargin<3, tri_ = load('tri_.mat'); end
if nargin<4, viewp_ = [.5 -1 .5]; end

%% create the resolution of beam with respect to time %%%
sN = scn_.smpNum; sT = scn_.t;
```

```
sR = (scn_.rng) * pi/180;
if sN == 1,
   sS = pi/2; sE = pi/2; sD = sR; dt = sT;
   s = pi/2;
else
   sS = (pi - sR)/2; sE = sS + sR;
   sD = sR/(sN-1);
   s = sS:sD:sE; %[rad]
end
% dt = sT/sN;
% tm = 0:dt:(sT*scn_.scnNum - dt);

Tr = Path_(:,2:4)'; Rt = Path_(:,5:7)';

%% measure distances %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Dst = [];
Dst(scn_.scnNum, sN).Pnt=[0 0 0]';
i=1;
Dst(scn_.scnNum, sN).dst = scn_.len;
rDst = []; t = 1;
rFrm = []; rVrt=[];
%%
% cPnt = .25*[ .7071 .7071 0; 0 1 0; -.7071 .7071 0]';
for i = 1:(scn_.scnNum)-1, %disp(i);%[i scn_.scnNum]%i 5[s]
   sPnt = Path_(i,2:4)';% +org_;
   prevPnt = []; fsPnt = []; fePnt = [];
%%
   for j = 1:sN, %[i j t]
      fsPnt = [fsPnt sPnt];
      Dst_ij = Dst_(i,j);
      if Dst_ij~=0,
         Pnt = Dst_ij*[cos(s(j)) sin(s(j)) 0]';
         trPnt = trans_yxz(Pnt, sPnt, Path_(i,5:7)');
         ePnt = trPnt(1:3);
         rVrt = [rVrt; ePnt'];
      end
   end
end
```

APPENDIX C
DEVELOPMENT TOOLS

Table C-1. Softwares.

Development Tool	Application
MATLAB 2009	Feature detection, Feature tracking, Simulation
OpenCV 2.0	Capturing, Image processing
Visual C++ 2005	Robot controller, GPS
C++Builder 2006	Real-time 3D reconstruction, Robot controller user interface
Delphi 2006	Virtual grove
GLScene	Virtual grove
ARToolkit	Augmented Reality

LIST OF REFERENCES

Bay, H., A. Ess, T. Tuytelaars, and L. V. Gool. 2008. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding* 110(3): 346–359.

Crane III, C. D., and J. Duffy. 2008. *Kinematic Analysis of Robot Manipulators*. New York, N.Y.: Cambridge University Press.

Crane III, C. D. 2010. *Screw Theory for Spatial Robot Manipulators*. New York, N.Y.: Cambridge University Press.

Computer Vision Research Group. 2008. Camera Calibration Toolbox for MATLAB. Pasadena, California: California Institute of Technology. Available at: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html. Accessed 19 February 2010.

Davison, A. J., and D. W. Murray. 2002. Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(7): 865–880.

Dornaika, F., and R. Chung. 2004. Mosaicking images with parallax. *Signal Processing: Image Communication* 19(8): 771–786.

Griffin, A., and J. Kittler. 2002. An active mesh based tracker for improved feature correspondences. *Pattern Recognition Letters* 23(4): 443–449.

Hasler, D., and S. Süsstrunk. 2004. Mapping colour in image stitching applications. *Journal of Visual Communication and Image Representation* 15(1): 65–90.

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

Isard, M., and A. Blake. 1998. CONDENSATION: Conditional density propagation for visual tracking. *International Journal of Computer Vision* 29(1): 5-28.

Kanazawa, Y., and K. Kanatani. 2004. Image mosaicing by stratified matching. *Image and Vision Computing* 22: 93–103.

Ke Y., and R. Sukthankar. 2004. PCA-SIFT: A more distinctive representation for local image descriptors. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2: 506–513.

Kise M., and Q. Zhang. 2006. Reconstruction of a virtual 3D field scene from ground-based multi-spectral stereo imagery. ASABE Meeting Presentation Paper No.063098. Portland, Oregon: ASABE.

Klien, G., and D. W. Murray. 2007. Parallel tracking and mapping for small AR workspaces. *In Proc. of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*: 1–10.

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

Ma, Y., S. Soatto, J. Kosecka, and S. S. Sastry. 2004. *An Invitation to 3-D Vision: From Images to Geometric Models*. New York, New York: Springer.

Mehta, S. S. 2007. Vision-based control for autonomous robotic citrus Harvesting. MS thesis. University of Florida, Department of Agricultural and Biological Engineering.

Meier, E. B., and F. Ade. 1999. Using the condensation algorithm to implement tracking for mobile robots. *In Proc. of the Third European Workshop on Advanced Mobile Robots, eurobot'99*: 73–80.

Molloy, D. 2000. Active meshes for motion tracking. PhD diss, Dublin, Ireland: Dublin City University, Department of Engineering and Design.

OpenCV. 2001. *Open Source Computer Vision Library Reference Manual*. Ver1.0. Santa Clara, California: Intel Corporation.

Rav-Acha, A., Y. Shor, and S. Peleg. 2004. Mosaicing with parallax using time warping. *In Proc. Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop* 11: 164–172.

Shlyakhter, I., M. Rozenoer, J. Dorsey, and S. Teller. 2001. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications* 21(3): 53–61.

Subramanian, V. 2005. Autonomous vehicle guidance using machine vision and laser radar for agricultural applications. MS thesis. Gainesville, Florida: University of Florida, Department of Agricultural and Biological Engineering.

Subramanian, V. 2008. Autonomous vehicle guidance for citrus grove navigation. PhD Diss. University of Florida, Department of Agricultural and Biological Engineering.

Szeliski, R. 2004. Image alignment and stitching: A tutorial. MSR-TR-2004-92. Redmond, Washington: Microsoft Research.

Trucco, E., and A. Verri. 1998. *Introductory Techniques for 3-D Computer Vision*. New York, New York: Prentice Hall.

Tumbo, S. D., M. Salyani, J. Whitney, T. Wheaton, and W. Miller. 2001. Laser, ultrasonic and manual measurements of citrus tree canopy volume. ASAE Meeting Presentation Paper No.011068. Sacramento, California: ASAE.

Wei, J., and M. Salyani. 2004. Development of a laser scanner for measuring tree canopy characteristics. ASAE Meeting Presentation Paper No.041168. Ottawa,Canada: ASAE.

Yoon, K., G. Jang, S. Kim, I. Kweon. 2001. Color landmark based self-localization for indoor mobile robots. *Journal of Control, Automation, and Systems Engineering* 7(9): 749–757.

Younse, P. 2005. Intersection detection and navigation for an autonomous greenhouse sprayer using machine vision. MS thesis. University of Florida, Department of Agricultural and Biological Engineering.

Zaman, Q. U., A. W. Schumann, and H. K. Hostler. 2005. Quantifying sources of error in ultrasonic measurements of citrus orchards. ASAE Meeting Presentation Paper No.051123. Tampa, Florida: ASAE.

Zhu, Z., E. M. Riseman, and A. R. Hanson. 2001. Parallel-perspective stereo mosaics. The Eighth IEEE International Conference on Computer Vision. Vancouver, Canada: ICCV

BIOGRAPHICAL SKETCH

SangHoon Han was born in Pusan, South Korea, 1972. He received his Bachelor of Science degree and Master of Science degree in mechanical engineering design at College of Engineering at Korea Aviation University, Korea, in February 1996 and February 1999, respectively. He attended the University of Florida in Jan, 2004 for the Doctor of Philosophy degree in agricultural and biological engineering. During his PhD program, he worked as a graduate research assistant with Dr. Thomas Burks.