Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 15-010

Robotic Surveying of Apple Orchards

Pravakar Roy, Nikolaos Stefas, Cheng Peng, Haluk Bayram, Pratap Tokekar, Volkan Isler

June 18, 2015

Revised: July 27, 2015

Robotic Surveying of Apple Orchards

Pravakar Roy[*], Nikolaos Stefas, Cheng Peng, Haluk Bayram, Pratap Tokekar[†], Volkan Isler
Department of Computer Science and Engineering, University of Minnesota

July 27, 2015

Abstract

We present a novel system for surveying apple orchards by counting apples and estimating apple diameters. Existing surveying systems resort to active sensors, or high-resolution close-up images under controlled lighting conditions. The main novelty of our system is the use of a traditional low resolution stereo-system mounted on a small aerial vehicle. Vision processing in this set up is challenging because apples occupy a small number of pixels and are often occluded by either leaves or other apples. After presenting the system setup and our view-planning methodology, we present a novel method to match and combine multiple views of each apple and report successful results from field trials. We conclude the paper with an experimental analysis of the diameter estimation error.

1 Introduction

Precision Agriculture refers to a collection of methods which aim to optimize farm practices by collecting high resolution data. For example, by predicting the nitrogen level in the soil, fertilizer application can be fine-tuned across a farm. Precision agriculture techniques can lead to tremendous environmental and cost benefits through, for example, targeted irrigation and fertilizer application and early diagnosis of fruit diseases and nutrient deficiencies (Mulla (2013)).

The goal of the work presented in this paper is to provide farmers with a convenient mechanism to obtain high resolution data about the status of their farms or orchards. We specifically focus on surveying apple orchards to collect data related to yield (e.g. number of apples, their colors and sizes, etc.). Existing work on robotic yield estimation uses ground vehicles, sophisticated sensors such as LIDAR and controlled lighting and imaging conditions (Section 2). These methods are not appealing to the farmers due to their cost and inconvenience. In contrast, we investigate whether an inexpensive Unmanned Aerial Vehicle (UAV) equipped with a pair of cameras can be used for yield estimation in standard outdoor conditions. We report promising results that indicate that this can be accomplished without resorting to expensive sensors and controlled imaging conditions.

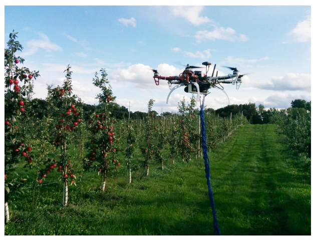


Figure 1: Field trials at the Pine Tree Apple Orchard, White Bear Lake, MN U.S.A. The UAV was tethered with a blue rope for safety purposes only.

Small UAVs are well-suited for automated data collection due to their speed and coverage spans. Rapid advancements in UAV technology

[*]Corresponding author. E-mail: proy@cs.umn.edu. Phone: 612-626-5681. For multi-media associated with this report, please visit http://agricultural-robotics.cs.umn.edu

[†]Currently at Virginia Tech.

have already led to early commercial efforts on using them in agriculture (Dobbs (2014 (accessed Jul 2015), Dillow (2014 (accessed Jul 2015)). UAV technology has matured to the point where many open hardware and software solutions are available (e.g., *DIYDrones* (2015 (accessed Jul 2015)). We envision such UAVs giving farmers the capability to determine useful statistics such as size and count of the yield in an apple orchard using onboard images. In natural settings, obtaining this information becomes challenging due to a number of factors: Occlusions from leaves and branches may prevent getting a good view of apples, colors may be unreliable because of specularities and varying brightness conditions, and lack of globally distinguishable features may cause finding stereo correspondences difficult.

As we review in the next section, existing methods for processing fruit images rely on high-resolution imagery usually taken from a ground platform. However, such images are hard to obtain from a camera mounted on a UAV. A lens with long focal length would limit the field of view, making navigation and coverage very hard. A wide angle lens, such as the one we use in our system, provides good coverage of the scene but apples occupy smaller number of pixels. A camera which provides simultaneously high resolution and wide coverage (e.g Cannon Rebel T5i) would be too heavy for a small UAV.

The resolution issue, combined with the non-planarity of the scene, cause standard techniques such as 3D alignment (Eggert et al. (1997)) and Structure From Motion (Huang & Netravali (1994)) to fail. These methods work well when the images are high resolution and have many distinguished features. In our case, we have very low resolution (640×480) images and the features we want to register are apples, which are very small in pixel resolution (20-40 square pixels) and are smooth and shiny lacking sharp corners. We discuss the performance of these techniques at length in Appendix D and Appendix E.

To overcome these challenges, we present a novel method for registering apples. We exploit the small camera motion within subsequent frames. Since the apple clusters move very little, the motion of the apple clusters can be represented with an affine transformation. We utilize this fact to find the dense correspondences between corresponding apple clusters and register them. Our main contribution is to provide a pipeline for yield estimation - segmenting apple clusters, registering these clusters, counting their numbers from the registered images, and when a cluster is found containing a single apple, estimate its diameter.

We start by describing our prototype system designed using low-cost, commercial, off-the-shelf hardware and software components in Section 3. We then present a view planning strategy to plan for the path of the UAV in order to collect stereo images in an apple orchard (Section 4). Next, we describe computer vision techniques to segment apples in images, register them and combine them with stereo data to estimate useful statistics (size and count) in an apple orchard (Section 5). We demonstrate the feasibility and utility of our system through field experiments conducted in apple orchards (Section 6). Finally, we conclude in Section 8 with a discussion of our ongoing research and long term goals of this project.

2 Related Work

Recently, there has been an increased activity in developing robotics and automation systems for various farming tasks such as Ball et al. (2013), Bergerman et al. (2012), and Bergerman et al. (2013). In this section, we restrict our attention to systems developed for yield estimation and view planning algorithms, which are the most relevant to our work. Wang et al. (2013) presented a system for automatic apple yield estimation in orchards. Their system in contrast to ours, uses a side facing vertical stereo rig and operates in controlled artificial illumination settings at night. The system is mounted on an autonomous ground vehicle which can move between orchard rows. Das et al. (2015) developed a sensor suite for extracting plant morphology, canopy volume, leaf area index and fruit count that consists of a laser range scanner, multi-spectral cameras, a thermal imaging camera and navigational sensors. In comparison, we have a simple setup with a small UAV with only onboard stereo cameras for fruit sensing.

Mao et al. (2009) developed methods for detecting Fuji apples based on color. Jimenez et al. (2000) presented a survey of existing computer vision methods for locating fruit on trees. These methods rely on high-resolution images taken from a stationary platform. In this paper, we focus on estimating apple yield using low resolution stereo cameras mounted on a small, low-cost UAV platform. We capture the images from a UAV while flying which makes vision based tasks difficult. We also register the images based on image features only. Linker et al. (2012) developed a system for estimating the number of apples from color images in orchards under natural illumination. They avoided specular lighting conditions by capturing images close to sunset. Our work does not rely on any particular assumption about the time of day or the position of the sun.

Other related work include the work of Hung et al. (2012) who presented an algorithm to detect tree crowns using overhead aerial images from a UAV flying at a high altitude. Jagbrant et al. (2015) used a 3D LIDAR to detect and localize trees from a robot navigating between tree rows.

Relevant to our work, Won et al. (2013) studied the problem of planning views for a UAV to build a 3D model of the tree. Their algorithm builds a voxel-based representation of the tree and generates the best viewing location from where to obtain the next image. They evaluate their algorithm through simulations. Our focus in this paper is yield estimation and for that purpose we register the images and count the apples from the all the views we captured for an apple cluster.

In our prior work Tokekar et al. (2013), we have presented a symbiotic robotic system consisting of aerial and ground robots for estimating the nitrogen levels in a plot. Our goal in Tokekar et al. (2013) was to develop a set of data collection algorithms for the UAV to estimate nitrogen content from multi-spectral overhead aerial imagery. Pixel values from multi-spectral images can directly be converted into Normalized Difference Vegetation Index, which serves as a proxy for the status of the crop over a large area. In contrast, this paper focuses on the task of estimating relevant apple yield statistics using only close-up color images while navigating within orchard rows. We start with an overview of our system in the following section.

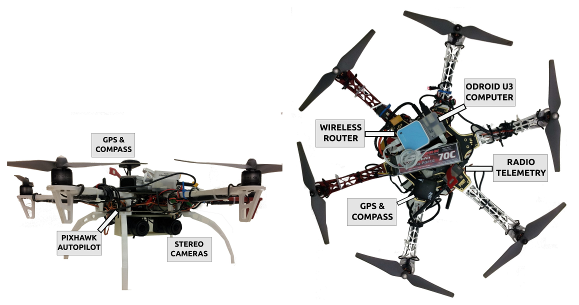


Figure 2: UAV with onboard computer, wireless router, and autopilot. The stereo camera is mounted with a custom frame on the bottom.

3 System Overview

The design goal in our system (Figure 2) was to use low-cost, commercial, off-the-shelf components and leverage freely available, open source software for navigation. The aerial robot is a hexarotor UAV built using the *DJI F550* (2015 (accessed Jul 2015) with a 55 cm wheelbase and 22.86 cm (10 in) propeller blades. A Pixhawk autopilot module (*Pixhawk* (2015 (accessed Jul 2015)) with on-board IMU and external GPS and compass runs the APM:Copter firmware from Ardupilot (*Ardupilot APM:Copter* (2015 (accessed Jul 2015)). The firmware implements the 6 DOF pose estimation and provides autonomous waypoint navigation in environments where GPS signal is available. Maximum flight time is around 30 minutes with a high capacity battery. Flying for extended periods of time requires the operator to pause and switch batteries- a short and trivial procedure.

We use an ODROID-U3 (*ODROID-U3* (2015 (accessed Jul 2015)) as the main computer onboard the robot. The software running on the

ODROID consists of two main components. The first component is the open-source ROS node called MAVROS (*MAVROS* (2015 (accessed Jul 2015)) which interfaces with the autopilot using the MAVLink protocol. The second component is our custom stereo system that uses two Point-Grey Chameleon global shutter cameras (*Point Grey Chameleon USB 2.0 Cameras* (2015 (accessed Jul 2015)). The cameras provide 640×480 resolution colored images at 24 fps and can be focused manually by changing the focal length between $2.8 - 8.0$ mm. During our experiments, we fix the focal length to 2.8 mm.

4 View Planning

In this section we describe our approach for planning the camera views along a row of trees in an apple orchard. Although trees have complex shapes in general, most orchards are designed and managed so that the fruits are easy to pick. Trees are typically aligned in rows with obstacle-free traversable pathway in between. This structure allows for a simple off-line view-planning strategy. Our goal is to take pictures so that every point on the current row is visible in at least one image.

Since the cameras are mounted perpendicular to the frontal direction of motion, the viewing objective can simply be achieved by moving on a viewing plane parallel to coverage plane V. We visit each location in V via a sawtooth trajectory as shown in Figure 3. The advantages of this particular trajectory is that each tree is captured in consecutive images in a bottom-up fashion and movement is limited to a single planar direction. The first assumption results in better image processing. We can easily identify the upward motion of the UAV and recover the set of images covering the same tree. The second results in smaller movement error from the UAV. We describe a strategy that takes advantage of this special type of motion in the image registration Section 5.2 of the vision pipeline.

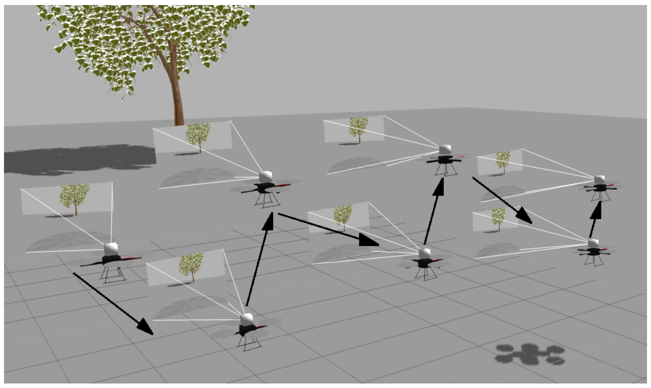


Figure 3: Simulated trajectory using GPS data reported from the experiment.

5 Vision Based Surveying

In this section, we describe the vision pipeline employed by our system described in Section 3. Given stereo images of the apple trees obtained in natural settings, the goal is to, (1) find the location of the apples, (2) count the total number of apples and, (3) estimate their size (i.e. diameter) in the presence of illumination changes, specularities and occlusions.

5.1 Segmenting Apples in Images

A key component of our vision pipeline is segmenting out parts of the image which contain apples. This task is difficult for a number of reasons. First, we have wide angle lenses and low resolution images from a flying UAV. Therefore, the observed apples are extremely small. Second, there are many occlusions. These two problems make it difficult to use shape priors. Third, as we operate in a natural environment with specularities, and in the presence of similarly colored objects such as trunks and apples; color thresholding is difficult. Figure 4b shows the result of color thresholding on the red channel alone.

To get accurate segmentation of the red apples, we start by eliminating all pixels which have dominant blue and green components. These pixels primarily belong to the sky and leaves (Figure 4b). Remaining pixels, which are dominant in red, contain apple pixels as well as pixels associated with the ground, trunks and reddish leaves. To separate the apple pixels, we make use of some domain information. Pixels with very high red values as well as pixels where the difference between red, blue or green channels is very small are likely to be over-bright or specular. Pixels where red and green channels have similar values typically belong to reddish leaves. Pixels where blue is very close to red belong to the trunks. To get from pixels to apples, we perform Canny edge detection (Canny (1986)) on segmented pixels, compute a set of contours and fit a bounding rectangle to each contour. Each bounding box may contain a single apple or an apple cluster. In Section 5.3, we describe how to count the number of apples in each box.We present the details and associated thresholds in Appendix B.

5.2 Image Registration

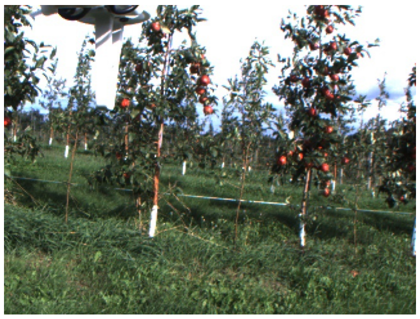
Following the segmentation step, we have apple clusters that overlap across multiple images. Our approach involves bringing these views of the clusters to a set of disjoint reference frames and count from the resultant cumulative views. Calculating rotation R and translation T between consecutive stereo pairs is the state of the art method for registering these type of non-planar image features ((Eggert et al. 1997)). The results of applying this method are presented in Appendix D. We observed that this method fails to successfully register images in our case. After registering the 3D point clouds, we can project them back to a camera frame and count from there. However, stereo data is noisy and the procedure also involves matching among left-right, as well as left-left and right-right pairs of images, resulting in accumulation of errors with increasing number of frames. Another big problem is that, as we are using a very wide angle lens, and the camera motion between consecutive images is very small, the baseline between the images from the same camera is less than the baseline of the stereo pairs. Consequently, we get significantly more left-left / right-right matches compared to left-right matches (Figure 5).

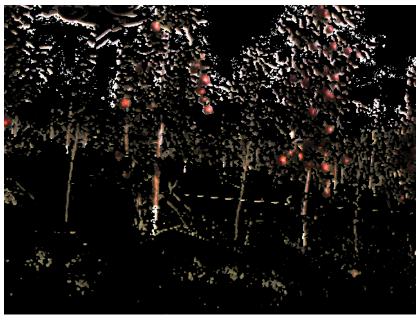
We now present an overview of our approach. The frontal views of the detected clusters in the segmentation step are almost planar within a single cluster. Since the on-board camera captures only the frontal views, there is very little depth variation across the clusters and across multiple images. As these views are consecutive and have a small difference, we assume that parallel lines remain parallel and the relationship between them is affine. Note that the affine relationship assumption is per cluster and not across the entire image. Between every pair of consecutive images, we align individual overlapping clusters with affine transformations. Multiplying the affine transformation matrices, we map a cluster from a later image to an earlier one. To bring non-matching parts of an image to the reference frame we calculate the homography using all the affine matches. This way we find all captured views of a cluster, in a single frame of reference. Note that a separate affine transformation is associated with each cluster in each image. We do not compute a single affine transform to warp an entire image.

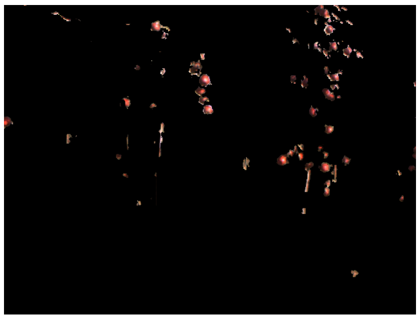
We execute this procedure for all the clusters in an image and count the number of apples from cumulative view. When none of the clusters in the current image matches some clusters in the reference frame we move our reference frame to the current frame. The parts of the new reference frame which contributes to the old reference frame are eliminated using the affine relationship. The change of reference frame is essential to the success of the counting procedure. If we keep bringing back everything to the first frame we will accumulate error and the warped clusters will become smaller and smaller. Figure 6 shows the registration pipeline.

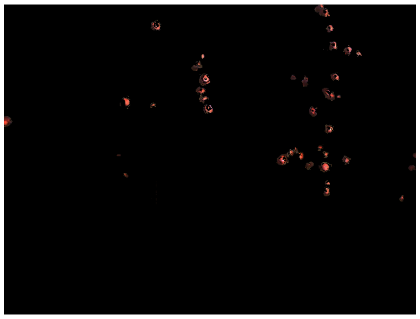
We now describe the following components of the pipeline.

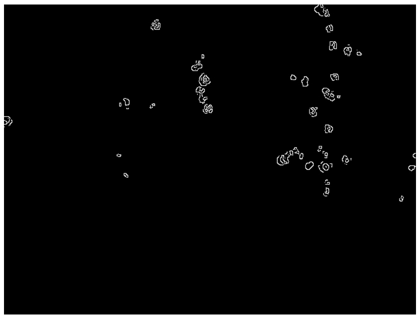
1. Feature Selection
2. Feature Matching
3. Matching Correction
4. Feature Alignment
5. Changing the Reference Frame


(a) Original image captured from the UAV


(b) Blue and green dominant pixels eliminated


(c) Specular, bright trunk and ground pixels eliminated


(d) Remaining trunk and ground eliminated


(e) Canny edge detection

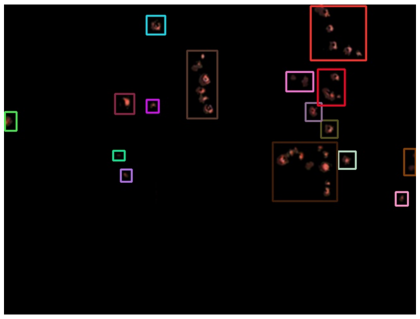
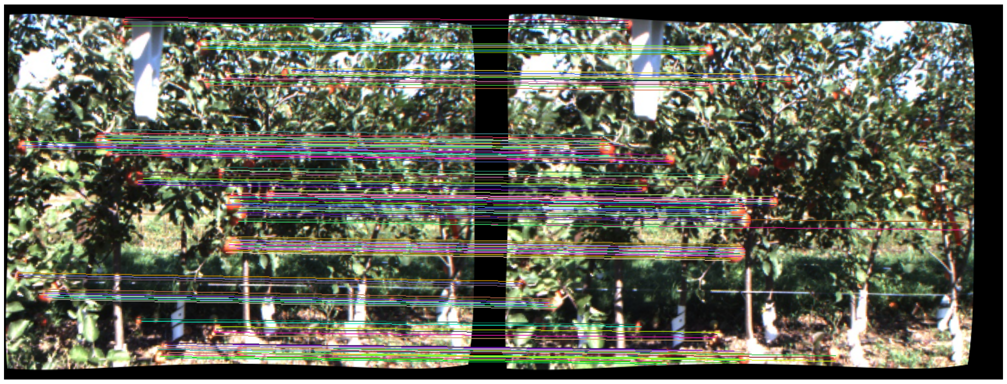

(f) Contours detected

Figure 4: Segmentation Pipeline. (a) The original image captured by the on-board camera. (b) Elimination of blue and green dominated pixels. (c and d) Elimination of specular, trunk and ground pixels. (e and f) Cluster boundaries obtained by canny edge detection and contour extraction.


(a) Matching between two consecutive left images

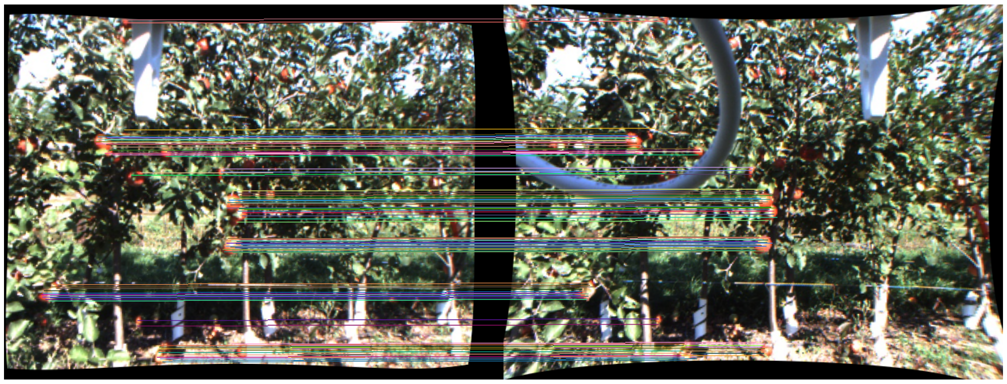

(b) Matching between a stereo pair (left and right image)

Figure 5: Left-left vs left-right matching. Matching two consecutive left images (Figure a) produces many more matches than matching a left and right image taken at the same time (Figure b). This is due to the fact that the stereo baseline is larger than the camera motion. In the figure only 30% of the total matches are shown for clarity.

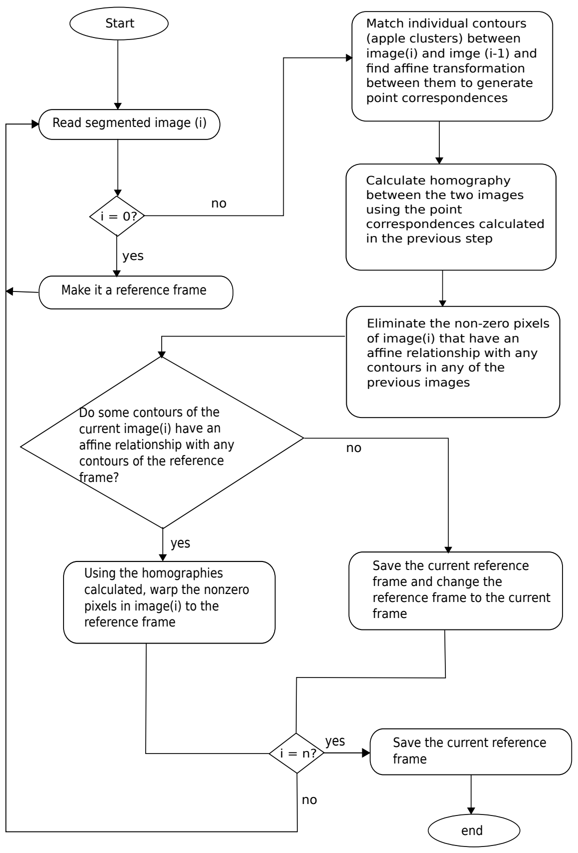

Start
Read segmented image (i)
i = 0?
no
yes
Make it a reference frame
Match individual contours (apple clusters) between image(i) and imge (i-1) and find affine transformation between them to generate point correspondences
Calculate homography between the two images using the point correspondences calculated in the previous step
Eliminate the non-zero pixels of image(i) that have an affine relationship with any contours in any of the previous images
Do some contours of the current image(i) have an affine relationship with any contours of the reference frame?
no
yes
Using the homographies calculated, warp the nonzero pixels in image(i) to the reference frame
Save the current reference frame and change the reference frame to the current frame
i = n?
yes
no
Save the current reference frame
end

Figure 6: The Registration Pipeline. The registration procedure outputs a set of disjoint reference frames required for counting. It starts by reading the segmented images, then calculates individual affine transformations between matching contours to generate point correspondences between images and warps the images to the current reference frame and changes the reference frame when current image and reference frame do not have any common features.

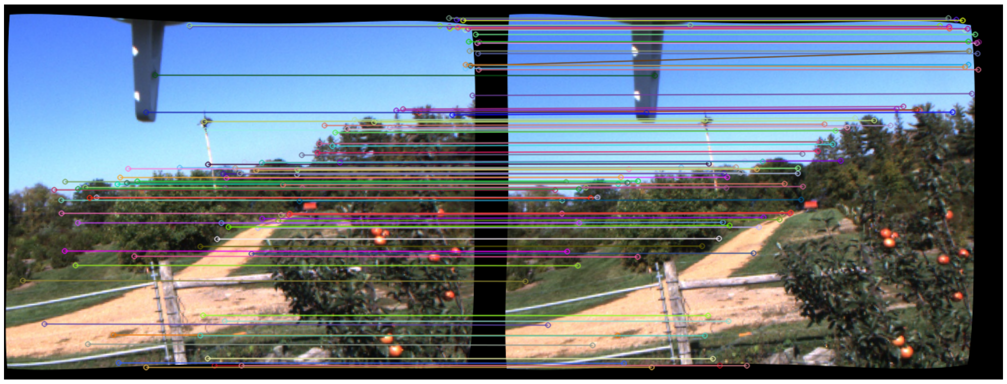
Conventional registration methods have only components one, two and four. We introduce two additional components to improve our matches and make sure the input to our counting procedure is a high resolution contour. In the following, we describe each of the five components in detail.

5.2.1 Feature Selection

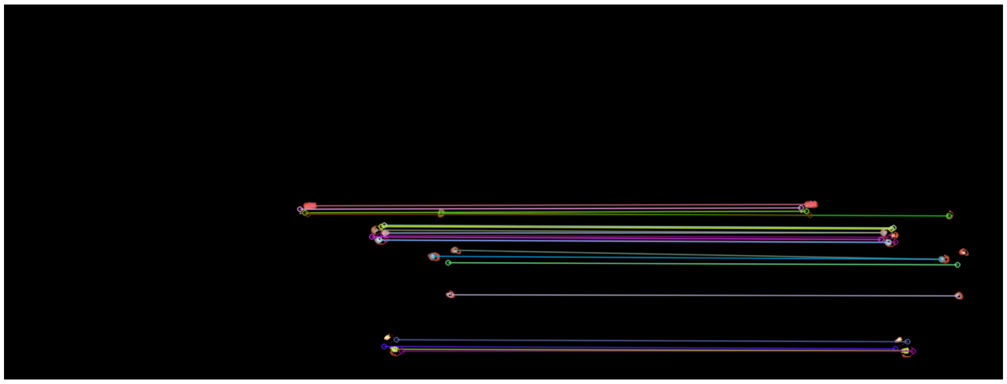
Corner-like features (such as SURF (Bay et al. (2006)) or SIFT (Lowe (1999))) which are commonly used in image registration are not appropriate for registering segmented apple images. This is because apples are smooth and shiny objects without sharp corners. Therefore, we use segmented regions directly as features. This requires novel methods for matching and aligning the features.

5.2.2 Feature Matching

To match the contours and find point correspondences, we utilize our set up to constrain the matching procedure. As the UAV moves 1 m/s and the on-board camera captures 24 frames within that time period, the motion between two successive camera frame is very small. If the direction of the movement does not change abruptly, the overlap between consecutive images should be regular and the matching contours should move very little in an approximately known direction. However, in natural environments, knowing the motion is a strong assumption due to weather conditions like the wind: heavy wind can cause the UAV to drift. Therefore, we assume that the movements are small, but we do not pose any constraints on the direction of the movement. For each apple cluster in one image, we find a corresponding one in the previous image, and keep our search limited within a small neighborhood of the original contour bounding box. The off-the-shelf strategy for this is template matching using normalized cross-correlation (Briechle & Hanebeck (2001)), but apples can look extremely similar, and there are many occlusions for which sometimes we have a wrong match. To avoid such

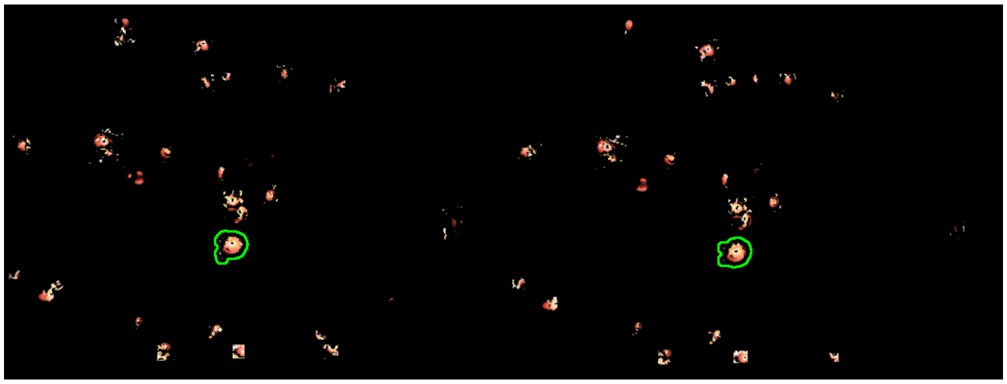


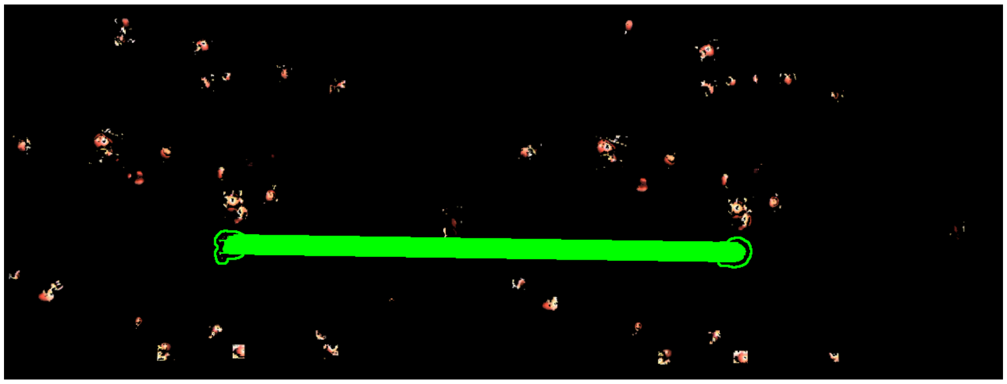
(a) Matching SURF features on a pair of images



(b) Matching SURF features on a pair of segmented images

Figure 7: SURF Feature Matching. Figure (a) shows the SURF features in the entire image. Most of them do not belong to apples. Instead, many of them actually belong to the sky. The number of SURF features on the segmented images (Figure b) is relatively small. They lie on the corners created by the segmentations rather than on the surface of the apples. In contrast, our features (Figure 8) are much more robust and dense than SURF features.


(a) Extracted matched contours


(b) Point correspondence between the contours

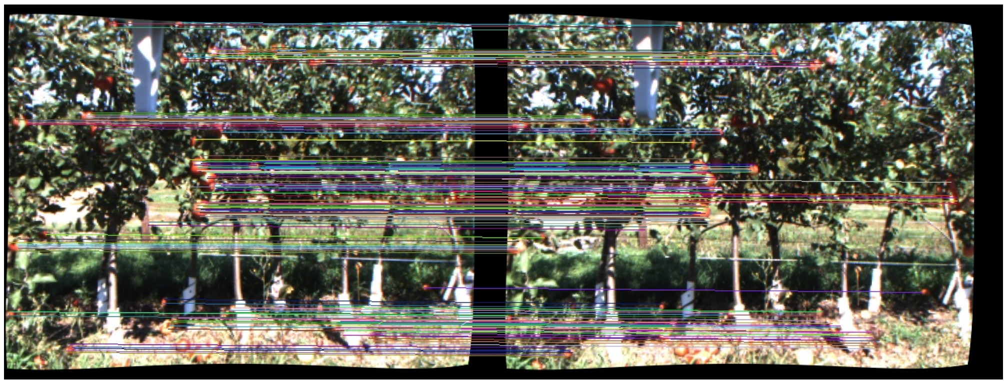

(c) Point correspondence between the full Images

Figure 8: Contour based feature matching. First we find the matching contours using the template matching method described in Section 5.2.2 (Figure 8a). We assume that, within this very small region, the relation between the contours is affine across consecutive frames. Since the solution space is small, we enumerate over the entire space to find the affine transformation which provides us with point correspondences (Figure 8b). Repeating this procedure for all the matching contours provides us with point correspondences between the two images (Figure 8c).

errors we will match both ways in the following way:

- Suppose I_1 and I_2 are the two images and cluster c_2 is in I_2.
- Find a template match for c_2 in I_1 within a small neighborhood of an identity transformation.
- If there is a match, identify the contour c_1 that corresponds to the template matching
- Use template matching to find the match of c_1 in I_2.
- If the resulting contours match, it is deduced that c_2 matches c_1.

With additional motion constraints we can speed up the matching procedure. Two such strategies are described in Appendix C. These matches do not provide point correspondences. To generate point correspondences we have to align the contours to each other. Since a single apple cluster is lying approximately in a 2D plane, and the camera motion is very small, we can assume that they are related by an affine transformation. In order to find the correct affine transformation between the matching contours, we seek to minimize:

$$\sum_x \sum_y I_1(x, y) - I_2(A * (x, y)) \tag{1}$$

where, I_1 and I_2 are the corresponding patches. x, y are pixel locations and A is the affine transformation. In our case, from our off-line path planning algorithm, we know that rotation is extremely small and there is no scaling. Therefore, we primarily solve for translation. Since the solution space is small, we find out the best affine transformation by exhaustive search. Several techniques can be used to speed up this process (Baker & Matthews (2004)).

5.2.3 Matching Correction

Though the template match method is quite robust, some outliers may exist. Since the amount

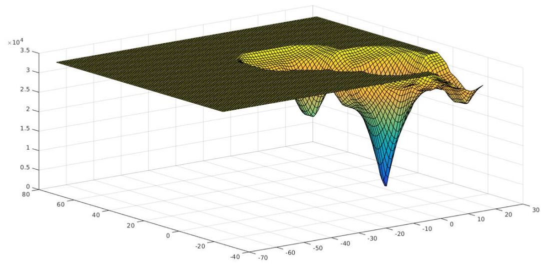


Figure 9: The affine solution space. The search space is small enough that the global minimum can be found by enumeration.

of rotation and scale is very small, the lines connecting the matching points have similar angles. The same can be deduced for the translation in x and y directions. A quartile based outlier removal technique (Hodge & Austin (2004)) is used to get rid of the outliers. We use the inliers to find some of the matches missed by template matching. For the contours having no match, we find corresponding contours within a small boundary and match them using an affine warp. If we find an affine warp which is within the quartile range we add it to the matches.

5.2.4 Alignment of Features

We utilize point correspondences found in the previous step for this procedure. The parts of the contours in subsequent images that correspond to each other in an affine way (i.e the parts of the same apples across different images) can be easily aligned with the calculated affine transformation matrices. For the alignment of parts not related by an affine transformation, we calculate the overall homography between the segmented images using all the affine correspondences. Conventionally, all the images are registered to a single frame of reference. This procedure causes the resolution of the entire image to be reduced with increasing number of frames- as we look at the images from a different perspective than it was originally captured from. Our pixel based counting algorithm performs better with higher resolution. In the next section, we discuss how to change the reference frame after a number of intervals to improve the resolution.

5.2.5 Changing the Reference Frame

In order to keep track of apples across images, we first find contours in each image. We then warp each contour to a reference frame. This way, contours corresponding to the same apple across images are mapped to the same location in the reference frame.

However, as the camera moves, the current view can be significantly different from the reference view. If this happens, the contour in the current view can be mapped to a much smaller area in the reference view. This, in turn, results in significant loss of resolution. In order to solve this problem, we dynamically change the reference frame.

In our approach, each apple is associated with a set of contours and a dedicated reference frame onto which the contours are warped. This way, each contour will be warped to a reference frame which is close to it. That is, the reference frame associated with the contour is close to the original viewpoint for each contour. Therefore the resolution will be preserved.

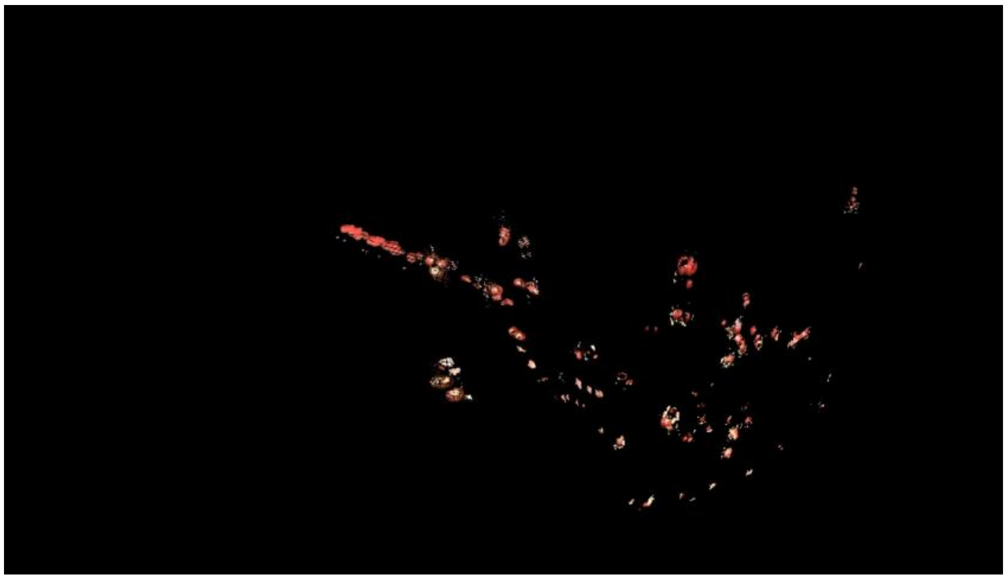
The flow diagram of the procedure is presented in Figure 6. We start with the first image frame as the reference frame. Afterwards, for each image pair, we find the affine transformation between each matched contour as described in Section Feature Mapping. When none of the contours of the current image matches any of the contours of the current reference frame, we make the current image the new reference frame. The procedure guarantees that for each apple, its contours across images are registered to the same region in the reference image without significant loss of resolution. These warped contours are then input to the counting algorithm, which is presented next.

5.3 Counting the Number of Apples

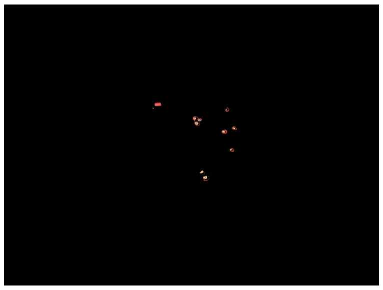
To count the number of apples, mutually exclusive images generated by the registration process (Figure 10) are employed. These views enable us to utilize all available information. From each of these views, contours are detected and an approximate size of the apples and depth is used to select a pixel resolution level – i.e. the size of a pixel window W which would contain a single apple. Given the resolution of the apples, a window of size W is slid through each cluster bounding box. For each W, if more than sixty percent of W contains apple pixels, it is concluded that this subsection of the cluster contains an apple. If the window has less than the required number of apple pixels,the edge of the window e to which most of the apple pixels are at close proximity is found and the neighboring window N ending on this edge is considered. If there are apple pixels close to the sharing edge e in N and together with the previously counted pixels more than the required number is achieved, it is concluded that this is an apple. The groups of pixels identified as apples are marked to avoid double counting. Algorithm 1 outlines this approach.

5.4 Estimating the Diameter of Individual Apples

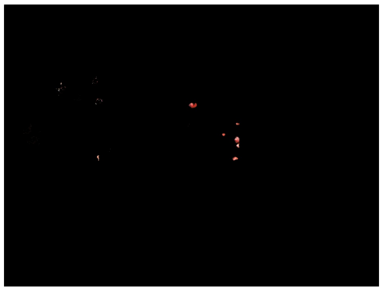
In order to estimate the size of the apples, we estimate their diameters using stereo images. We use the registered segmented images as input for this step. Since calculating the individual apple boundaries from low resolution images is difficult, we limit our approach to clusters which contain only a single apple. To detect clusters containing a single apple, we use the counting algorithm from the previous section. The registration procedure ensures cumulative use of all the information available. We start by computing the 3D coordinates of all points on the contour (of a single apple) with stereo. Since we roughly know the distance from the apples d, we prune all points there at a depth of $d + \Delta_t$. Here, Δ_t models the navigation and modeling uncertainty (we use $\Delta_t = 1\,\text{cm}$). From the remaining points, we consider all pairs of points which have similar depth (less than 1 cm in our implementation). The reasoning behind this choice is simple - the camera captures frontal views which are almost planar resulting in minimal depth difference. All the remaining points ideally belong to the visible surface of an apple. To find the diameter



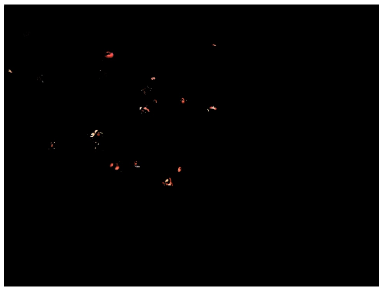
(a) Frames warped to a single frame of reference resulting in loss of resolution



(b) First reference frame in our approach



(c) Second reference frame



(d) Third reference frame

Figure 10: As we use a pixel based counting method, counting from the total image is difficult as warping results in poor resolution. Therefore we break up the frames when the current frame has no matches with the reference frame and make the current frame a new reference frame. Here we show results across twenty five frames. Figure (a) shows the combined image which has poor resolution as a result of warping. Figures b,c and d show the breaking up of the total frame into three disjoint frames. We count from the independent reference frames using our counting procedure described in Section 5.3.

we just fit a circle to this points. The diameter of this circle provides an estimate of the radius of the apple. Figure 11 demonstrates the stereo pipeline.

```
1:  INPUT:
    IMG: best contour of the apple cluster across frames at a particular location,
    $res$: resolution of the apples to be detected $w \times h$,
2:  OUTPUT:
    count: total number of apples in cluster
3:  begin
4:  make a copy of IMG and assign to $I$.
5:  pad $I$ so that its resolution is a scalar multiple of $res$
6:  $count \leftarrow 0$
7:  for all pixels $I(x, y)$ where $x - res.w < I.width \wedge y - res.h < I.height$ do
8:      consider a rectangular window $rec$ of dimension $res.w \times res.h$ starting from $I(x, y)$
9:      $pCount \leftarrow$ total number of Apple pixels in the rectangle
10:     if $\frac{pCount}{area(rec)} > .6$ then
11:         count = count +1
12:         set all the RGB components of $I(x, y)$ inside $rec$ to zero
13:     else
14:         find the edge of the $rec$ for which the number of apple pixel nearest to it is maximum
15:         let $tCount$ be the number of pixels close to this edge
16:         $nCount \leftarrow$find the pixels in the neighboring rectangle which are nearest to this edge
17:         if $\frac{tCount+nCount}{area(rec)} > .6$ then
18:             count = count +1
19:             set all the RGB components of the pixels
                counted in this step to zero
20:         end if
21:     end if
22: end for
23: end
```

Algorithm 1: Counting the number of Apples from Segmented Apple Cluster

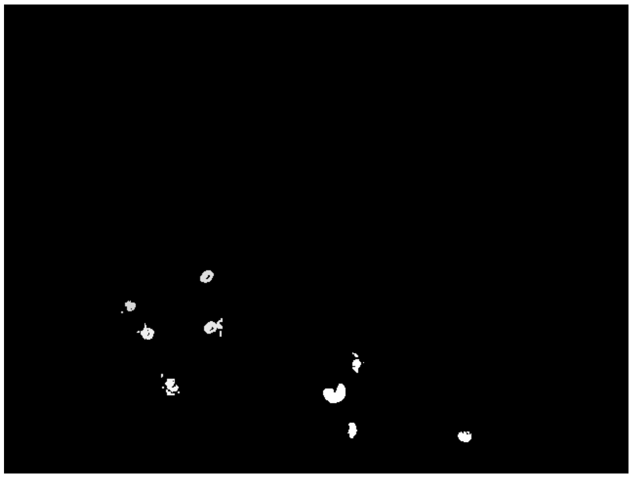
6 Experiments

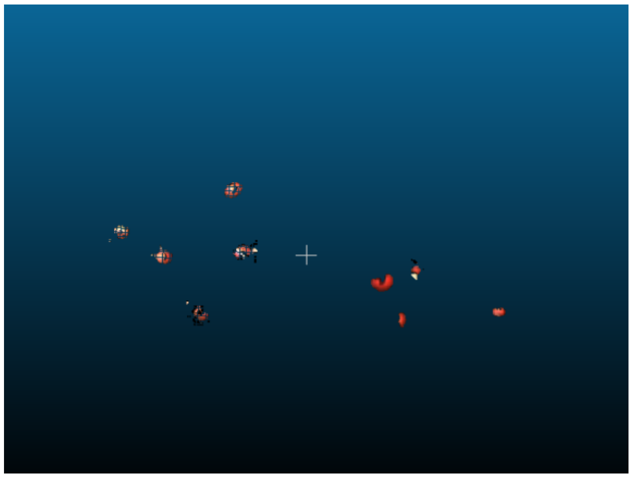
In order to validate the performance of the offline planning algorithm, we performed trials in three different apple orchards. We started by marking the GPS coordinates of the two ends of the traversable paths for each row. We then computed a sawtooth trajectory for the UAV using the offline view planning algorithm presented in Section 4. The algorithm takes as inputs the bounding GPS coordinates, the maximum height of the trees in a row, the camera field-of-view angles and distance of the traversable path from the apple trees. The output produced is the sequence of GPS coordinates that must be visited in order to have full coverage of the row. This sequence is given as input to the on-board autonomous waypoint navigation controller. Figure 12 shows the computed and actual trajectory for one trial. Figure 13 shows the errors between the actual trajectory and the input trajectory. The mean square errors of the Cartesian coordinates between the trajectories at each time step are $MSE_x = 26.82\ cm^2$, $MSE_y = 11.83\ cm^2$, $MSE_z = 16.82\ cm^2$ while the Euclidean distance error is $MSE_{ED} = 58.31\ cm^2$. The errors are in part due to the accuracy of the GPS receiver.

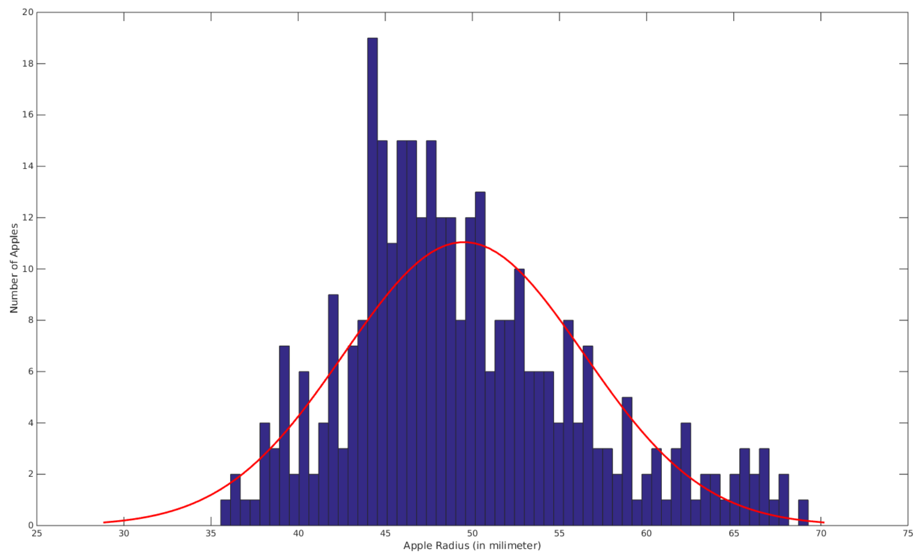
To verify the counting process, we segmented apple clusters from the captured images and applied our counting algorithm on these clusters. Table 1 shows some representative results. The counting algorithm is conservative and depends only on the total number of apple pixels. Consequently, in some cases the number of apples counted by this algorithm is less than the actual count. We estimated the diameter of apples for part of a row in an orchard. Figure 11e shows a histogram of the results. We analyze the diameter estimation process in the following section. The video accompanying this paper demonstrates results for the segmentation and registration procedure presented in Section 5.

(a) Left image

(b) Right image


(c) Disparity image


(d) Point cloud from stereo



(e) Histogram of apple radius (in mm) for an entire row

Figure 11: Disparity map and apple radius histogram.

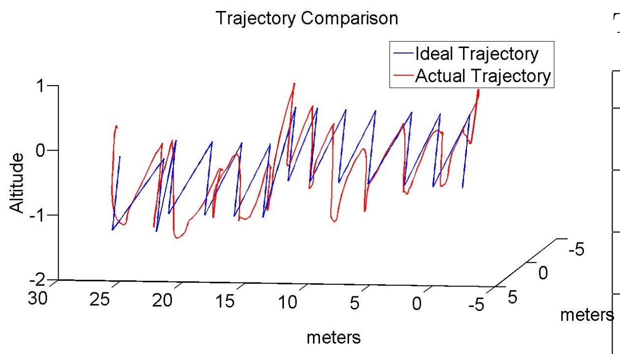


Figure 12: Real trajectory (Red) vs computed trajectory (Blue).

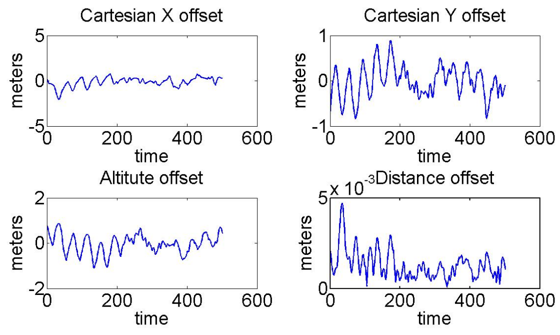
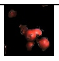
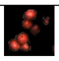
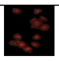


Figure 13: Errors between the computed trajectory and the actual trajectory.

Table 1: Counting Apples from Apple segments.

Apple Cluster	Prediction	Ground Truth
	1	1
	2	2
	2	2
	3	4
	4	6
	3	3
	7	7
	7	11

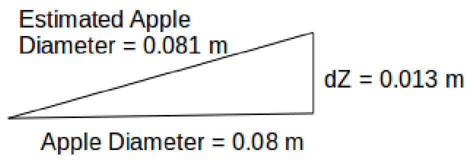


Figure 14: Finite resolution error calculation.

7 Error Analysis

In this section, we investigate the error in estimating apple diameter using the stereo system. Since diameter computation can be performed in the stereo camera frame, errors in UAV localization do not directly affect this computation.

We first investigate errors due to finite pixel resolution: even if we could find the pixel containing the contour precisely, we do not know the true location inside the pixel. This error propagates to diameter estimation as follows. Basic geometry yields the equation $Z = fB/d$ where f is the focal length, Z is the depth, B is the baseline, and d is disparity. A lower bound on the expected error can be obtained by a simple sensitivity analysis which yields that errors in disparity propagate to errors in depth by:

$$\partial Z = \frac{Z^2}{fB}\partial d \tag{2}$$

In our set up, the camera focal length is $f = 2.8 \times 10^{-3}m$, the baseline is $B = 0.01m$ and the pixel size is $3.75 \times 10^{-6}m$. Hence, when the camera is $1m$ away from the apple, the error in depth is $0.013m$. Then the depth error ∂Z causes the diameter estimation error as shown in Figure 14. For our test apple, which is $0.08m$ in diameter, this yields an error of 1.25 %.

In general, the error is expected to be more than a single pixel, since blurriness makes it hard to localize the contour. In the setup shown in Figure 16, at $1m$, the average error was $4.3 \times 10^{-4}m$, which is 0.61% error for a $0.08m$ apple. At $1.5m$ away, the error averaged out to be $1.63 \times 10^{-3}m$, which is 2.25% error for a $0.08m$ apple. The measurements are shown in Table 2.

8 Conclusion

In this paper, we showed that images obtained from a standard stereo camera pair mounted on

Table 2: Marker distance estimation error on A,B,and C apples

Apple ID	Camera distance from Apple [m]	Actual Distance [m]	Measured Distance [m]	Error [m]	Error [%]
A	1	0.066	0.068	0.002	3.03
B	1	0.066	0.0661	0.0001	0.15
C	1	0.064	0.0632	-0.0008	-1.25
A	1.5	0.066	0.069	0.003	4.55
B	1.5	0.066	0.0697	0.0037	5.61
C	1.5	0.064	0.0622	-0.0018	-2.81

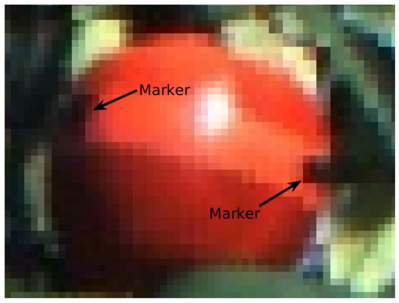


Figure 15: An apple with two black markers on left and right sides.

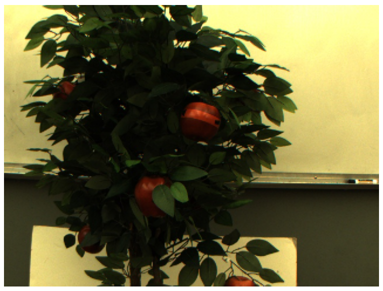


Figure 16: Experiment setup showing the markers.

a small aerial vehicle can be used for detecting, counting and measuring the diameters of apples in an orchard under natural conditions. Our main technical contribution is a new method to register apples across images in a robust fashion which lays the ground for estimating their diameter and number.

For our next step, we would like to collect ground truth data and evaluate during the harvest season to better understand the limits of the current system in terms of robustness, longevity and accuracy.

9 Acknowledgment

The authors would like to thank Professors James Luby and Cindy Tong from the University of Minnesota, Department of Horticultural Science for numerous helpful discussions and access to the UMN field station. This work is supported in part by MnDrive Robotics, Sensing and Advanced Manufacturing Initiative as well as NSF grants #1317788 and #1111638.

Appendix A Index to Multimedia Extension

Index to Multimedia Extension

Extension	Media Type	Description
1	Video	Yield estimation process in an apple orchard: `http://agricultural-robotics.cs.umn.edu/`

Appendix B Segmentation Algorithm

```
INPUT:
IMG: input image,
blocksize: blocksize for color averaging,
color: primary color of the apples to be detected
OUTPUT:
ISEG: segmented apple image,
boundaries: contour boundary for apple clusters
begin
Make a copy of IMG and assign to $I$.
for all pixels $I(x,y)$ do
  $[\hat{r},\hat{g},\hat{b}] \leftarrow$ average colors using blocksized neighborhood around $I(x,y)$
  if $\max\{\hat{r},\hat{g},\hat{b}\} \neq \hat{r}$ then
    set all the RGB components of $I(x,y)$ to zero
  else
    if $((\hat{r}-\hat{b}<25 \vee \hat{r}-\hat{g}<25) \wedge \hat{r}>200) \vee (\hat{r}-\hat{b}<10) \vee (\hat{r}-\hat{g}<10)) \wedge$ color = red then
      set all the RGB components of $I(x,y)$ to zero
    else if $(\hat{g}-\hat{b}>30) \vee (\hat{g}+\hat{b}>270) \vee (\hat{g}-\hat{b}<40 \wedge \hat{r}-\hat{g}<50 \wedge \hat{r}>79)$ then
      set all the RGB components of $I(x,y)$ to zero
    end if
  end if
end for
ISEG$\leftarrow I$
edge $\leftarrow$ CANNY($I$) {Canny Edge detection Canny (1986)}
boundaries$\leftarrow$FINDCONTOUR($edge$) {contour finding function from opencv Bradski & Kaehler (2008)}
end
```

Algorithm 2: Segmentation of Apple Pixels

Appendix C Speeding up Contour Matching with Known Motion and Small Motion Constraints

While matching contours, we made no assumption about the direction of motion. If however, the directional constraints are known (e.g from inertial measurement unit (imu) data) we can use additional global matching strategies. For example, we can use the Hu moments with motion constraints to find out the matches. This process takes $O(k_2)$ time where k is the number of contours and k is very small compared to the size of the image. The exact way of comparing the contours c_1, c_2 using this method is the following:

$$M(c_1,c_2) = \max_{i=1,\dots,7} \frac{|m_i^{c_1} - m_i^{c_2}|}{|m_i^{c_1}|} \tag{3}$$

$$m_i^{c_1} = sign(h_i^{c_1}) \log \dot{h}_i^{c_1}$$

$$m_i^{c_2} = sign(h_i^{c_2}) \log \dot{h}_i^{c_2}$$

where $h_i^{c_1}$ and $h_i^{c_2}$ are hue moments (Hu (1962)) of c_1 and c_2 respectively. We used the OpenCV (Bradski & Kaehler (2008)) implementation of the function.

If the motions are small enough that all the contours are overlapping in both images, we can just take a bounding rectangle based approach. For each contour we will find a bounding rectangle and find the intersecting rectangles from the previous image. We will pick the contour whose bounding rectangle produced the biggest intersection.

Appendix D Using 3D Alignment for Registration

The onboard stereo pair provides depth measurements for each frame. Using the correspondences between successive stereo pairs, we set up a system of linear equations to solve for the rotation and translation between the 3D point clouds and register them. This procedure involves matching between both left-left and left-right images which results in accumulation of error with increasing number of frames. As a result, we find multiple views of a single cluster are being registered as separate apple clusters. This technique also suffers from the fact that the number of left-left matches are significantly more compared to left-right matches causing many points to have non-decipherable depth. Bundle adjustment techniques (Agarwal et al. (2010)) which are used to refine this reconstruction, require robust matches which are hard to obtain using conventional feature detectors (Figure 7). Figure 17 shows the comparison between 3D alignment and our registration method.

Appendix E Using Structure from Motion for Registration

In this section we show the performance of registration using structure from motion techniques. We used a state of the art tool "Visual SFM " from Wu (2011). This tool has the capability to produce sparse and dense reconstruction given a set of captured images (with/without a calibrated camera) and also incorporates bundle adjustment. This tool did not work very well for the apple images. The SIFT (Lowe (1999)) feature detector it uses struggles to find the correct correspondences (Figure 18a). Consequently, despite bundle adjustment the sparse reconstruction had very few points and the calculated camera trajectory was incorrect. Figure 18b shows the reconstruction from Visual SFM for ten frames. These are the same frames used by our algorithm and 3D alignment. The 3D reconstruction using those two method can be found in Figure 17.

References

Agarwal, S., Snavely, N., Seitz, S. M. & Szeliski, R. (2010), Bundle adjustment in the large, *in* 'Computer Vision–ECCV 2010', Springer, pp. 29–42.

Ardupilot APM:Copter (2015 (accessed Jul 2015)).
URL: *http://copter.ardupilot.com*

Baker, S. & Matthews, I. (2004), 'Lucas-kanade 20 years on: A unifying framework', *International journal of computer vision* **56**(3), 221–255.

Ball, D., Ross, P., English, A., Patten, T., Upcroft, B., Fitch, R., Sukkarieh, S., Wyeth, G. & Corke, P. (2013), 'Robotics for sustainable broad-acre agriculture', *Proceedings of Field and Service Robotics (FSR 2013)* pp. 1–14.

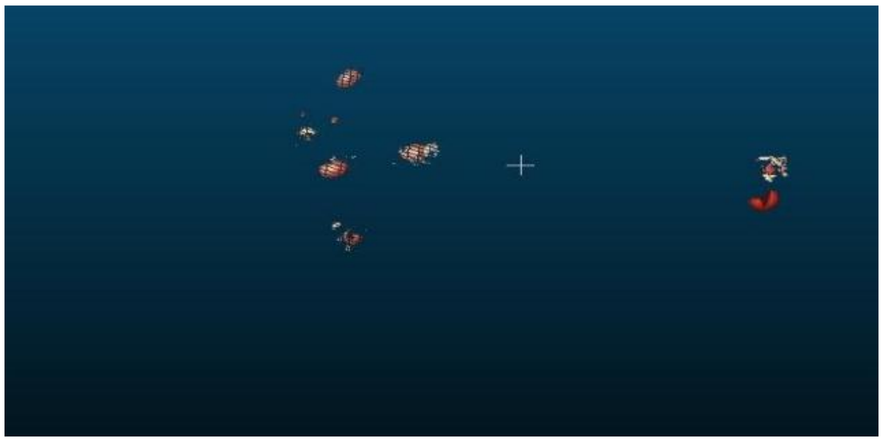
Bay, H., Tuytelaars, T. & Van Gool, L. (2006), Surf: Speeded up robust features, *in* 'Computer vision–ECCV 2006', Springer, pp. 404–417.

Bergerman, M., Singh, S. & Hamner, B. (2012), Results with autonomous vehicles operating in specialty crops, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1829–1835.

Bergerman, M., Van Henten, E., Billingsley, J., Reid, J. & Mingcong, D. (2013), 'Ieee robotics and automation society technical committee on agricultural robotics and automation', *IEEE Robotics and Automation Magazine* **20**(2), 20–23.

Bradski, G. & Kaehler, A. (2008), *Learning OpenCV: Computer vision with the OpenCV library*, " O'Reilly Media, Inc.".

Briechle, K. & Hanebeck, U. D. (2001), Template matching using fast normalized cross correlation, *in* 'Aerospace/Defense Sensing, Simulation, and Controls', International Society for Optics and Photonics, pp. 95–102.


(a) Reconstruction from ten frames using our contour based registration

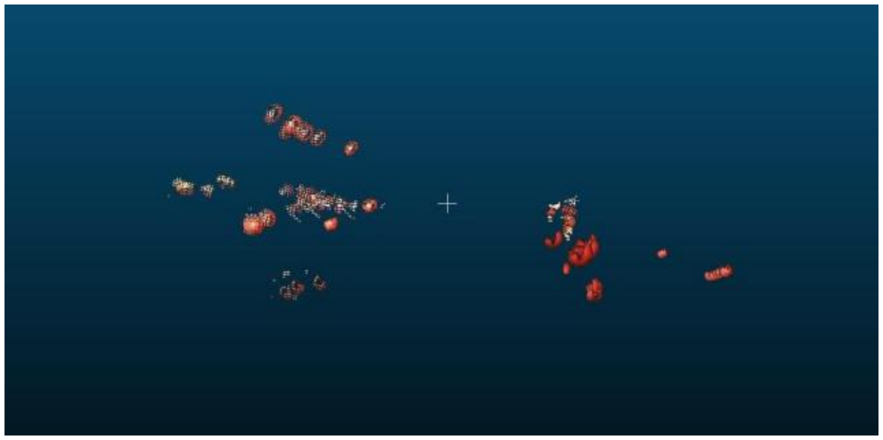
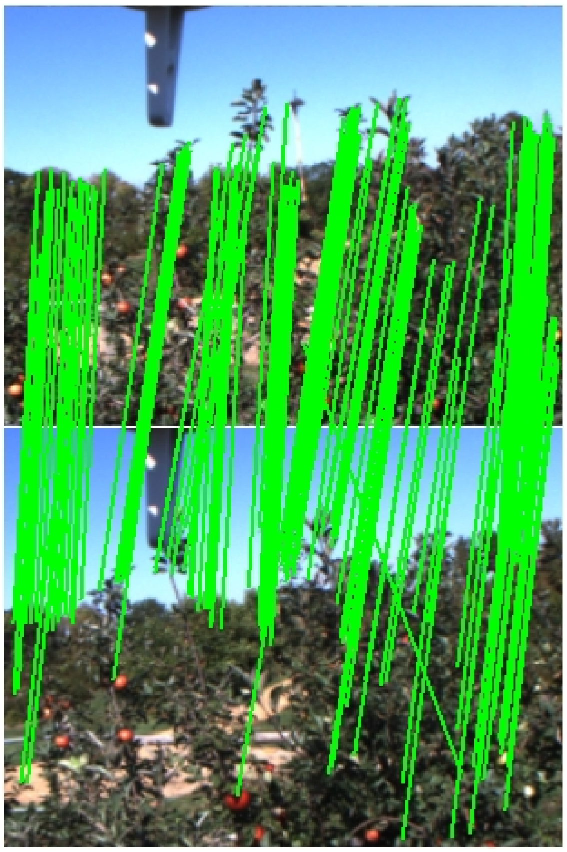

(b) Reconstruction from ten frames using 3D alignment

Figure 17: Comparison of our registration technique (which performs image based registration) vs 3D alignment based registration. The clutter in the 3D registration shown below is due to the projection of alignment error. This results in erroneous counting of the apples.


(a) Matching by Visual SFM

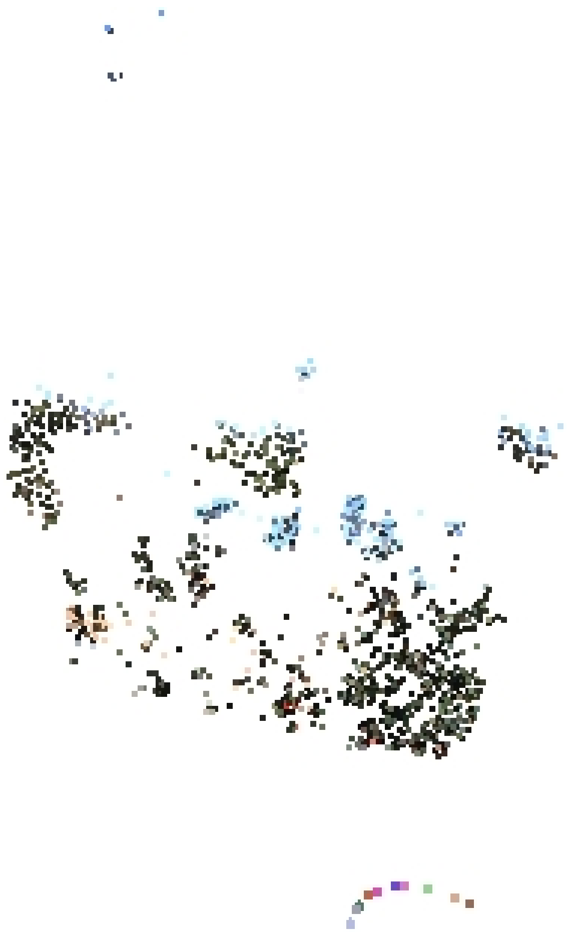

(b) Reconstruction from ten frames using Visual SFM

Figure 18: Visual SFM Performance. Visual SFM does not work well on the captured images. The matching tool struggles to find correct matches (Figure a). Following the matches it has a very sparse reconstruction that does not contain any of the apples (Figure b).

Canny, J. (1986), 'A computational approach to edge detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* .

Das, J., Cross, G., Qu, C., Makineni, A., Tokekar, P., Mulgaonkar, Y. & Kumar, V. (2015), Devices, systems, and methods for automated monitoring enabling precision agriculture, *in* 'Proceedings of IEEE Conference on Automation Science and Engineering'. to appear.

Dillow, C. (2014 (accessed Jul 2015)), 'Despite FAA dithering, a drone economy sprouts on the farm'.
URL: *http://fortune.com/2014/09/16/despite-faa-dithering-a-drone-economy-sprouts-on-the-farm/*

DIYDrones (2015 (accessed Jul 2015)).
URL: *http://diydrones.com*

DJI F550 (2015 (accessed Jul 2015)).
URL: *http://www.dji.com*

Dobbs, T. (2014 (accessed Jul 2015)), 'Farms of the future will run on robots and drones'.
URL: *http://www.pbs.org/wgbh/nova/next/tech/farming-with-robotics-automation-and-sensors/*

Eggert, D. W., Lorusso, A. & Fisher, R. B. (1997), 'Estimating 3-d rigid body transformations: a comparison of four major algorithms', *Machine Vision and Applications* **9**(5-6), 272–290.

Hodge, V. J. & Austin, J. (2004), 'A survey of outlier detection methodologies', *Artificial Intelligence Review* **22**(2), 85–126.

Hu, M.-K. (1962), 'Visual pattern recognition by moment invariants', *Information Theory, IRE Transactions on* **8**(2), 179–187.

Huang, T. S. & Netravali, A. N. (1994), 'Motion and structure from feature correspondences: A review', *Proceedings of the IEEE* **82**(2), 252–268.

Hung, C., Bryson, M. & Sukkarieh, S. (2012), 'Multi-class predictive template for tree crown detection', *ISPRS Journal of Photogrammetry and Remote Sensing* **68**, 170–183.

Jagbrant, G., Underwood, J. P., Nieto, J. & Sukkarieh, S. (2015), Lidar based tree and platform localisation in almond orchards, *in* 'Field and Service Robotics', Springer, pp. 469–483.

Jimenez, A., Ceres, R. & Pons, J. (2000), 'A survey of computer vision methods for locating fruit on trees', *Transactions of the ASAE-American Society of Agricultural Engineers* **43**(6), 1911–1920.

Linker, R., Cohen, O. & Naor, A. (2012), 'Determination of the number of green apples in rgb images recorded in orchards', *Comput. Electron. Agric.* **81**, 45–57.

Lowe, D. G. (1999), Object recognition from local scale-invariant features, *in* 'Computer vision, 1999. The proceedings of the seventh IEEE international conference on', Vol. 2, Ieee, pp. 1150–1157.

Mao, W., Jia, B., Zhang, X. & Hub, X. (2009), Detection and position method of apple tree image, *in* D. Li & C. Zhao, eds, 'Computer and Computing Technologies in Agriculture II, Volume 2', Vol. 294 of *IFIP Advances in Information and Communication Technology*, Springer US, pp. 1039–1048.

MAVROS (2015 (accessed Jul 2015)).
URL: *http://wiki.ros.org/mavros*

Mulla, D. J. (2013), 'Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps', *Biosystems Engineering* **114**(4), 358–371.

ODROID-U3 (2015 (accessed Jul 2015)).
URL: *http://hardkernel.com*

Pixhawk (2015 (accessed Jul 2015)).
URL: *http://www.pixhawk.org*

Point Grey Chameleon USB 2.0 Cameras (2015 (accessed Jul 2015)).
URL: *http://ww2.ptgrey.com/USB2/chameleon*

Tokekar, P., Vander Hook, J., Mulla, D. & Isler, V. (2013), Sensor planning for a symbiotic uav and ugv system for precision agriculture, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 5321–5326.

Wang, Q., Nuske, S., Bergerman, M. & Singh, S. (2013), Automated crop yield estimation for apple orchards, *in* J. P. Desai, G. Dudek, O. Khatib & V. Kumar, eds, 'Experimental Robotics', Vol. 88 of *Springer Tracts in Advanced Robotics*, Springer International Publishing, pp. 745–758.

Won, D.-Y., Göktoğan, A. H., Sukkarieh, S. & Tahk, M.-J. (2013), View planning of a multirotor unmanned air vehicle for tree modeling using silhouette-based shape estimation, *in* 'Frontiers of Intelligent Autonomous Systems', Springer, pp. 193–207.

Wu, C. (2011), 'Visualsfm: A visual structure from motion system', *VisualSFM: A Visual Structure from Motion System* .