Comparison of Pose Estimation Algorithms for a Deleafing Greenhouse Robot

Vishanth Easwaramurthy

Supervised by Prof. dr. Robert Babuska

Image Credits - Priva BV

TU Delft

Comparison of Pose Estimation Algorithms for a Deleafing Greenhouse Robot

by

Vishanth Easwaramurthy

For the degree of Master of Science in Mechanical Engineering (Vehicle Engineering) at the Delft University of Technology

May 31, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3ME) · Delft University of Technology

This thesis was done under the supervision of

Prof. dr. Robert Babuska, CoR, Delft University of Technology

The thesis committee consisted of

Prof. Dr. Robert Babuska,	CoR, Delft University of Technology
Dr. J.F.P.Kooij,	CoR, Delft University of Technology

ABSTRACT

For autonomous navigation of mobile robots in an unknown environment, mapping the robot's environment, and localizing its relative position in the environment is of utmost importance. However, in a known and controlled environment, being able to localize its position at a given time instant is sufficient for the robot to navigate autonomously. This thesis focuses on the pose estimation of a greenhouse robot using images from an on-board camera and only for its motion on the guided rails. Therefore, to solve the problem of pose estimation of a mobile greenhouse robot, Visual Odometry (VO) was the best-suited approach. A review of the literature revealed that most of the current state-of-the art VO techniques were developed for autonomous navigation in an urban environment or an indoor space. Even though there were publicly available benchmark datasets for VO evaluation such as KITTI and TUM datasets, there was no VO dataset available for a greenhouse environment. A key challenge in collecting visual datasets in a greenhouse was obtaining the ground truth poses associated with all the images. Therefore, an experimental setup was developed to collect VO datasets along with ground truth using store-bought table plants, tomatoes, a mobile camera, and a stereo camera. The experimental design was focused to emulate the greenhouse environment as much as possible. The camera's motion during the experiment was also chosen to emulate the robot's motion on the guided rails. Subsequently, multiple pose estimation algorithms were run on the collected datasets for a comparison study. The experiments provided insights into the algorithms' performances, and how it gets affected when there are changes to the image resolution, the overlapping regions between two consecutive images, and the camera to plants distance. The results show that, for the designed experimental setup, the pose estimation of SVO Pro using the images obtained from a monocular camera works best.

Contents

1

INTRODUCTION

For autonomous mobile robots to navigate in an unknown environment, the autonomous agent should be able to build a map of the environment, and estimate its current position in the map. Subsequently, using the estimated position and map of the environment, the required path planning can be done for the robot to reach the target location from its current location. However, for applications where the robot's working environment is known, pose estimation alone is sufficient to solve the path planning problem. Structure from Motion (SfM), Visual Odometry (VO), and Visual Simultaneous Localisation and Mapping (V-SLAM) are three different, but closely interlinked concepts that estimate poses using images obtained from an on-board camera. The differences lie in their approaches, applications, and the problems they solve. In all three concepts, key points are detected in the input images, tracked or matched across multiple frames, and subsequently, the relative poses are obtained. While V-SLAM and SfM deal with simultaneous scene reconstruction and camera poses estimation, VO only estimates the camera poses.

SfM estimates the three-dimensional structure of a scene using multi-view images captured by multiple cameras or a single camera at multiple positions [1]. The relative camera poses are obtained by detecting and tracking feature points across all the images, and the 3D locations of these points are obtained by triangulating the estimated camera poses and feature points. Usually, in SfM, all the estimates are in a relative scale and additional information, such as the size of a scene object, is required to obtain the absolute scale. SfM is an offline implementation, whose computational time grows with the number of images obtained and these images can be ordered or unordered.

SLAM solves the localization problem faced by an autonomous mobile robot when navigating through an unknown environment. With each new incoming sensor measurement, SLAM incrementally builds a consistent map of the environment while simultaneously determining the robot's location within this map. In SLAM, both the trajectory of the robot and the location of all landmarks are estimated online without the need for any prior knowledge [2]. While SLAM starts pose estimation and map building relative to its initial estimates, it detects loop closures by recognizing a previously visited location,

thereby obtaining a global trajectory and also improving its estimates. When SLAM is implemented only using cameras it is called Visual-SLAM(V-SLAM).

VO solves the localization problem faced by an autonomous mobile robot when navigating through a known environment. In VO, only the poses are estimated using images from the onboard camera(s) [3]. The term 'Visual Odometry' was first defined by Nister et al in [4], analogous to the wheel odometry which incrementally estimates motion by integrating the number of turns of wheels. For VO to function properly there should be a static scene with enough texture to extract the apparent motion and also sufficient scene overlap between consecutive frames. Contrary to SfM, V-SLAM, and VO are sequential and online techniques that output new estimates and update previous estimates with each new incoming image or image pair. This thesis will focus on solving the localization problem of a deleafing green house robot.

1.1. KOMPANO DELEAFING ROBOT

The Kompano Deleafing Robot is used to cut the leaves of a tomato plant autonomously. The robot is deployed in a greenhouse environment where it travels on guided rails, with tomato plants on either side. For the deleafing task, the robot is equipped with a telescopic arm with a stereo camera attached to it. During the robot's motion from the beginning to the end of a guided rail, it sequentially removes leaves from the plants on one side of the rail. At the end of the rail, the robot's arm switches sides, and the deleafing is done to plants on the other side of the rail on its way back. When the robot reaches the beginning of the guided rail, it rides onto a concrete platform, moves and positions itself to match the orientation of the next guided rail, and continues its guided motion along the next rail. In addition to the guided motion on the rails and the motion on the concrete platform, the robot will also be subjected to travel for other purposes such as servicing and charging.

The robot's driven rear wheels have servo motors and at the front, there are omniwheels for maneuvering on the concrete platform. When the robot drives off the rails, the rear wheels are the first to get on the concrete platform. The robot then rotates and translates accordingly to get on the next guided rail. For the robot navigation, currently, the following sensors are used:

- wheel encoders for odometry,
- Time of Flight (ToF) sensor at the front to detect the guided rail before moving on it,
- a proximity sensor at the rear to detect if the robot is on the guided rail.

The navigation of the deleafing robot will be affected by the irregularities in the greenhouse infrastructure. Given that the distance between two pillars that fit 5 guided rails is precisely 8m, the average distance between two guided rails should be 1.6m. However, that is not always the case (Figure 1.1a). Each guided rail's length also varies between 70m and 120m. The guided rails and the concrete platform are not always perpendicular to each other. Due to this shift in the orientation, the robot should change its position accordingly on the concrete platform before moving onto the guided rails (Figure 1.1b).

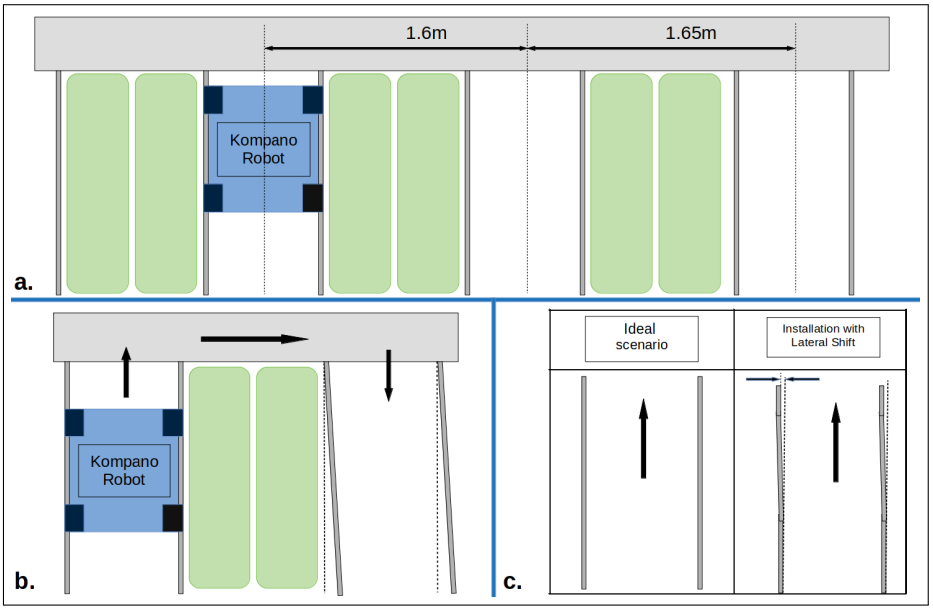

Figure 1.1: Irregularities in the guided rails: (a) inconsistency in the distance between adjacent rails, (b) irregularities in the rail's orientation with respect to the concrete platform, (c) lateral Shift in the rails.

The rails might also be affected by the lateral shift. Locally, for a distance of 2 m on the rails, the lateral shift can be 0.1 m, and globally, for the total length of 110 m of the rails, the lateral shift is not more than 0.2m. (Figure 1.1c). There might also be cracks, uneven spots, and other irregularities on the concrete platform. Therefore, for autonomous navigation on the concrete platform and guided rails, precise pose estimation is required. Furthermore, reliable pose estimates on the guided rails will help in accurately locating crops with distinct features (disease-stricken crops), and the locations of vine tomatoes can help in predicting the yield estimates.

1.2. RESEARCH QUESTION

Since the working environment for a deleafing robot is known, only pose estimation is required for solving the navigation problem. This thesis will focus on the performance comparison of different pose estimation techniques only for the motion of the robot on the guided rails. The main research questions pertaining to this goal are:

1. **Most of the existing publicly available VO algorithms have been developed for an urban environment [5], [4] or an indoor space [6], [7]. How will these algorithms perform on datasets that emulate the greenhouse environment?**

2. **Which camera (monocular or stereo) is best suited for the deleafing robot?**

3. **How does the camera's positioning relative to the plants affect the performance of the algorithms?**

4. **What is the best achievable accuracy of VO on a datasets that emulate the greenhouse environment?**

1.3. THESIS OUTLINE

Chapter 2 will proved an overview of VO by explaining the core task of the all VO algorithms, and a generic pipeline of how that task is performed. Subsequently, the list of pose estimation algorithms used for the comparison study is also explained in this chapter. Chapter 3 explains the experimental setup and different datasets collected. The results obtained from running the pose estimation algorithms on the custom datasets is explained in Chapter 4. Finally the conclusions drawn from the results and recommendations for future work is discussed in Chapter 5.

2

VISUAL ODOMETRY

2.1. PROBLEM FORMULATION

Consider a monocular camera mounted on a mobile robot capturing images at discrete time steps. At each time instant k, the cameras capture an image (I_k). Therefore, for n time steps, there will be n images, $I_{l,0:n} = [I_0, I_1, ..., I_n]$. Figure 2.1 shows the camera poses corresponding to the first three time instances ($k = 0, 1$, and 2).

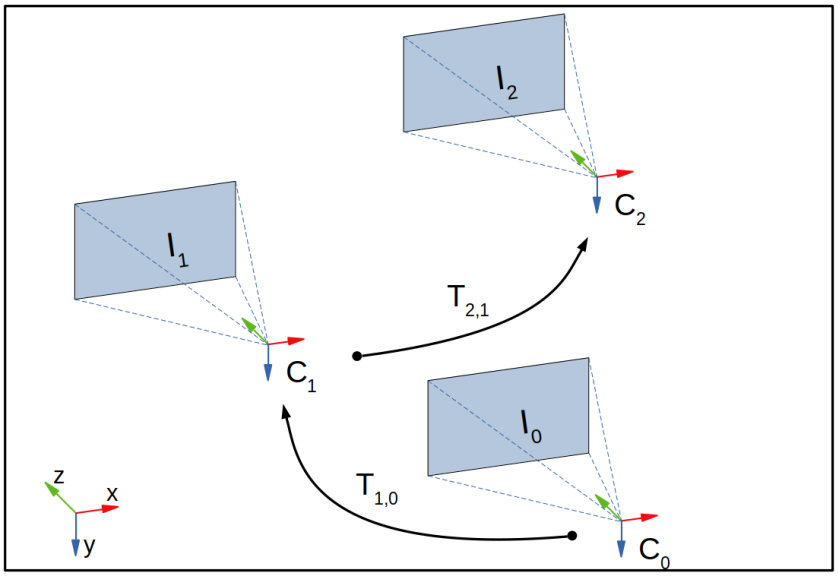

Figure 2.1: The camera's poses at time instances $k = 0, 1$ and 2

Egomotion estimation involves estimating the camera pose, C_k, at each time instant k. The camera pose is defined as

$$C_k = \begin{bmatrix} x_k \\ y_k \\ z_k \\ \alpha_k \\ \beta_k \\ \gamma_k \end{bmatrix}, \tag{2.1}$$

where, at each time instant k, the coordinates (x_k, y_k, z_k) specify the camera's position and $(\alpha_k,\beta_k,\gamma_k)$ are the rotations about the x-axis, y-axis, and z-axis respectively. Since poses in VO are estimated relative to the initial pose, the origin of the world coordinate frame coincides with the origin of the camera coordinate system at time instant $k = 0$.

The change in a camera's pose, from C_{k-1} to C_K, can be caused either by translation or rotation of the camera. The translation can be expressed by separate motions along the x, y and z directions as $x_k = x_{k-1} + a$, $y_k = y_{k-1} + b$, and $z_k = z_{k-1} + c$. Using homogeneous coordinates, this translation can be represented as

$$\begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \\ 1 \end{bmatrix} \tag{2.2}$$

Given that α, β, and γ are the rotations about the x-axis, y-axis, and z-axis respectively, their corresponding rotation matrices are

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}; R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}; R_z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the $x-y-z$ rotation is used, the rotation matrix representing the camera's orientation at time step k, relative to its orientation at $k-1$, is given by

$$R_{k,k-1} = R_x R_y R_z$$

Therefore, the change in the camera's orientation can be represented as

$$\begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} = \begin{bmatrix} R_x R_y R_z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \\ 1 \end{bmatrix} \tag{2.3}$$

Combining equations (2.2) and (2.3) results in

$$\begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \\ 1 \end{bmatrix} \tag{2.4}$$

where

$$R_{k,k-1} = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \cos\alpha\sin\gamma+\cos\gamma\sin\alpha\sin\beta & \cos\alpha\cos\gamma-\sin\alpha\sin\beta\sin\gamma & -\cos\beta\sin\alpha \\ \sin\alpha\sin\gamma-\cos\alpha\cos\gamma\sin\beta & \cos\gamma\sin\alpha+\cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta \end{bmatrix}$$

is the orthogonal rotation matrix, and

$$t_{k,k-1} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

is the translation vector. The matrix is in equation (2.4) is defined as the transformation matrix

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}. \tag{2.5}$$

$T_{k,k-1}$ relates the camera poses C_k and C_{k-1} as

$$\begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} = T_{k,k-1} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \\ 1 \end{bmatrix}. \tag{2.6}$$

The core task of all VO algorithms is to estimate this transformation matrix, $T_{k,k-1}$, using the images obtained at time steps $k-1$ and k.

2.2. PIPELINE OVERVIEW

The main components of a VO pipeline are shown in Figure 2.2. For each image or image pair, key features are detected and matched with those obtained from the previous time steps. Let f_{k-1} be the feature set at time step $k-1$, and f_k be the feature set at time step k. Motion estimation is the core task of all VO algorithms. Using the matched features between the current frame k, and previous frame $k-1$, the camera motion between the two frames is computed. By concatenating all the frame-to-frame motion, the entire trajectory of the camera can be obtained [3].

Depending on how the feature correspondences are specified (2D or 3D), different motion estimation techniques are used.

- 2D-to-2D: Both the set of features f_k and f_{k-1} are specified in 2D image coordinates.

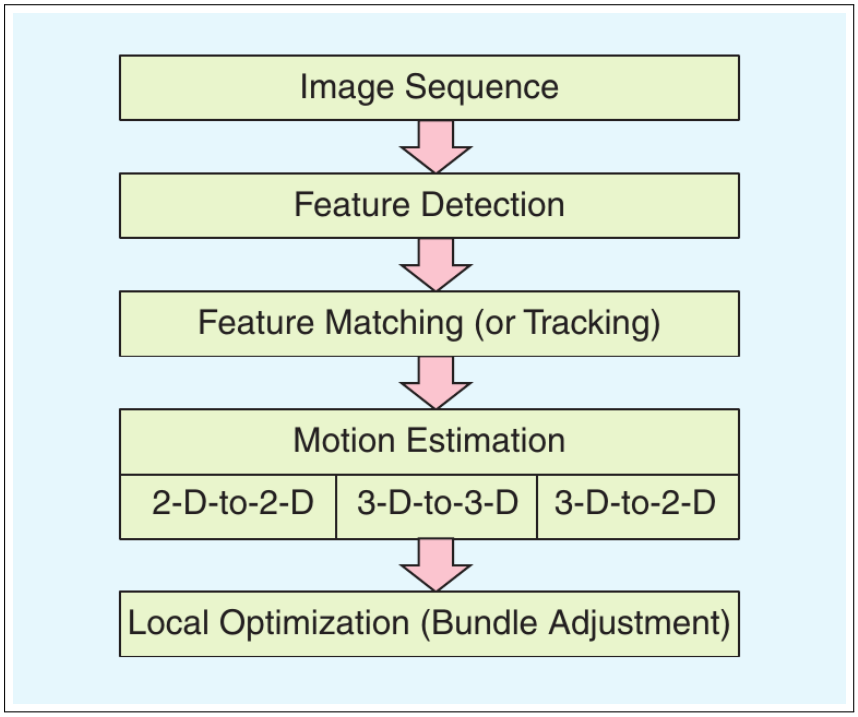

Figure 2.2: VO Pipeline - Geometric Approach [3].

- 3D-to-3D: Both the set of features f_k and f_{k-1} are specified in 3D by triangulating the points using a stereo camera at each time step.
- 3D-to-2D: In this case, the features f_{k-1} are specified in 3D, and f_k are specified in 2D image coordinates.

In motion estimation techniques that rely on detecting and matching features across images, usually, there are outliers in the feature matches. Possible causes of outliers are image noise, occlusions, blur, viewpoint changes, and illumination changes. These outliers affect the frame-to-frame pose estimates, causing a significant drift in the estimated trajectory from the ground truth trajectory. One of the standard methods used for outlier removal is Random Sample Consensus (RANSAC) [8]. In RANSAC, a model hypothesis is generated using randomly sampled data points and the generated hypothesis is verified on other data points. The model with the highest consensus from the other data points is selected as the solution. For a VO problem, the model is the estimated $T_{k,k-1}$ matrix between two consecutive camera views and the data points are the feature matches.

Instead of using only a certain number of features in each image, dense motion estimation techniques work directly on the images using the pixel intensity values [9], and [10]. While the direct motion estimation techniques eliminate feature detection and matching steps, they are relatively more computationally intensive. Semi-dense approaches are a trade-off between the feature-based and direct approaches. Semi-dense

approaches eliminate the entire process of detecting and describing features without compromising on the computational power requirement [7], and [11].

For a total number of n time steps, the pose C_k at each time step k is obtained by applying equation (2.4) as follows

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = T_{1,0} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \text{ and } C_1 = \begin{bmatrix} x_1 & y_1 & z_1 & \alpha_1 & \beta_1 & \gamma_1 \end{bmatrix}^\top$$

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = T_{2,1} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = T_{2,1} \left(T_{1,0} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \right) \text{ and } C_2 = \begin{bmatrix} x_2 & y_2 & z_2 & \alpha_2 & \beta_2 & \gamma_2 \end{bmatrix}^\top$$

$$\vdots$$

$$\begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix} = T_{n,n-1} \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ z_{n-1} \\ 1 \end{bmatrix} \text{ and } C_n = \begin{bmatrix} x_n & y_n & z_n & \alpha_n & \beta_n & \gamma_n \end{bmatrix}^\top$$

where the angles α_k, β_k, and γ_k are obtained from the $R_{k,k-1}$ component in $T_{k,k-1}$. Therefore, for a set of n image pairs, there will be $n-1$ transformation matrices. The above equations show that VO is a sequential process and the pose C_k depends on all the transformation matrices $\{T_{1,0}, T_{2,1}, ..., T_{k,k-1}\}$. Therefore, the error associated with the estimation of each $T_{k,k-1}$ matrix gets accumulated over time, leading to a significant drift in the final pose estimate from the ground truth. To remove the noise and obtain a more accurate estimate, local optimization is required. One of the most popular optimization techniques used for VO is windowed bundle adjustment. Bundle adjustment simultaneously optimizes both the 3D structure and the estimated camera parameters using a cost function. In windowed bundle adjustment, this optimization is done only over a fixed number of frames.

2.3. ALGORITHMS

To perform the comparative study of different pose estimation techniques, new datasets were collected to emulate a greenhouse environment, and the following algorithms were run on the collected datasets. Chapter 3 discusses in detail the experimental setup and data collection process.

2.3.1. IMAGE REGISTRATION USING CORRELATION & FEATURE DISPLACEMENT

Image registration is a process that overlays two or more images from the same scene to geometrically align them for analysis. The overlaying is done by transforming all the

images into one coordinate system, thereby establishing correspondences between the images. The main task of image registration is estimating the transformation function that relates the images. One of the most commonly used methods for estimating the transformation is correlation [12].

The transformation between two consecutive images in a dataset was estimated using MATLAB's built-in function **'imregcorr'**. The function returns a geometric transformation object that maps the pixels in the current image to pixels in the previous (reference) image. As will be explained in Chapter 3, in the datasets collected for this project, the transformation of the images is predominantly caused by translation. Therefore, the output of the **'imregcorr'** function will be the estimated translation in image pixels. To obtain the translation in terms of unit length, a scale value representing the number of pixels per unit length should be known. The required scale value is obtained by computing the feature displacements.

For all the images in the dataset, SURF features are detected and matched across two successive frames. Subsequently, the feature displacements between the two frames are calculated in image pixels. To remove the incorrect feature matches, matches with horizontal and vertical displacements higher than user-defined threshold values are filtered out. For the remaining matches, the median feature displacement value is calculated in image pixels. Using the median feature displacement and corresponding ground-truth camera displacement, the scale value representing the number of pixels per unit length can be approximately calculated. Using this scale value, the translation in terms of unit length can be calculated.

This method for computing the scale value and estimating the transformation using correlation is reliable only when the images contain objects with depth values that do not vary drastically, and the frame to frame camera displacement is known. The datasets collected for this project satisfy these criteria (Chapter 3).

2.3.2. LIBRARY FOR VISUAL ODOMETRY 2 (LIBVISO2)

The LIBVISO2 is a sparse feature-based VO algorithm for monocular and stereo cameras [6], built on the authors' earlier work LIBVISO1 [13]. In comparison with LIBVISO1, LIBVISO2 has higher density of feature matches, a speed-up in feature matching and and pose estimation, and supports monocular camera. In this method, the input images are filtered with 5×5 blob and corner masks. Subsequently non-maximum and non-minimum suppressions are employed on the filtered images to obtain features belonging to one of the four classes - blob max, blob min, corner max, and corner min. The features are only matched within their respective classes. Given two feature points, the algorithm compares 11×11 block windows of horizontal and vertical Sobel filter responses to each other using the sum of absolute differences error metric. Sobel filter is a discrete differentiation operator used to compute an approximation of the gradient of an image intensity function.

The inputs to the stereo version are feature matches between the right and left images and two consecutive frames. For each feature candidate in the current left image, using an MxM search window, the best matches are found in the previous left image, next in the previous right image, then the current right image, and finally in the current left image again. A feature match, called 'circle match', is accepted if the last feature

coincides with the first feature. The stereo algorithm uses the 3-point RANSAC estimation and the monocular algorithm uses 8-point RANSAC algorithm. In either case, the reprojection error of the feature matches is minimized to compute the camera pose.

The algorithm is available as a C++ library with MATLAB wrappers for computing the 6 DOF motion of a moving stereo or monocular camera. The LIBVISO2 algorithm was implemented on the Karlsruhe Dataset. The dataset contains stereo sequences of an urban environment recorded from a moving vehicle in Karlsruhe.
Project URL: `http://www.cvlibs.net/software/libviso/`

2.3.3. ORB-SLAM2

ORB-SLAM2 is built on the authors' earlier work, ORB-SLAM [14], which is a simultaneous localization and mapping algorithm that uses Oriented multiscale FAST and Rotated Brief (ORB) corner feature detection and matching [15]. FAST corners are detected at eight scale levels and with a scale factor of 1.2. Each scale level is divided into cells, and corners are detected in each cell to ensure a minimum of 5 corners per cell. Subsequently, the orientation and ORB descriptors are computed on the detected corner points. With every new image, the algorithm matches 2-D points in the current frame to the 3-D points obtained by triangulating matched points from the previous frames, and then performs motion-only Bundle Adjustment(BA) to optimize the camera pose. Motion only BA optimizes the camera orientation and position, minimizing the reprojection error between matched 3-D points in the world coordinates and 2-D points in image coordinates.

While ORB-SLAM is a monocular SLAM algorithm, ORB-SLAM2 supports stereo and RGB-S cameras as well. A key difference between the monocular and stereo versions is the need for map initialization in the monocular version. Map initialization is done to compute the relative pose between two frames to triangulate an initial set of map points [5]. In the monocular version, two geometric models are computed in parallel: a homography for a planar scene and a fundamental matrix assuming a non-planar scene. A heuristic is used to select a model and recover the relative pose for the selected model. When stereo cameras are used, the depth information can be obtained from just one frame, and an initial map is created from the stereo keypoints.

The implementation is available in C++. The ORB SLAM algorithm was implemented on the KITTI benchmark dataset [16], indoor sequences of TUM RGB-D dataset [17], the robot sequences of NewCollege [18] and the EuRoC dataset [19] collected from a micro aerial vehicle in an industrial environment.
Project URL: `https://webdiis.unizar.es/~raulmur/orbslam/`

2.3.4. FAST SEMI-DIRECT VO (SVO PRO)

SVO Pro is an extension of SVO [20], a semi-direct VO algorithm that estimates the relative camera pose by minimizing the photometric error between pixels corresponding to the projected location of the same 3D points. However, feature-based methods are used for optimizing the estimated motion. Hence, it's a semi-direct method. Consider an interest point in the current frame specified in 2D image coordinates. SVO Pro back-projects this interest point to a 3D world point and subsequently projects it onto the next frame. The positioning of the re-projected point in the current frame depends on the rel-

ative pose between the current and the previous frames. Each interest point is denoted by a patch of 4×4 pixels around the interest point. In SVO Pro, the transformation matrix is optimized to obtain the minimum photometric difference between these image patches. In comparison to SVO, SVO Pro supports different camera models, and has an active exposure control method to maximize the gradient information in the images [7].

Extending the monocular version to the stereo version is done by introducing a body frame B that is rigidly attached to the camera frame with known extrinsic parameters. The algorithm estimates the incremental motion of the body frame by minimizing the photometric error. Using the known extrinsic parameters, the relative pose of the cameras with respect to the body frame can be obtained.

The implementation is available in C++. SVO Pro was implemented on an indoor dataset collected from a camera attached to a MAV and sequences from a handheld camera, with ground truth obtained from a motion tracking system.
Project URL: `http://rpg.ifi.uzh.ch/svo_pro.html`

2.3.5. COLMAP

COLMAP is a general-purpose SfM and Multi-View Stereo pipeline for reconstruction of ordered and unordered images [21] and [22]. It starts with feature extraction and matching, followed by geometric verification. For a given set of input images, scene overlap and the projections of the same points in the overlapping images are identified. The overlapping images are then verified by estimating a transformation that maps feature points between images using projective geometry. If a valid transformation maps a sufficient number of features between the images, they are considered geometrically verified. The geometrically verified image pairs are represented as a scene graph, with images as nodes and verified image pairs as edges. The reconstruction stage starts with a carefully selected two-view reconstruction. Subsequently, new images are registered by solving the Perspective-n-Point (PnP) problem using matches between the 3-D world points (triangulated from previous frames) and 2-D image points (current image). With each new image, there might also be additional feature points that extend the scene coverage. When a minimum of two images cover the same scene containing these new points, they are triangulated to increase the stability through redundancy. Finally, the reconstruction is refined using BA.

The COLMAP SfM and Multi-View Stereo pipeline has a graphical and command-line interface.
Project URL: `https://colmap.github.io/`

3

Experimental Setup and Data Acquisition

In order to study the performance of the algorithms, datasets were collected using store bought tomatoes, and table plants. A mobile camera was used for the collecting the monocular images and an Intel RealSense Depth Camera D455 was used for collecting stereo images.

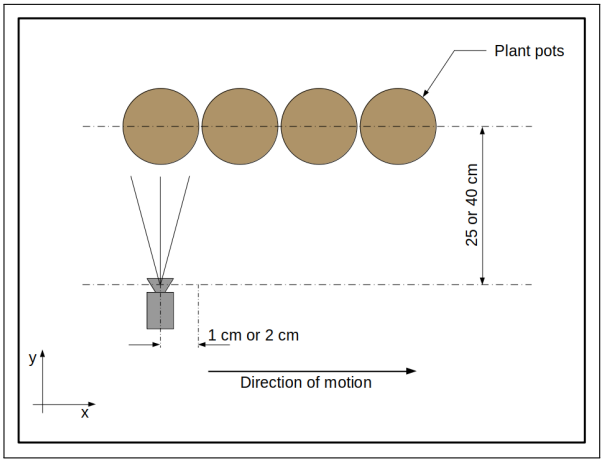

Figure 3.1: Experimental Setup

Figure 3.1 shows the pictorial representation of the top view arrangement of the setup. The setup had four table plants positioned next to each other with the tomatoes on the plants. The camera was positioned facing the plants, at a height of 22 cm, and at a fixed distance of 25 cm or 40 cm from the plants with an expected positional accuracy of 3 mm. To prevent unintended camera motion during data collection, the mobile phone

was fixed to a holder and the stereo cameras were fixed to a tripod. The cameras fixed to the holder or the tripod stand, was placed on a cuboidal box. The frame to frame motion was obtained by moving the cuboidal box along a guided line parallel to the plants. The images were captured in a stop and go method. After each image is captured, the cuboidal box is moved by 1 cm or 2cm along the guided line and then the next image is captured. The expected accuracy of the frame to frame camera displacement is 2mm.

3.1. MONOCULAR DATASETS

Four different datasets were obtained using the monocular camera, with changes in camera resolutions, camera to plants distance, and frame to frame displacement.

- **Dataset 1:** Number of images = 21; camera resolution = 1280 x 720 pixels; camera to plants distance = 25 cm; frame to frame displacement = 2 cm.
- **Dataset 2:** Number of images = 40; camera resolution = 1280 x 720 pixels; camera to plants distance = 25 cm; frame to frame displacement = 1 cm.
- **Dataset 3:** Number of images = 37; camera resolution = 4032 x 3024 pixels; camera to plants distance = 25 cm; frame to frame displacement = 1 cm.
- **Dataset 4:** Number of images = 23; camera resolution = 4032 x 3024 pixels; camera to plants distance = 40 cm; frame to frame displacement = 1 cm.

Figure 3.2: Each row contains the first four consecutive frames from each dataset. The rows are in the order of the datasets.

The camera was calibrated to obtain the intrinsic parameters using a standard calibration checkerboard and MATLAB's camera calibration toolbox. To perform the camera calibration, around 20 images of a 10 x 7 checkerboard were captured by the mobile camera. Separate set of images were captured for the two camera resolutions. Using these

images and MATLAB's Camera Calibrator app, the intrinsic parameters of the camera were obtained. Table 3.1 contains two sets of parameters obtained for the two image resolutions.

Table 3.1: Intrinsic Parameters of the mobile camera.

Parameters	**Camera Resolution (Pixels)**	
	1280 **x** 720	4032 **x** 3024
Focal Lengths (Pixels)	[1007.6908, 1007.6336]	[3148.4217, 3156.1837]
Principal Point (Pixels)	[639.5568, 361.3941]	[2012.2010, 1506.4947]
Radial Distortion	[0.0985, −0.3629]	[0.0324, −0.0917]
Tangential Distortion	[0, 0]	[0, 0]
Skew	0	0

3.2. STEREO DATASETS

The stereo images were captured by two camera sensors with a baseline of 95 mm. Three different datasets were obtained using the stereo cameras, with changes in the camera resolution and camera to plants distance.

- Dataset 1: Number of image pairs = 22; camera resolution = 640 x 480; camera to plants distance = 25 cm; frame to frame displacement = 1 cm;
- Dataset 2: Number of images = 18; camera resolution = 1280 x 720; camera to plants distance = 25 cm; frame to frame displacement = 1 cm.
- Dataset 3: Number of images = 11; camera resolution = 1280 x 720; camera to plants distance = 40 cm; frame to frame displacement = 1 cm.

Rectified grayscale images were obtained directly from the stereo cameras. The intrinsic parameters for the stereo camera was obtained from the real-sense viewer application. Table 3.2 contains the intrinsic parameters of the stereo cameras for both the image resolutions.

Table 3.2: Intrinsic parameters of the stereo camera.

Parameters	**Camera Resolution (Pixels)**	
	640 **x** 480	1280 **x** 720
Focal Lengths (Pixels)	[384.8152, 384.8152]	[641.3586, 641.3586]
Principal Point (Pixels)	[319.5187, 237.9671]	[639.1979, 356.6119]
Radial Distortion	[0.0, 0.0]	[0.0, 0.0]
Tangential Distortion	[0, 0]	[0, 0]
Skew	0	0
Baseline (mm)	95.1063	95.1063

Figure 3.3: Each row contains a pair of stereo images from each dataset. The rows are in the order of the datasets

4

Results

Since the experimental setup was designed to emulate the deleafing robot's motion on the guided rails, the performances of the algorithms are evaluated using the estimated camera displacement along the x-axis only (Figure 3.1). Therefore, the algorithms were run on the datasets to estimate the individual camera poses, and subsequently, using the camera poses, the frame to frame displacement along the x-axis was calculated.

4.1. Image Registration using Correlation & Feature Displacement

The translation between two images estimated using MATLAB's 'imregcorr' function will be in image pixels. The estimated translation can be used for the comparison study only if the number of pixels per unit length is known. This value depends on the distance of the camera from the plants, frame to frame displacement, and image resolution. Hence, it should be computed individually for each dataset.

4.1.1. Monocular Datasets

In each of the four datasets, the entire image does not contain leaves or tomatoes. There are regions in each image with little to no leave or tomatoes. Therefore, to compute the number of pixels per unit distance and the translation, the images were cropped to remove the regions without leaves or tomatoes. This also improved the results of the 'imregcorr' function.

SURF features were detected and matched between two successive frames, and the corresponding feature displacements in pixel coordinates were calculated. The SURF feature detection was done with 3 octaves, 4 scale levels per octave, and a strongest feature threshold of 1000. Out of the detected SURF points, 500 points with the strongest metric value were used for matching. The incorrect matches were filtered out with a user-defined threshold for maximum feature displacement along the horizontal and vertical directions. Using the matched features, the median feature displacement between two consecutive images is calculated in image pixels. Finally, the number of pixels per

Figure 4.1: Regions with minimal leaves or tomatoes removed in Dataset 2.

unit distance (meter) is obtained using the median feature displacement, and frame to frame camera displacement.

Figure 4.2: Filtering out incorrect feature matches.

MATLAB's built-in function 'imregcorr' gives the translation between two consecutive images in image pixels. The frame to frame displacements along the x-axis are computed using the number of pixels per unit distance and the estimated translation. Figure 4.3 shows the results for all monocular datasets obtained using the mobile camera. The frame to frame displacements calculated using the median feature displacements and the number of pixels per unit distance is also shown for qualitative comparison.

4.2. LIBVISO2

4.2.1. MONOCULAR DATASETS

The monocular version of LIBVISO2 assumes that the camera is moving at a known and fixed height over the ground. It uses the 8-point algorithm for estimating the fundamental matrix. The camera poses are estimated up to an unknown scale value. This scale value is computed using the frame-to-frame displacement obtained from the correlation method. The results obtained from running LIBVISO2 on datasets 1 and 2 are shown in Figure 4.4.

LIBVISO2 did not successfully estimate the camera poses for all the frames in datasets 3 and 4. In pose estimation, using the estimated fundamental and essential matrices, the rotation matrix, translation vector, and 3D coordinates of the matched points are computed upto an unknown scale value. Subsequently, for all the points that fall in front of the image plane, the sum of the absolute values of the x,y, and z coordinates are computed in meters. If this median is greater than the pre-set motion threshold value (100 m), the pose estimation fails for the corresponding frame. Figure 4.5 shows the results for datasets 3 and 4, obtained by setting larger motion threshold values. To prevent the pose estimation failure for dataset 3, the motion threshold had to be greater than 365 m.

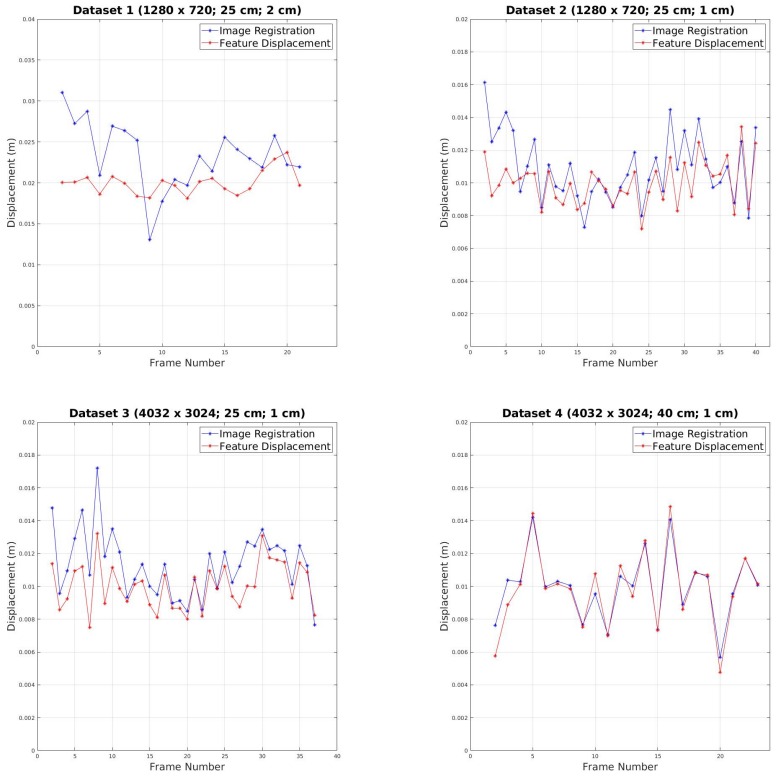

Figure 4.3: Estimated camera displacement using phase correlation for the monocular datasets. For each dataset, the results obtained using the median feature displacement is also shown. The values mentioned in the titles are image resolution in pixels, camera to plants distance, and frame to frame displacement for the corresponding dataset.

And for dataset 4, it had to be greater than 105 m. The estimated poses for dataset 3 show the algorithm's performance when the motion threshold is increased by more than 3.5 times the initial value. The parallax effect reduces in the features that are relatively far away from the camera, thereby negatively affecting the performance of the algorithm. The threshold of 100 m ensures that only feature points that are closer to the camera are used for pose estimation.

4.2.2. Stereo Datasets

Pose estimation using LIBVISO2 failed for all three stereo datasets. The minimum required number of feature matches for LIBVISO2 to perform pose estimation was 6. The feature matching algorithm in LIBVISO2 did not produce enough matches for more than half the frames in all the stereo datasets.

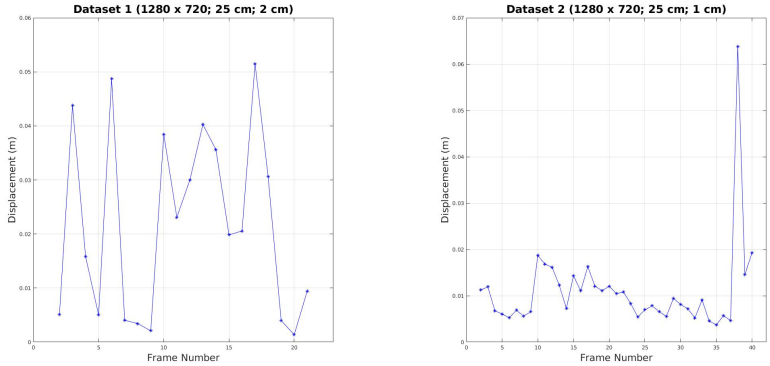

Figure 4.4: Estimated camera displacement using LIBVISO2 for the monocular datasets 1 & 2.

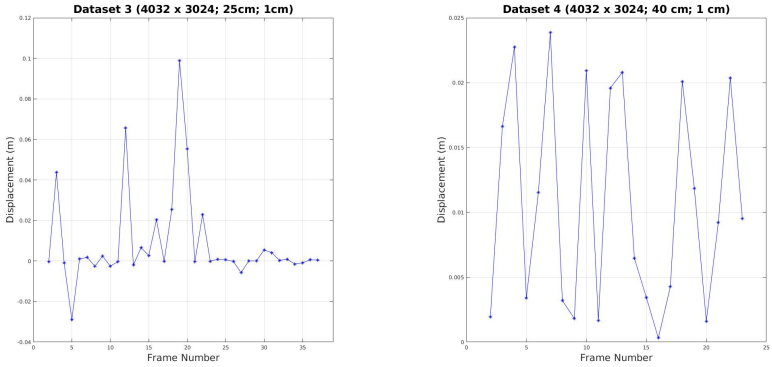

Figure 4.5: Estimated camera displacement using LIBVISO2 with increased motion threshold for monocular datasets 3 & 4.

4.3. ORB-SLAM22

4.3.1. MONOCULAR DATASETS

Running the ORB-SLAM22 on the monocular dataset did not result in any feature matches. Subsequently, the feature detection and description parameters were tuned to obtain feature matches and pose estimates. Below are the default values of the ORB parameters in ORB-SLAM2:

- Number of features per image = 2000
- Scale factor = 1.2
- Number of levels in scale pyramid = 8
- Default FAST threshold = 20

- Minimum FAST threshold = 7

ORB feature detection uses a multi-scale image pyramid, and the scale factor is the pyramid decimation ratio. The value of the scale factor is always greater than 1. A higher value of 2 will significantly degrade the feature matching scores. The algorithm was run with different scale factor values starting from 1.2 to 1.5 with an increment of 0.1 during each run.

For ORB feature detection and description in an image of size MxN in MATLAB, the maximum number of levels in the multi-scale image pyramid is limited by the

$$\text{level}_{\text{max}} = \left(\frac{\log(\min(M,N)) - \log(63)}{\log(\text{Scale factor})} \right) + 1 \tag{4.1}$$

If the value of $\text{level}_{\text{max}}$ is a fraction, the nearest integer less than $\text{level}_{\text{max}}$ is chosen. The number of levels in the scale pyramid, calculated using equation 4.1 and scale values (1.2 to 1.5), ranges from 7 to 14 levels for the images of size 1280 x 720, and 10 to 22 levels for images of size 4032 x 3024.

In ORB-SLAM2, the default threshold value for FAST feature extraction is 20. If no feature points are detected, the minimum threshold value of 7 is used instead of 20. The algorithm was run with different minimum threshold values ranging between 3 to 7 with an increment of 1 during each run. Irrespective of any of the above-mentioned changes to the ORB parameters, there were no feature matches and hence, no pose estimates for all of the datasets.

4.3.2. Stereo Datasets

While ORB-SLAM2 failed to estimate poses for the monocular datasets, the stereo version did not fail. Similar to the monocular setting, using the pose estimates, the frame to frame displacement of the left camera is computed. Figure 4.6 shows the results for the stereo datasets.

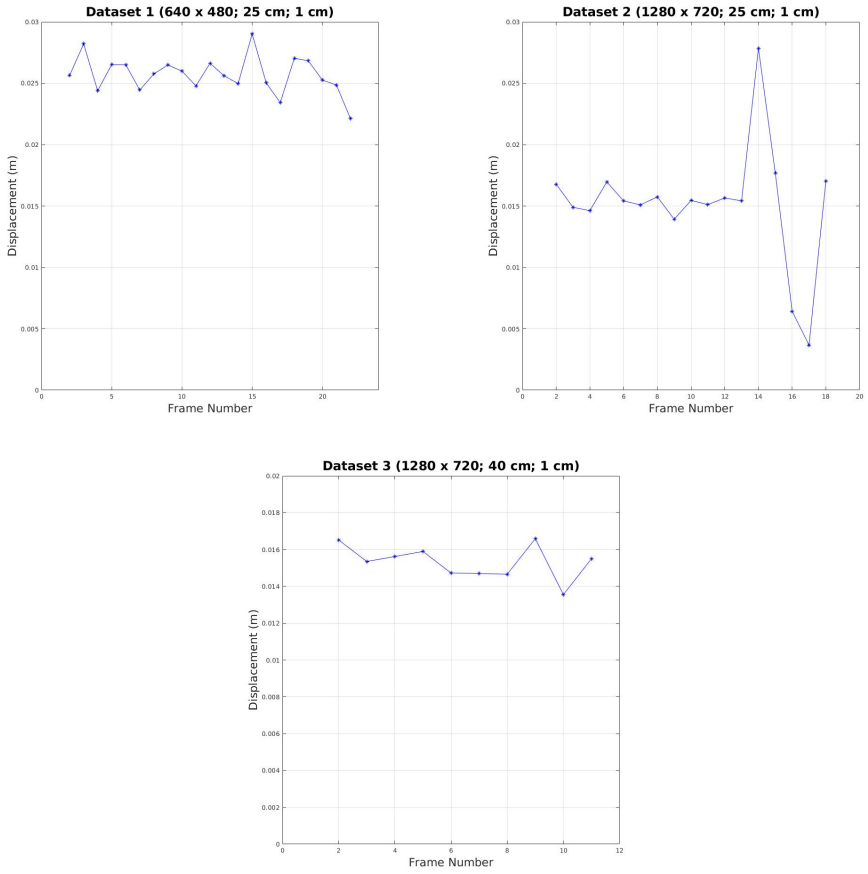

Figure 4.6: Estimated camera displacement using ORB SLAM for the stereo dataset.

4.4. SVO PRO

4.4.1. MONOCULAR DATASETS

Running SVO Pro on the datasets 1, 2 and 3 did not result in pose estimates beyond the third frame due to lack of matched features. Beyond the third frame, the number of matched features keep decreasing with each new frame. However, SVO Pro did estimate the relative camera poses for the first 19 frames in dataset 4, in which the camera was positioned at a distance of 40 cm from the plants as opposed to 25 cm in the other sequences. It should be noted that, unlike the results obtained from the other VO algorithms, the estimated poses were not scaled using the ground truth. Figure 4.7 shows the frame to frame displacement for the dataset 4.

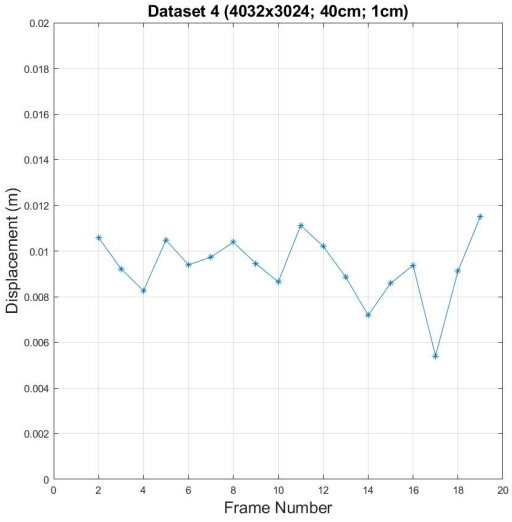

Figure 4.7: Estimated camera displacement using SVO Pro for monocular dataset 4

4.4.2. STEREO DATASETS

Similar to SVO Pro's performance on the monocular dataset, pose estimation failed for datasets containing images captured with the camera positioned at a distance of 25cm from the plants. The results obtained from running SVO Pro on the stereo dataset 3 is shown in Figure 4.8. Irrespective of the camera used (monocular or stereo), SVO Pro estimates poses only for the datasets where the camera was positioned at 40 cm from the plants.

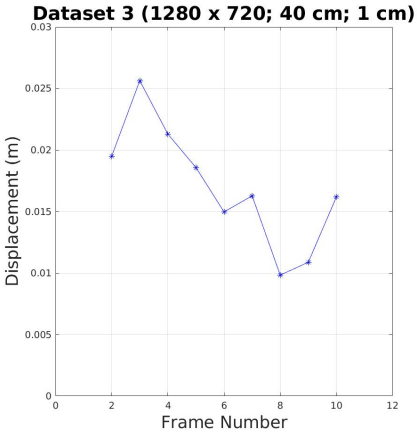

Figure 4.8: Estimated camera displacement using SVO Pro for stereo dataset 3.

4.5. COLMAP

COLMAP was run on the monocular dataset to recover the camera poses corresponding to each image. COLMAP is a general-purpose Structure-from-Motion (SfM) and multi-view stereo pipeline. The pipeline recovers a sparse representation of the scene and the camera poses pertaining to all the input images. Unlike VO algorithms, COLMAP isn't an active pose estimation algorithm. The poses aren't estimated with each new image. It is an offline process. However, the pose estimation can be used to compare with the poses estimated from algorithms. In COLMAP, results were obtained by running the algorithm only on the monocular datasets. The camera poses were estimated up to an unknown scale value. The scale value was computed using the frame to frame displacement obtained from the correlation method.

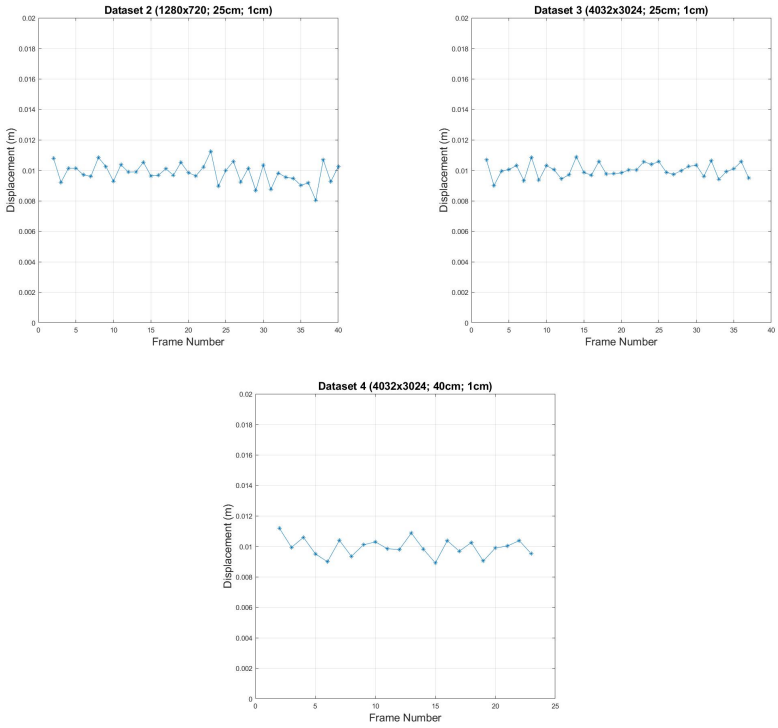

Figure 4.9: Estimated camera displacement using COLMAP.

4.6. QUANTITATIVE COMPARISON

For all of the above mentioned algorithms, the median and standard deviations of the frame to frame displacements were calculated for quantitative comparison. Table 4.1 contains the the results for the monocular datasets obtained using the mobile camera.

Table 4.1: The median and standard deviation of the estimated frame to frame displacement for the monocular datasets.

	Image Reg. (mm)		**LIBVISO2 (mm)**		**SVO Pro (mm)**		**COLMAP (mm)**	
	median	**std**	**median**	**std**	**median**	**std**	**median**	**std**
Dataset 1	21.4980	4.7165	20.2014	17.1177	-	-	-	-
Dataset 2	9.9723	1.7793	8.3293	9.6282	-	-	9.8422	0.6833
Dataset 3	11.2010	1.7547	0.5479	23.2351	-	-	10.0035	0.4676
Dataset 4	10.7952	2.4345	9.3586	8.3815	9.7756	1.5029	9.9107	0.5887

The results of running LIBVISO2 on the monocular datasets 1 and 2 show that the variance of the estimated displacements for dataset 2 is less than the variance for dataset 1. While both the datasets have images of to same resolution, the frame to frame displacement in dataset 2 is half the displacement in dataset 1. Therefore, dataset 2 has more overlapping regions between two consecutive images than dataset 1. Furthermore, in LIBVISO2, the average number of feature matches in dataset 2 was 57% more than in dataset 1, and the average percentage of RANSAC inliers was 54.43% in dataset 2, and 41.45% in dataset 1. Therefore, larger frame to frame displacement would lead to relatively poorer performance of VO algorithms. This condition is even more critical in the case of the datasets collected for this project because the images contain objects with depth values that do not change drastically.

Increasing the image resolution should increase the number of feature points detected, and subsequently the number of feature matches. Evidently, the average number of feature matches produced by running LIBVISO2 on dataset 3 was 57% more than the matches in dataset 2. However, LIBVISO2 performed poorly on dataset 3 with an estimated median frame to frame displacements of 0.5479 mm. The performance of LIBVISO2 improved for dataset 4, with the standard deviation being the least of all the monocular datasets. While, datasets 3 and 4 have the same image resolution and frame to frame displacement, the camera to plants distance is 25 cm for dataset 3 and 40 cm for dataset 4. Furthermore, dataset 4 had 400% more matches than dataset 3, and the average percentage of RANSAC inliers was 84.68% for dataset 4, and 39.05% for dataset 3. This is because, with the increase in the image resolution, more feature matches are needed to produce reliable pose estimates. Between datasets 2 and 3, even though the resolution increased, the overlapping regions remained the same. In dataset 4, the camera was positioned at a distance of 40 cm from the plants for the same frame to frame displacement as dataset 3. This created more overlapping regions, and subsequently more feature matches. Additionally the monocular version of SVO Pro estimated results only for dataset 4, and also performed well without the need for scaling the estimated displacement.

Image registration works best for dataset 2 since it contains images with the lowest parallax effect. Note that the image registration techniques do not use feature detection and mapping. COLMAP failed to estimate poses for dataset 1, and the results for datasets 2, 3, and 4 weren't drastically different from each other.

Table 4.2: The median and standard deviation of the estimated frame to frame displacement for the stereo datasets

	LIBVISO2 (mm)		**ORB SLAM2 (mm)**		**SVO Pro (mm)**	
	median	**std**	**median**	**std**	**median**	**std**
Dataset 1	-	-	25.6354	1.5367	-	-
Dataset 2	-	-	15.4181	4.9234	-	-
Dataset 3	-	-	15.4245	0.9310	16.2737	4.9476

Table 4.2 contains the results for the stereo datasets. For the stereo datasets, while LIBVISO2 failed to estimate poses for all datasets and SVO Pro failed for the datasets 1 and 2, ORB-SLAM2 successfully estimated the camera poses for all the three datasets. Datasets 1 and 2 were collected with the same camera to plants distance, and datasets 2 and 3 contain images of the same resolution. The results of ORB-SLAM2 also shows that the performance on dataset 3 is better than the performance on datasets 1 and 2. SVO Pro estimated results only for the dataset with the camera positioned at 40 cm from the plants in the stereo setting as well.

5

Conclusions and Future Work

This thesis compared different pose estimation techniques for a deleafing robot, by comparing each algorithm's performance on multiple datasets collected using store-bought table plants and tomatoes. The motivation behind collecting multiple datasets was the lack of availability of a greenhouse dataset with ground truth for VO. The thesis focused only on the motion of the deleafing robot on the guided rail. The experimental setup for data collection prioritized the feasibility of recording the ground truth along with the images and emulating the robot's motion on the guided rail in a greenhouse environment as much as possible.

5.1. Addressing Research Questions

1. **Most of the existing publicly available VO algorithms have been developed for an urban environment or an indoor space. How will these algorithms perform on datasets that emulate the greenhouse environment?**
   For the monocular datasets, while ORB-SLAM2 failed to estimate poses for all the datasets, LIBVISO2 and SVO perform well for images in dataset 4. For the stereo dataset, while LIBVISO2 failed to estimate poses for all the datasets, ORB-SLAM2 and SVO perform well for images in dataset 3. For both the cameras, the VO algorithms perform better on datasets containing images with resolution of 4032 x 3024 pixels, collected with the camera positioned at 40 cm from the plants.

2. **Which camera (monocular or stereo) is best suited for the deleafing robot? And how does the camera's positioning relative to the plants affect the performance of the algorithms?**
   The results show that positioning the cameras relatively close to the plants (25 cm) negatively impacted the performance of the VO algorithms, especially for the datasets with images of higher resolutions. The results also show that the monocular camera is better suited for the deleafing robot, with the camera facing the plants and its viewing direction perpendicular to its direction of motion on the rails. The only drawback with monocular versions of LIBVISO2 and COLMAP is

that the poses are estimated up to an unknown scale value. The pose estimation improved with increase in the image resolution

3. **What is the best achievable accuracy of VO on datasets that emulate the greenhouse environment?**
   For the monocular dataset 4, SVO Pro estimated a median frame to frame displacement of 9.7756 mm, and with a standard deviation of 1.2 mm without the need for computing the scale value. This once again shows that the monocular camera is better suited for the deleafing robot.

5.2. FUTURE WORK

Using the available results and inferences, more insights into the pose estimation of a mobile robot in a greenhouse environment can be obtained by addressing the following questions:

1. In all of the datasets, the mobile camera's viewing direction was perpendicular to its direction of motion. How will the performance be affected if the camera is slightly tilted toward the direction of motion? The images obtained in this setting will contain objects with varying depth values, and how would it affect the performance of the VO algorithms? Will the stereo cameras be better suited than the monocular camera for this setting?

2. Increasing the frame to frame displacement leads to relatively poorer performance of the VO algorithms. Therefore, collecting images by recording a video with 5 or 10 fps instead of the stop and go technique might further improve the performance of VO algorithms. However, the issue with recording a video is the difficulties in recording the ground truth. This issue can be addressed by scaling up the experimental setup with guided rails, bigger plants, and a motion tracking system. How will these changes to the data collection process affect the performance of the VO algorithms?

3. Can learning algorithms be used for pose estimation in a green house environment? To use learning algorithms for end-to-end pose estimation, a significant amount of data with ground truth is required. Even a scaled-up experimental setup will produce only a limited number of images with ground truth. However, with the limited number of images, would it be possible to generate synthetic greenhouse datasets for VO? In the datasets collected for this thesis, the feature displacement between two consecutive images does not vary significantly. Can the mean feature displacement with a tolerance value be used as prior for generating the synthetic datasets?

BIBLIOGRAPHY

[1] Richard Szeliski. *Computer vision: Algorithms and applications*. Springer Cham, 2022.

[2] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[3] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, Dec 2011.

[4] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[5] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[6] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, 2011.

[7] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017.

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[9] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011.

[10] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

[11] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, Dec 2013.

[12] Daniel I. Barnea and Harvey F. Silverman. A class of algorithms for fast digital image registration. *IEEE Transactions on Computers*, C-21(2):179–186, 1972.

[13] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *2010 IEEE Intelligent Vehicles Symposium*, pages 486–492, 2010.

[14] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011.

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012.

[17] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[18] Mike Smith, Ian Baldwin, Winston Churchill, Rohan Paul, and Paul Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, 2009.

[19] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[20] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014.

[21] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[22] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.