A robot system for pruning grape vines

Tom Botterill[1], Scott Paulin[1,2], Richard Green[1], Samuel Williams[1], Jessica Lin[1], Valerie Saxton[3], Steven Mills[4], XiaoQi Chen[2] and Sam Corbett-Davies[5].

[1]Department of Computer Science
University of Canterbury, Christchurch, NZ
tom@hilandtom.com

[2]Department of Mechatronics Engineering
University of Canterbury, Christchurch, NZ

[3]Faculty of Agriculture and Life Sciences
Lincoln University, Lincoln, NZ

[4]Department of Computer Science
University of Otago, Dunedin, NZ

[5]Computational Vision and Geometry Lab.
Stanford University, Stanford, CA.

This is the pre-peer reviewed version of the following article: "A robot system for pruning grape vines", which is to be published in full in the Journal of Field Robotics (http://www.journalfieldrobotics.org/).

Abstract

This paper describes a robot system for automatically pruning grape vines. A mobile platform straddles the row of vines, and images them with trinocular stereo cameras as it moves. A computer vision system builds a 3D model of the vines, an AI system decides which canes to prune, and a six degree-of-freedom robot arm makes the required cuts. The system is demonstrated cutting vines in the vineyard.

This papers main contributions are the computer vision system that builds 3D vine models, and the test of the complete integrated system. The vine models capture the structure of the plants so that the AI can decide where to prune, and are accurate enough that the robot arm can reach the required cuts. Vine models are reconstructed by matching features between images, triangulating feature matches to give a 3D model, then optimising the model and the robot's trajectory jointly (incremental bundle adjustment). Trajectories are estimated online at 0.25m/s, and have errors below 1% when modelling a 96m row of 59 vines.

Pruning each vine requires the robot arm to cut an average of 8.4 canes. A collision-free trajectory for the arm is planned in 1.5s/vine with a Rapidly-exploring Random Tree motion planner. Total time to prune one vine is two minutes in field trials, which is similar to human pruners, and could be greatly reduced with a faster arm. Trials also show that the long chain of interdependent components limits reliability. A commercially-feasible pruning robot should stop and prune each vine in turn.

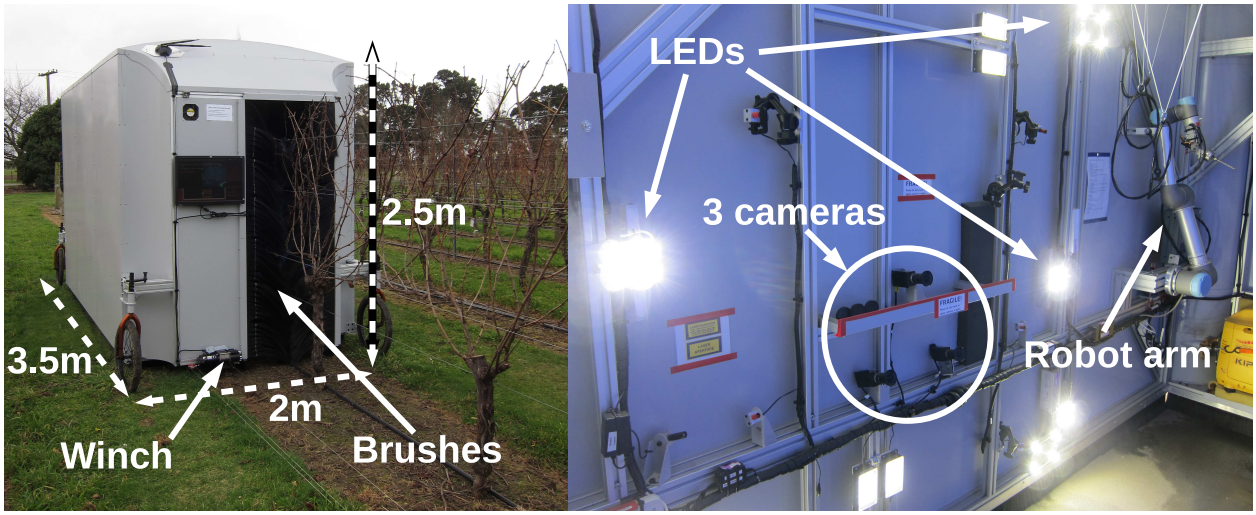


Figure 1: The mobile platform completely covers the vines, blocking out sunlight (left). Inside the platform are high-powered LEDs, three cameras, and the robot arm (right). Also mounted inside the platform are a generator and the desktop PC that runs all of the software.

1 Introduction

Grapes are an internationally important crop, and vineyards cover 75000km^2 worldwide (FAO, 2013). Grape vines are pruned annually to increase yield and prevent disease (Kilby, 1999). Many vine species are pruned by hand, and this is often the most labour-intensive and expensive task in the vineyard (Dryden, 2014). This paper describes a prototype system for automating the common hand-pruning technique known as *cane pruning*.

Vines are pruned in winter, when dormant and free of leaves. Each vine (individual plant) is pruned by cutting off some of its canes (branches), while leaving others. To prune a vine using the *cane pruning* technique, most of the year's new canes and all older canes are cut off, leaving a selection of long healthy canes for the following year (Christensen, 2000). Typically between four and twelve cuts are made per plant, and two to six canes are kept. Automating cane pruning is challenging because a complete and connected model of the structure of the vine is required in order to make cut decisions. Stereo algorithms struggle to model the tangled network of overlapping canes, while other 3D imaging sensors often lack the required resolution, or cannot operate while moving. A major contribution of this paper is the computer vision system for reconstructing sufficiently complete and connected models of vines from 2D images.

The vine pruning robot is mounted on a mobile platform which straddles the row of vines (Figure 1). The vines are imaged by three colour cameras in a trinocular stereo rig, and are cut with a spinning mill-end cutter attached to a six degree-of-freedom arm. The computer vision system follows a standard feature-matching and bundle adjustment pipeline (Hartley and Zisserman, 2003; Triggs et al., 1999), with each stage of the pipeline customised for vines. The first stage of the reconstruction pipeline is to extract the 2D positions of canes from each image. The second stage is to find a correspondence between the individual 2D canes viewed in different images, and to reconstruct them in 3D. The final stage is to optimise the 3D model, and the robot's trajectory, in an incremental bundle adjustment framework. As the robot moves, more canes come into view and the 3D model is extended.

Once a complete 3D model of a vine is available, an AI algorithm decides which canes to prune. The 3D model and cutpoints are computed online while the robot is moving. The platform then stops while the robot arm makes the cuts. A collision-free trajectory for the arm is computed using a Rapidly-exploring Random Tree path planner (Kuffner and LaValle, 2000). The arm then reaches to each cutpoint to make

the required cuts.

Our prototype system is pulled along the row of vines with a winch, but a commercial system would be either self-powered or towed by a tractor. As with hand-pruned vines, cut canes stay attached to the trellis, and are removed later by a mechanical 'stripper' machine (Dryden, 2014).

The ultimate aim of our research is to develop a commercially-viable pruning robot that operates while moving continuously along the row of vines. This paper describes a prototype system that demonstrates many of the key components that are necessary. The two principle novel contributions of this paper are first, to describe and evaluate the computer vision system for incrementally reconstructing vines from 2D images; and second, to demonstrate a complete integrated system that uses the computer vision system to reconstruct vines. This prototype system is a first step towards developing a commercially-viable vine pruning robot.

This paper is organised as follows: Section 2 describes related agricultural robot systems, and methods for imaging the structure of plants. Section 3 describes the entire system. Individual components of the system have been described in several short papers over the last four years, and this section summarises these earlier papers. This paper is the first time a description and evaluation of the entire computer vision system has been presented, and it provides the first description of the entire robot system. Section 4 describes how the system was parametrised to perform well for vines. Section 5 evaluates the computer vision software on large datasets showing hundreds of vines, and describes the first tests of the pruning robot in the vineyard. Finally, Sections 6 and 7 discuss our findings, and make recommendations for how the knowledge gained in this research can be applied to future pruning robots.

2 Background

This section first describes related systems using robot arms in agriculture, then summarises the state-of-the-art for capturing 3D models of plants.

2.1 Robot arms in agriculture

Robot arms are widely used in controlled factory environments to automate tasks once done manually (Brogårdh, 2007). Agriculture also has many labour-intensive tasks, e.g. harvesting, pruning and planting (Zhang and Pierce, 2013), however automation of these tasks has so far had limited success. Challenges for building automated systems include natural variability of crops and challenging lighting conditions (Bac et al., 2014; Li et al., 2011; Kapach et al., 2012).

One target for automation is fruit and vegetable harvesting. Bac et al. (2014) review fifty recent fruit picking robot systems. These systems use cameras to identify fruit, then reach and pick with robot arms. Historically, a limitation of fruit picking robots was their low accuracy at spotting fruit amongst foliage (De-An et al., 2011; Li et al., 2011), however modern computer vision techniques, which combine different sensors and/or different visual cues, and which are robust to lighting variation, are overcoming this limitation (Bac et al., 2014; Kapach et al., 2012; Edan et al., 2000). Removing fruit without damage remains a major challenge, and picking speeds are low compared with humans.

A robot system has also been developed for automating an alternative pruning method known as *spur pruning* (Vision Robotics Corporation, 2015). This commercial prototype system uses stereo cameras to identify canes, which robot arms then cut. Spur pruning requires different canes to be cut at different heights, however knowledge of the structure of the plant is not necessarily required.

For these applications, the targets (fruit or cutpoints) are generally easy to reach, so simple point-to-point

paths are sufficient, and collision avoidance is unnecessary. Cameras are often mounted on the arm's end effector, to provide the feedback that ensures the arm reaches the correct point. An exception is the melon harvesting robot developed by Edan et al. (2000). Melons are picked while the platform is moving, so the robot arm's trajectory must be planned and updated dynamically.

Many of these systems control lighting to reduce natural variation. The Vision Robotics vine pruner uses a canopy to shut out sunlight; Edan et al.'s melon harvester performs best at night, when artificial light is used; and the grape counting system by Nuske et al. (2014) and the apple counting system by Wang et al. (2013) operate only at night, with active illumination.

These previous systems all identify specific targets (cutpoints, fruit, etc.), without requiring a complete 3D model of the scene. By contrast, our pruning robot needs to model the structure of the entire plant in order to decide where to cut, and to plan collision-free paths for the robot arm. Each of these previous systems customise and parametrise computer vision algorithms for their particular application, and control lighting where possible. As with these approaches, our 3D reconstruction system is customised for the application, and our platform enables lighting to be controlled.

2.2 Computer vision for plant modelling

3D models of the structure of plants are used for phenomics research, plant breeding, crop monitoring and computer graphics, and systems with many different sensor and algorithm combinations are used for these applications. Sensors include structured light or time-of-flight depth cameras, and regular 2D colour or greyscale cameras. Systems using 2D cameras apply a range of photometric reconstruction algorithms, including feature-based methods, shape-from-silhouette methods (using voxels), and dense stereo algorithms.

Feature-based methods work by extracting a set of feature locations from each image, matching features showing the same object between images (establishing a *correspondence* between features), then triangulating to reconstruct objects in 3D. Point features with appearance described by SIFT (Scale Invariant Feature Transform; Lowe, 2004), or similar descriptors, are frequently used. Features are matched across multiple views, and the 3D model and camera positions are optimised jointly to give an accurate 3D model. This sparse non-linear optimisation is known as *bundle adjustment* (Triggs et al., 1999).

Point features work well for many reconstruction applications, but perform poorly in scenes where appropriate features are not present, or where many features appear similar and hence matches are ambiguous (Hofer et al., 2014). Plant images contain many occlusion boundaries around thin structures (branches), where detected corners or regions in images don't correspond to 3D points in the world. Plants also have many similar-looking parts, so matching features by appearance is challenging.

A complementary 3D reconstruction method is dense stereo. Dense stereo algorithms use a pair of images from cameras with known relative pose to find a disparity map mapping each pixel in one image to its matching pixel in the other. The depth of the scene at each pixel is computed from its disparity, giving a dense 3D model of the object's surface. Disparity maps are found by minimising an objective function that measures both the similarity in appearance between matching pixels, and how well the disparity map matches assumptions about the structure of the world. The reconstruction is often assumed piecewise planar or piecewise continuous, and depth discontinuities and occluded pixels are penalised (Boykov et al., 2001; Woodford et al., 2009; Gimel'farb, 2002). These assumptions do not hold for images of branching plants, which have many occlusions and depth discontinuities. Dense stereo algorithms are frequently used to generate 3D surface models following feature-based reconstruction of camera poses and a sparse set of points (Tingdahl and Van Gool, 2011).

Dey et al. (2012) and Ni and Burks (2013) use state-of-the-art dense stereo methods to reconstruct vines and citrus trees respectively, for measuring the canopy area and for counting fruit. The dense reconstructions are initialised with sparse reconstructions from feature matching. The detailed structure of the plants are

not required for these applications, and in both cases, many branches are missing from the reconstructions. Dey et al. require several minutes per reconstruction.

An alternative feature-free method for 3D reconstruction is shape-from-silhouette (Laurentini, 1994; Szeliski, 2010; Franco and Boyer, 2009). These methods first fill 3D space with a dense 3D array of cubic 'voxels' (or a similar volumetric representation of space, e.g. a volume enclosed by a mesh; Franco and Boyer, 2009), then remove all voxels (or regions of the mesh) that are inconsistent with any images, either because they project to background pixels, or are not 'photoconsistent' (consistent in appearance) between views. Shape-from-silhouette works well for imaging fine branching structures which can be segmented easily from the background, e.g. plants imaged against a coloured background. Kumar et al. (2012) and Paproki et al. (2011) use shape-from-silhouette to model wheat and cotton plants respectively, and observe that very accurately-calibrated cameras (or very accurate relative camera poses for a moving camera) are required so that fine structures are not lost. Zheng et al. (2011) model plant roots growing in a gel, and find that an accurate foreground/background segmentation is essential to give a complete reconstruction. Tabb (2013) use shape-from-silhouette to model plants, and propose a soft-labelling of voxels in order to provide increased robustness to calibration errors. All of these approaches require a large number of voxels in order to model fine structure (cubic in the model resolution), which makes efficient implementation challenging.

3D data can be captured directly using depth cameras. Historically, high resolution depth cameras required stationary scenes, because both coded light and time-of-flight systems work by capturing multiple views in rapid succession (Geng, 2011). Cameras that can image moving scenes (e.g. Microsoft's Kinect) often have insufficient depth map resolution for detailed structures like vines and wires. Recent advances in technology are increasing the resolution available, and these cameras are now a viable alternative to regular cameras, however fitting a 3D vine structure to a 3D point cloud presents similar challenges to fitting a vine model to images, including occlusions, point cloud segmentation, and structural ambiguities. Another advantage of 2D cameras is the wider range of algorithms available when designing complex application-specific reconstruction tasks (Davis et al., 2013).

Nock et al. (2013) used a Kinect-like structured light camera for measuring simple branching plants. They fit a mesh to the point cloud, however the mesh interpolation leads to gaps in branches and incorrect joins between branches. Chéné et al. (2012) use a Kinect to image crops, but again there are gaps in the reconstruction around fine structures. They recommend imaging at night to prevent interference from sunlight, and recommend using knowledge of the plant structure to guide a more accurate reconstruction. Several authors have used 3D cameras to image trees with automated pruning as the planned application: Nir and Linker (2014) image apple tree branches in a laboratory setting with a laser line structured light scanner mounted to a robot arm. The relatively high resolution of scanning structured light enables over 90% of branches to be identified, although 39% of detected branches are 'fragmented' into different parts. Elfiky et al. (2015) image apple tree branches with a Kinect 2 camera. They fuse multiple images into a single point cloud, identify the trunk, then identify the points at which branches leave the trunk in the point cloud.

Skeletonisation is the process of recovering a model from images of objects with a network structure, such as branching plants. Liu et al. (2012) and Karkee et al. (2014) each propose offline systems for modelling trellised apple trees for deciding where to prune. The trees are structurally simple, with one trunk and few overlapping branches. Liu et al. use a combination of a high resolution camera, active illumination (so only nearby branches are imaged) and a Kinect, with a skeleton from the high resolution image used to fill gaps in the 3D Kinect image. Karkee et al. fuse multiple images from a time-of-flight camera, skeletonise the resulting volumetric model, then select branches which should be pruned based on their length and spacing. 77% of branches are detected.

In summary, reconstructing the structure of branching plants is challenging because of their thin and self-occluding structures. Fine structures can be reconstructed using high-resolution voxel-based shape-from-silhouette, however this is inefficient for large plants, and requires very accurate relative camera poses, which are challenging to estimate for moving cameras.

A good solution for reconstructing scenes where point-based features perform poorly is to use straight lines as features instead. Hofer et al. (2014) reconstruct power pylons and buildings using a line-based bundle adjustment framework. Micusik and Wildenauer (2014) reconstruct indoor scenes using line segments. Geometrical constraints on line feature matches (from the epipolar geometry) are weaker than for point features, because lines in images often end at occlusion boundaries. This makes unambiguous matching challenging. Despite this, line features allow more complete reconstructions in environments where they better represent the structure of the objects being modelled.

Historically, a challenge of incrementally reconstructing scenes online has been the growing cost of bundle adjustment as more observations are made. Recently, this challenge has largely been overcome by optimisers including g2o (a general (hyper)graph optimiser; Kummerle et al., 2011) and iSAM2 (incremental Smoothing And Mapping; Kaess et al., 2012). These optimisers were designed for the closely-related problem of Simultaneous Localisation and Mapping, where a mobile robot uses its sensors to jointly estimate its pose and a map of its environment. As the robot moves, newly-observed features are added to the optimisation. Once feature positions are accurately estimated, they are removed from the optimisation, hence maintaining constant time complexity. Incremental bundle adjustment systems is ideal for our pruning robot, where plant models are built up online from many views. Linear features are more suitable than point features for representing thin structures like canes. Our computer vision system combines these ideas in order to reconstruct entire rows of vines online.

3 System design

This section describes each component of the pruning robot in turn: firstly the imaging platform, which creates optimal lighting conditions for imaging the vines; secondly the computer vision system, which builds a parametric 3D model of the vines; thirdly the AI system, which uses the vine model to decide which canes to cut; and finally the path planner, which computes a collision free trajectory taking the robot arm to the correct cutpoints.

3.1 Robot platform

The robot platform provides a controlled environment for imaging the vines. The platform straddles a row of vines, and brushes at each end block out sunlight (Figure 1). The cameras are mounted on one side, and the other side is coloured blue to aid the segmentation of vines from the background (Botterill et al. (2015) describe the canopy design and camera setup in detail). A Universal Robots UR5 six-jointed robot arm is mounted 1.6m behind the cameras, so that it can prune the vines after the entire two-metre wide plant has been modelled.

Three 1.3 megapixel Point Grey Grasshopper2 colour cameras are arranged in a triangular configuration (Figure 1). 3D reconstruction of linear features (such as vines) is ill-conditioned when they are aligned with the stereo baseline, but with three cameras we can select the pair for which each reconstruction is well-conditioned. The cameras are calibrated using standard methods (as provided in the OpenCV Computer Vision Library, n.d.), and the robot-to-camera transform is computed by attaching a calibration target to the robot arm.

To illuminate the vines, we use 14 sets of high-power LEDs (each set contains four 900 lumen LEDs). Providing even illumination across the scene is challenging due to the high depth of field of the vines (light intensity decreases quadratically with increasing distance from light sources), large field-of-view of the cameras (light sources' luminance varies with angle) and because of shadows. We built a computer model of the lights within the frame, and calculated lighting levels throughout the region where vines are imaged. We then optimised the positions of light sources to minimise lighting variation. Illumination levels vary by 48% for the optimal configuration of 14 light sources, whereas simpler configurations (a regular grid) give

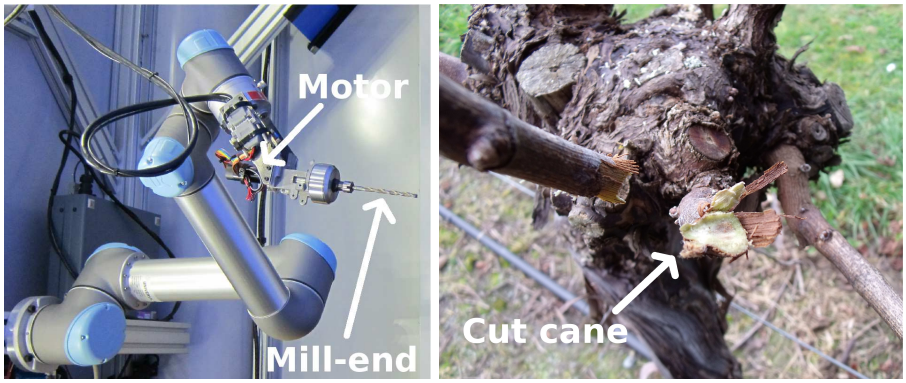


Figure 2: The cutting tool consists of a router mill-end attached to a high-speed motor (left). A vine with a cut cane is shown on the right (best viewed on-screen). Cuts are often close to other canes or the head. Canes to cut are typically 6-15mm in diameter.

70% variation, and a single point source gives 96% variation (Botterill et al., 2012b).

To cut canes, we use a 6mm CNC router mill-end attached to a 100W 24000 revolution-per-minute Maxon brushless DC motor (Figure 2). To cut a cane the robot arm sweeps the cutter through the cane (a *cut motion*). As the cut motion requires considerable collision-free space, and the cutter sometimes fails to make separation cuts, we will replace this with a secateur-based cutter in future (Section 6).

3.2 Computer vision system

The computer vision system takes the images from the trinocular stereo cameras and constructs a high-level 3D model of the vine and trellis's structure. We refer to each individual branch as a *cane*, and the entire plant (composed of many connected canes) as the *vine*. Canes are connected in an acyclic tree structure, with each cane growing (*forking*) from a larger parent cane, or the vine's trunk, and terminating in a *tip*, a point not connected to any other cane. We describe a vine model as *complete and connected* if all of the canes in the vine are modelled, and are connected together within the model. A complete and connected model of a vine is necessary for reasoning about where cuts should be made and which canes should be removed in order to prune the vine. Canes (and other components) in 2D and 3D are modelled by *polylines*; a polyline is the connected sequence of straight line segments defined by a sequence of control points. Attributes including cane thickness and the connections between canes are also estimated. These parametric models are essential for making decisions about which canes to prune, and are useful for efficient computation of collision-free paths for the robot arm.

The computer vision system is based on a standard feature matching and incremental bundle adjustment pipeline, but each stage of the pipeline is customised to work well for vines. Figure 3 gives an overview of the system, and the following sections describe each module in turn. Our approach contrasts with systems using an off-the-shelf general purpose 3D reconstruction (Dey et al., 2012; Nock et al., 2013), where 3D surfaces or point clouds are computed and plant structures are fitted to this 3D data.

The computer vision system uses Support Vector Machines (SVMs), which are a standard machine learning method for classification, decision making, and estimating probabilities. SVMs use labelled training data to learn a model, then the model is used to classify new examples. Full details are given in Appendix A.1.

For each stereo triple of frames:

1. For each image:
   - Segment foreground/background.
   - Detect 2D posts, trunks, wires, vines (2D polylines with attributes).
2. Initialise new camera pose (from correlation between images).
3. Assign 2D structure to existing 3D model.
4. Optimise 3D structure and camera poses.
5. Estimate 3D head model (shape-from-silhouette).
6. Resolve connections between canes.
7. Find a correspondence between unassigned 2D posts, trunks, wires, vines
   $\rightarrow$ new 3D structure (3D polylines with attributes).
8. Optimise 3D structure and camera poses.

Output: Connected parametric 3D model of vines.

Figure 3: Overview of 3D vine reconstruction pipeline.

3.2.1 Foreground/background segmentation

The first stage of the processing pipeline is to segment the vines and wires from the blue background. Even though lighting is controlled, segmentation is challenging due to lighting variation, shadows, and demosaicing artefacts[1] around thin structures like wires. We use a Radial Basis Function SVM (RBF-SVM) to label each pixel as foreground (vines, posts, etc.), background, or wire, based on its colour and the colour of its neighbourhood. The SVM was trained on features from hand-labelled images. The features are 5×5 patches from the Bayer image, with opposite pixels averaged for rotation invariance. For efficiency, we classify most pixels (95.5%) with simple linear SVMs, and only use the computationally more expensive RBF-SVM for the remainder.

3.2.2 2D structure detection

The next step is to find the structure of the trellis and vines in each foreground/background segmented image. The posts and wires in the trellis have simple structures so can be detected with standard computer vision methods. Wires are detected by fitting straight line models to neighbouring wire pixels, then merging collinear parts. Posts are detected by applying large filters to detect vertical edges, then applying a Hough transform (Duda and Hart, 1972).

Next, we find the structure of the vines in each image, i.e. the position of each cane and the connections between canes. Each image shows a tangled network of overlapping canes, so to recover the individual canes and the connections between them we use knowledge about the vines' structure. We know that each cane is a smooth curve of approximately uniform thickness, and that the canes are connected in a tree. We use a *Stochastic Image Grammar* (Zhu and Mumford, 2007) to represent this knowledge. Botterill et al. (2013b) describes our Stochastic Image Grammar-based vine finder in detail.

Figure 4 shows a parse tree describing how vines generate an image. To find the vine's structure, we first extract cane edge segments from each image, by taking the outline of the foreground segment and

[1] Digital cameras capture colour images with a colour filter array (a Bayer filter) so that each pixel detects either red, green, or blue. To give a full colour image, a demosaicing algorithm interpolates values for the other colour channels (Li et al., 2008).

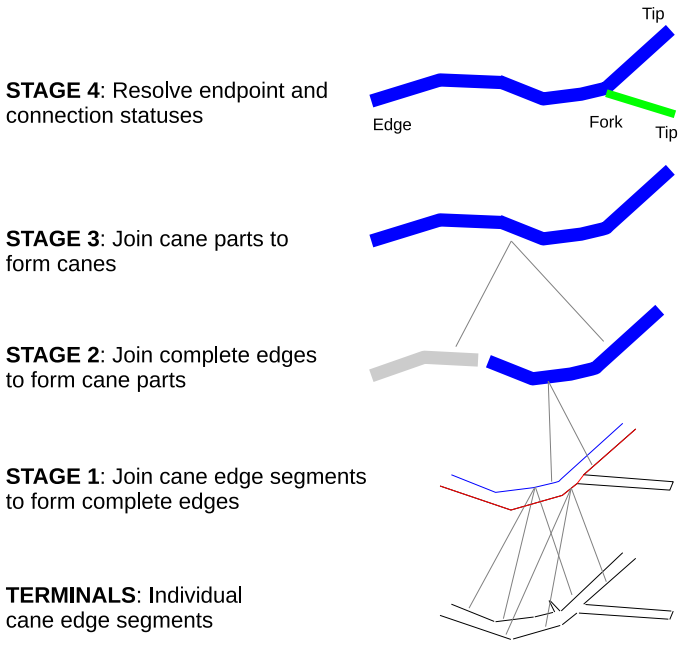


Figure 4: Our vine model explains how a vine creates an image, and enables the image to be parsed to recover the underlying vine structure.

approximating it as a set of straight line segments using the Ramer-Douglas-Peucker algorithm (Ramer, 1972; OpenCV Computer Vision Library, n.d.). These straight line segments are our terminal nodes. We bottom-up parse the cane edge segments by repeatedly joining nodes to form cane edges, then canes, then a connected cane network with labelled endpoints. Join decisions are made by SVMs, with feature vectors including lengths, thicknesses, curvature, and distances and angles between pairs of parts.

The 'head' of a vine is the irregular-shaped lump of old wood at the top of the trunk. We detect the head region after the canes have been detected: we find the foreground region not explained by any other structure and apply morphological opening (Szeliski, 2010) to remove noise.

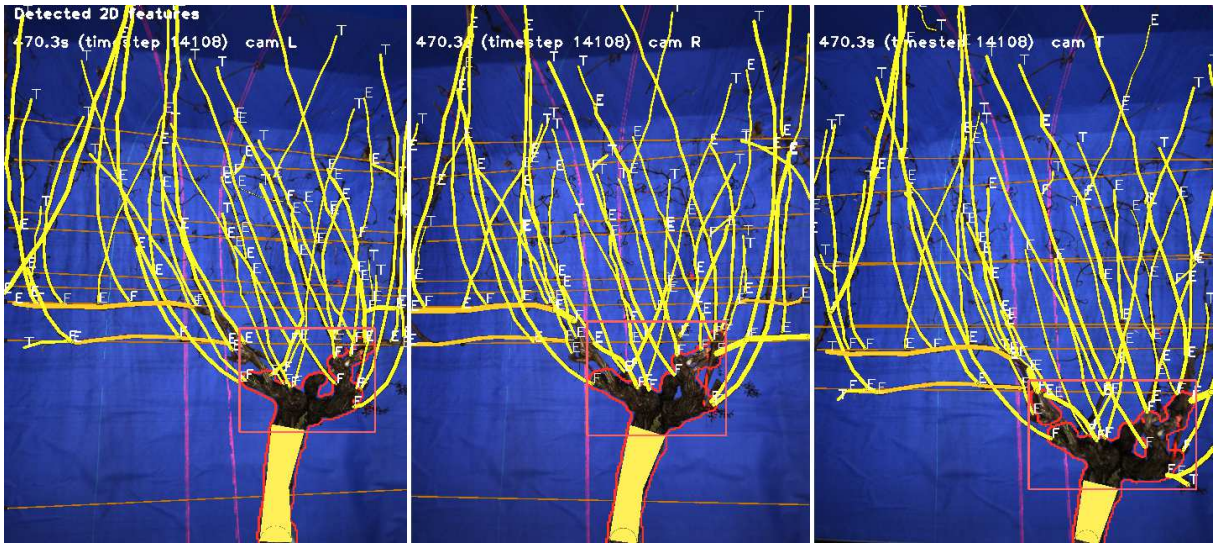
The vines and trellis detected in an image are illustrated in Figure 5b. Segmentation presents a major challenge for many agricultural applications because of variation in lighting and variation in colour (Section 2). For our application, we have simplified the segmentation task by controlling the lighting as much as possible, and by using colour only for foreground/background segmentation.

3.2.3 Correspondence

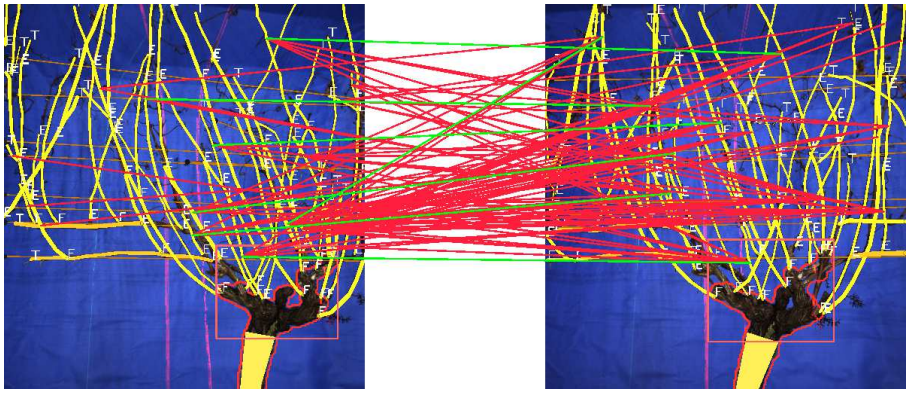
The next stage is to find a correspondence (a set of matches) between canes detected in different images. This is challenging because many cane detections are incomplete or incorrectly-connected, and many are not detected (because they are occluded). We aim to select a correspondence which maximises the completeness of the reconstruction while minimising the amount of incorrect structure.

Given two matched 2D canes, we reconstruct a 3D cane using Levenberg-Marquardt optimisation. 3D canes are modelled as polylines, and to fit a 3D cane model to 2D canes we optimise the 3D polyline's control points to minimise the reprojection error (the distance between the detected 2D canes and the 3D control points projected into the images). Pairs of incompletely-detected canes can often be reconstructed, giving a plausible 3D cane, even though they are not correctly matched. To find all candidate matches between

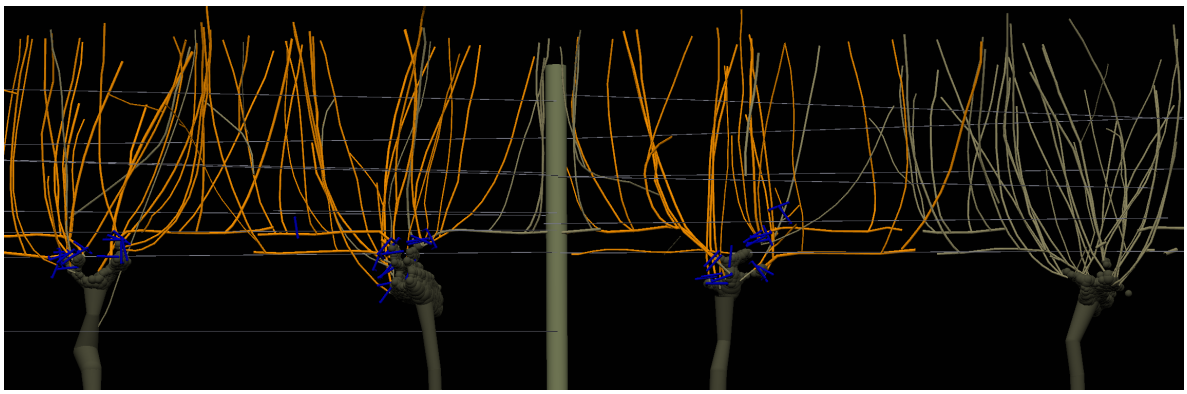
(a) One set of stereo frames. A red laser-line structured light pattern is visible, but this is currently unused.



(b) 2D structure is extracted from each frame, including canes (yellow), the head region (red) and wires (orange).



(c) We find a correspondence between canes detected in each view: all candidate correspondences are shown, and the minimum expected loss correspondence is marked green.



(d) Complete 3D model built up over many frames. Canes to be cut are highlighted in orange.

Figure 5: 3D vine reconstruction pipeline.

Input: 2D canes detected in each image

1. For each pair of canes:
   - Attempt to reconstruct.
   - If successful add to set of candidate matches.
2. For each candidate match:
   - Compute P(match is correct | other matches known) by SVM.
3. Use Gibbs sampling to sample a set of correspondences (a set of sets of matches)
4. For each candidate match:
   - P(match is correct) = relative frequency of match in sample of correspondences
5. Select matches where P(match is correct) $> 1/(1+\alpha)$.

Output: Minimum expected loss correspondence

Figure 6: Algorithm for selecting the minimum expected loss correspondence.

canes we attempt to reconstruct every pair of detected canes. Each detected cane has candidate matches to 1.9 others on average. These matches are interdependent, because each cane can be matched to at most one other cane.

We compute the probability of each match being correct conditional on every other match being known: two canes cannot match if either cane is matched to any other cane, otherwise the match probability is computed from attributes of the reconstructed cane (lengths, thicknesses, epipolar errors, and curvature) using an SVM. The marginal probability of each match being correct is then given by Gibbs sampling (Geman and Geman, 1984).

We formulate the problem of selecting a good correspondence as a problem in decision theory: given the probability of each match being correct, what set of matches should we select to give the best 3D reconstruction? In other words, given a loss function measuring the cost of choosing a particular correspondence, what correspondence C minimises the expectation of this loss? Our proposed loss function L measures the rate of errors in the 3D reconstruction, and is defined to be:

$$L(C) = (\#\text{ incorrect matches in } C) \;+\; \alpha \times (\#\text{ correct matches not in } C). \tag{1}$$

α is a penalty for not selecting a correct match, relative to the cost of selecting an incorrect match. Equivalently, α controls the trade-off between precision and recall. The minimum expected loss correspondence is precisely the set of matches with marginal probabilities greater than $1/(1+\alpha)$.

This procedure for selecting the minimum expected loss correspondence is summarised in Figure 6, a selected correspondence is shown in Figure 5c, and the method is analysed in detail by Botterill et al. (2014a).

3.2.4 Incremental 3D reconstruction

The 3D model is built up incrementally from many images as the robot moves along the row of vines. When new frames are captured, the 2D structure is extracted from each frame. The newly-detected 2D structure (canes, wires, posts) is then either assigned to existing 3D structure, or is used to reconstruct new 3D structure, extending the 3D vine model (Botterill et al., 2012a). The robot's motion is approximately parallel to the row of vines, so vines move right-to-left through the images as the robot moves. The image motion maximising the correlation between consecutive frames is used to initialise the new camera pose.

To assign detected 2D canes to existing 3D canes, each 3D cane is projected to 2D and compared to the detected 2D canes. As with correspondence, we resolve ambiguities by selecting the minimum expected loss assignment of 2D to 3D canes.

Once 2D canes are assigned to 3D structure, we use the g2o bundle adjustment framework (Kummerle et al., 2011) to optimise the camera poses and cane control points to match the assigned measurements. Unassigned 2D canes are corresponded to initialise new structure. Bundle adjustment minimises reprojection error, which is the image distance between detected features and the projected 3D feature. For reprojection errors between a 3D polyline projected to 2D and a detected 2D cane we use the closest distance between each projected control point and the detected 2D cane.

To prevent the size of the optimisation growing over time, we remove structure and constraints from the optimisation once they are reconstructed accurately. Camera poses are removed after a fixed number of timesteps (currently three). 3D structure is removed when it is out of view, and the cameras from which it was observed are no longer being optimised.

The only part of the scene not optimised in the bundle adjustment is the head region, because this has a complex, self-occluding shape which is not easy to model in a feature-based framework. We model the head as a set of spheres, a flexible representation which is ideal for computing collisions with the robot arm and for inferring connections with canes. We use shape-from-silhouette to fit a head model to the silhouettes of the head viewed in multiple images. We initialise a 3D grid of spheres (equivalent to voxels in similar shape-from-silhouette systems, e.g. Szeliski (1993)) throughout the head region, project each sphere into each image, then remove (or shrink) any which fall outside the silhouette of the head region. The resulting "visual hull" is guaranteed to enclose the actual head, so is ideal for collision avoidance (Laurentini, 1994). Figure 5 shows 2D head regions, and the head models inserted into the 3D model of the vines, and Botterill et al. (2014b) describe the efficient implementation in detail.

Finally, we join 3D canes into a connected structure. Joining canes is relatively simple: canes are joined when they are close (within a threshold) to intersecting in 3D.

We customised the optimisation to work well for vines as follows:

- Canes' endpoints are hard to localise accurately, canes wave around as the platform moves along the row, and incorrect assignments are occasionally made. A least-squares cost function is inappropriate for these large residuals, and gives an inaccurate reconstruction. Instead, we minimise the Pseudo-Huber robust cost function (Appendix A.2). This cost function is approximately quadratic when residual errors are low, and linear when they are high, so that outliers are not as heavily penalised as with a least-squares cost.

- Estimating cane thicknesses in the optimiser does not improve the overall accuracy, but increases the size of the problem to solve, so it is more efficient to compute thicknesses separately.

- Polylines become kinked, and control points can become bunched, because of errors and noise in detected 2D canes. We add constraints in the optimisation to penalise large angles and short sections, and to pin wires to posts, canes to their parent cane, and trunks to the ground.

- Rows of vines are almost perfectly straight, so we add constraints to the optimisation to keep the camera's trajectory approximately straight (within about 30cm of a straight line). This prevents the reconstruction from drifting, and enables us to use the physical dimensions of the frame to define a cuboid in which all 3D structure must lie. We can then reject correspondences and assignments which push structure outside this cuboid.

- Sometimes canes are incorrectly reconstructed. These project to the background in subsequent frames, so we can detect them and remove them from the optimisation. Moving canes are not

explicitly modelled and may be reconstructed multiple times, but this check removes the excess reconstructions.

A 3D model of four vines, constructed incrementally over multiple frames, is shown in Figure 5d.

3.3 AI

When people prune vines, they first select several canes to keep, then make cuts to remove the rest. Canes which are long, not too thick or thin, and which will keep the head compact in subsequent years are selected. Corbett-Davies et al. (2012) describes the AI for deciding where to prune. We call the set of canes to keep a *pruning scheme*. We describe each pruning scheme by a feature vector of appropriate attributes of the canes which will be kept (length, position, angle from head, distance below wires, and whether they grow from the head, the trunk, or another cane). We parametrised a cost function measuring pruning quality using vines labelled by viticulture experts. The cost function is a simple linear combination of features. For each vine, every possible pruning scheme is considered in turn, and the highest scoring pruning scheme is selected.

3.4 Robot path planning and control

The path planner computes a collision-free trajectory taking the robot arm to the required cutpoints, even when cutpoints are close to wires and the vine head. The robot's state is represented by the angles of each of its six joints. These joint angles form a point in 6D *configuration space*.

Randomised path planners are widely-used for robot arm path planning. These algorithms require the following three functions: first, a collision detector, which takes a set of joint angles and tests if this configuration puts the robot in self-collision, or in collision with obstacles in its environment. Second, an inverse kinematics solver, which calculates sets of joint angles putting the robot's end effector in a particular pose. Third, a local path planner, which tests whether a path between two points that are nearby in configuration space is collision-free. The local path planner works by interpolating configuration points between the two points, and collision-checking each in turn.

The randomised path planner we use is a Rapidly-exploring Random Tree (RRT)-based planner, RRT-Connect (Kuffner and LaValle, 2000). RRT path planners construct a random tree in configuration space where the edges are collision-free. The tree is grown by sampling new nodes at random (with a bias in the direction of the goal), and then attempting to join each new node to the tree with a collision-free local path. The tree expands throughout the configuration space, and will eventually connect the source and goal with a collision-free path if one exists. Once the goal is reached, the path is simplified and converted into a trajectory for the robot arm to follow. RRT-Connect expands trees from the source and goal pose simultaneously, and joins them when they meet. We use the RRT-Connect implementation from the Open Motion Planning Library in Robot Operating System (ROS) (n.d.).

To make each cut, the robot arm sweeps the cutting tool through the cane. This is a *cut motion*. There are many possible cut motions that cut each cane. To find one that is collision-free, we sample random cut motions, and test each one with the local path planner. When we cannot find a collision-free cut motion close to the head we move the cutpoint further away.

Once we have planned each cut motion, we plan a path which visits each cut in turn. Currently the order is arbitrary, but in future we will optimise the order for speed (Section 6).

Initially, we used the Flexible Collision Library (FCL; Pan et al., 2012) for collision-detection, and the polygonal mesh-based model of the UR5 robot arm provided in ROS, however planning times were unacceptably slow (Section 5.4). Now we use a collision detector based on geometric primitives (cylinders and spheres), which is described in detail in Paulin et al. (2015). Computing whether two cylinders are in collision is com-

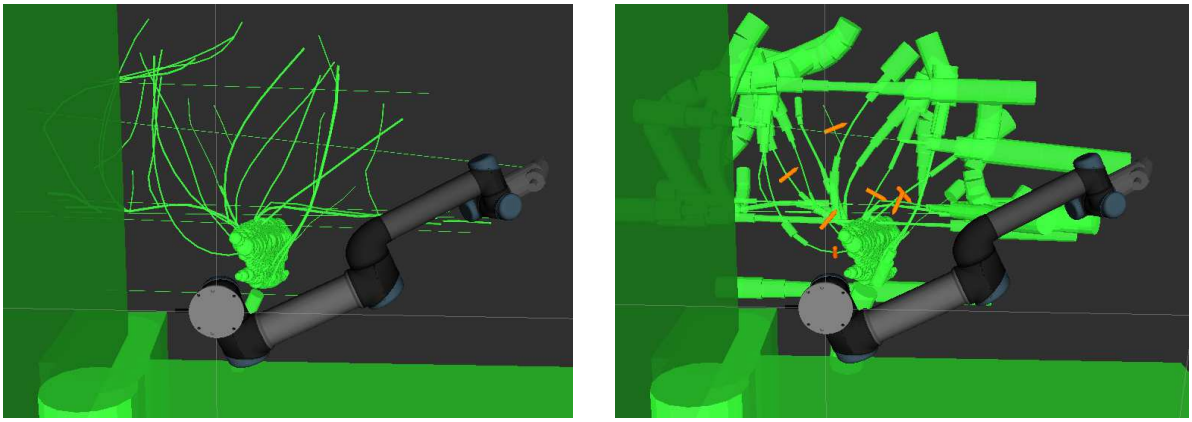


Figure 7: Visualisation of the robot in its workspace, viewed from behind the robot. Obstacles (vines, trellis wires, and the walls of the platform) are green. For collision checking, the scene and arm are modelled as cylinders, spheres, and planes (left). Only obstacles within reach of the robot arm are used. We first find points to cut (marked in orange), then adaptively add safety margins to obstacles before planning paths between cutpoints (right).

putationally cheap compared with computing whether two meshes intersect, and the cylinders are a better approximation to the actual shape of the arm. A simple method to reduce the number of checks required is to sort the geometric primitives by their distance from the robot's base. When checking if part of the arm is in collision, we only check against obstacles that are closer than the maximum distance of the arm part from the base. This optimisation is known as "sweep-and-prune".

The robot's accuracy was tested by placing a coloured marker amongst the vines (a 2cm-wide red square), then directing the robot arm to touch the marker instead of a cutpoint. The robot can touch the marker, indicating that accumulated errors the robot's trajectory, calibration and robot position are typically 1cm. Even 1cm errors lead to collisions, because the path simplification uses 'shortcutting', which pulls the path close to obstacles. The most common approach to reduce the risk of collisions is to add a safety margin to obstacles (Fahimi, 2009). Safety margins prevent the planner from reaching many cutpoints, so we first compute cut motions, then compute the distance to collision for every obstacle, then adaptively add safety margins to all obstacles so that the cut motions are not in collision (Figure 7).

4 Ground truth data and system parametrisation

The system was developed and tested using images captured in the vineyard. To parametrise and test individual computer vision modules we used small sets of images where every cane is hand-labelled (Section 4.1). To test the performance of the entire 3D reconstruction pipeline, and of the system as a whole, we used tens of thousands of images showing entire rows of vines (Section 4.2).

4.1 Ground-truth labelled images

Each module of the computer vision system was parametrised on hand-labelled training data (Figure 8). We labelled images of nine vines (seven *Sauvignon Blanc*, one *Müller Thurgau* and one *Riesling*). For each vine, we labelled the three stereo frames from each of three timesteps. We created a simple GUI application for labelling the images: wires, 2D canes, and connections between canes are labelled by clicking on the image to mark their position. Cane correspondences between stereo frames and between timesteps are marked by clicking canes in a pair of images. In total 3228 2D canes in 81 images are labelled. These images are divided into a training set of six plants, used for system parametrisation, and a test set of three plants, used for the

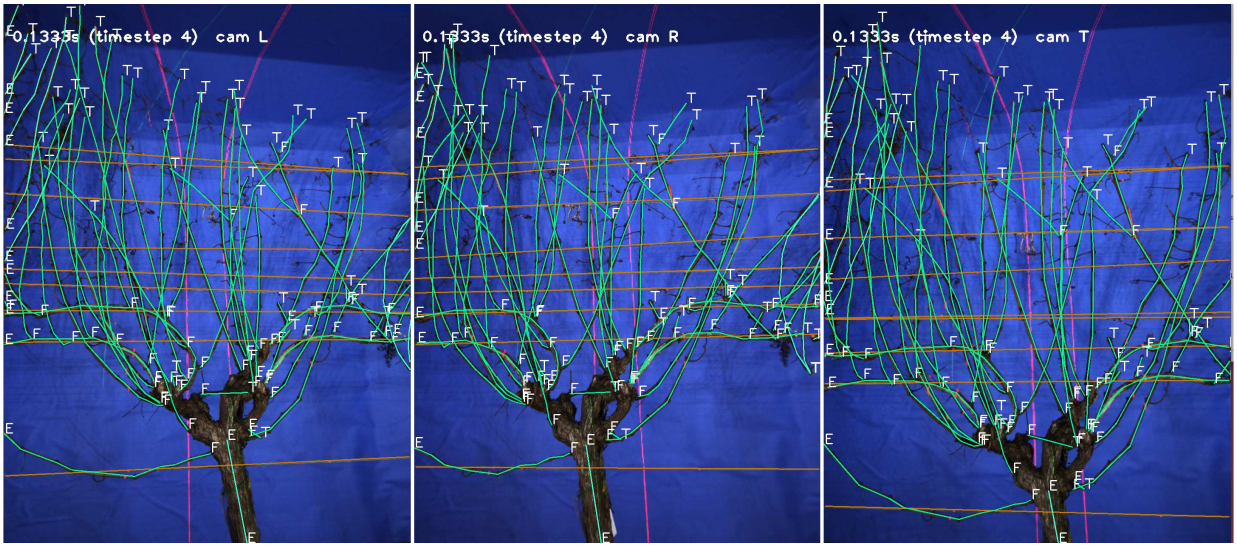


Figure 8: 3 stereo frames from the hand-labelled ground truth dataset. The positions of vines and wires (green and orange), connections between them, and correspondences between them, are labelled by hand.

results in the following section.

We used the labelled training images to train the SVMs used to join cane parts in the 2D cane detector. To obtain training data for each SVM, we extracted feature vectors from the training images, then automatically compared each feature with the ground truth cane network to test if each was correct.

We next trained the SVMs that give probabilities that matches and assignments are correct, using the labelled training images again. We generated training data by finding all candidate matches or assignments, computing a feature vector for each one, and then used the ground truth to determine which were correct.

4.2 Vineyard-scale datasets

The system builds up a 3D model of vines over many frames, so to parametrise and test it we use image sequences showing entire rows of vines. We captured images of five rows of vines at 30 frames per second, showing 226 plants over 373m. We selected a training set by subsampling these frames to match the range of speeds at which the robot operates. We kept 9532 frames; on average one every 8.5cm when moving. These vineyard-scale image sets are too large to feasibly hand-label, so to measure reconstruction accuracy we use the *intersection-over-union* error[2], $\frac{I}{U}$. $\frac{I}{U}$ measures how well the reconstructed vine models match the foreground/background segmented images. It is computed by backprojecting the 3D model into every image and comparing it with the foreground/background segmentation (Figure 9). It is defined as:

$$\frac{I}{U} = \frac{\#(\{\text{model pixels}\} \cap \{\text{foreground pixels}\})}{\#(\{\text{model pixels}\} \cup \{\text{foreground pixels}\})}. \tag{2}$$

$\frac{I}{U}$ is 1 if the 3D model perfectly matches the foreground/background segmentation, and is 0 if there is no alignment. It measures incorrect 3D structure, missing 3D structure, and errors in the camera trajectory. It also includes approximation errors (canes are not actually polylines of uniform thickness), unmodelled junk (dead leaves and tendrils), and canes which haven't been reconstructed yet, but which will be reconstructed in later frames.

[2] $\frac{I}{U}$ is also known as the Jaccard index.

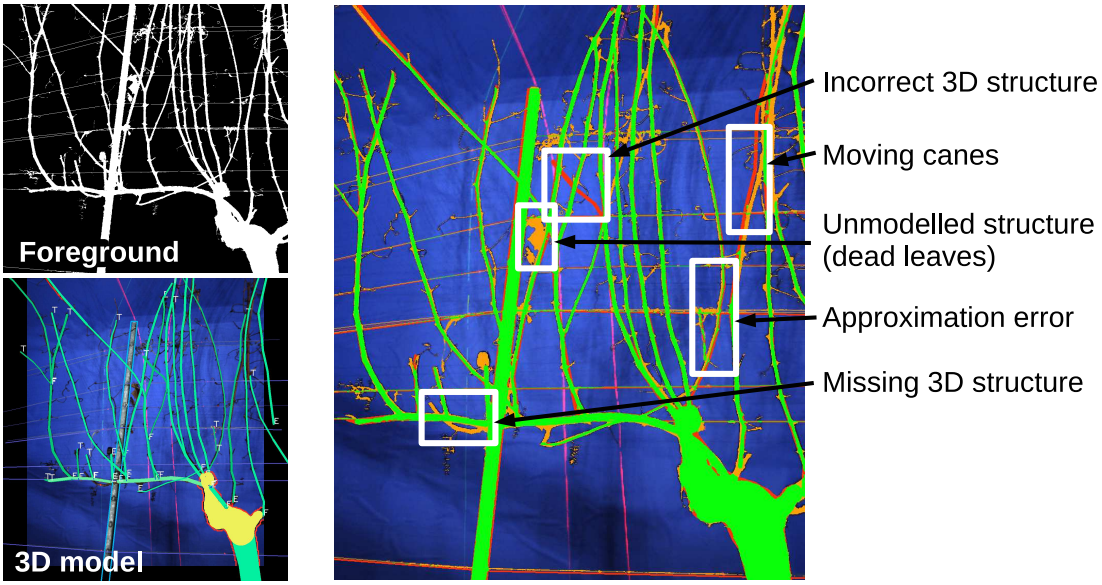


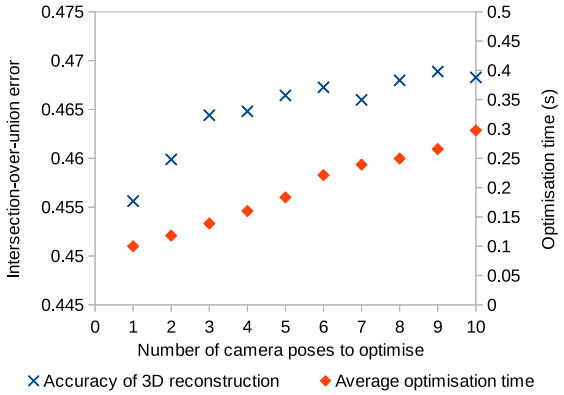
Figure 9: Model quality was evaluated by projecting the 3D model into each frame (bottom-left) and comparing it with the foreground/background segmentation (top-left). The main image shows the differences between the frames: correctly reconstructed structure is marked green, 3D structure which doesn't match the foreground is marked red, and unmodelled 2D structure is marked orange. These errors are due to incorrectly-reconstructed canes, missing canes, approximation error and structure which is not yet modelled. This frame has intersection-over-union error of 0.65 (a precision of 85% at recall of 85%).

We ran the system over this vineyard-scale training set to set the parameters of the g2o optimisation that affect optimisation times and reconstruction accuracy over long sequences of frames. These parameters include the number of camera poses to optimise each time and the number of Levenberg-Marquardt iterations. Results are shown in Figure 10. The reconstruction accuracy increases as more camera poses are kept active in the optimisation, and with more iterations, but the increase is small for more than three poses or six iterations. These values provide a suitable tradeoff between accuracy and efficiency.

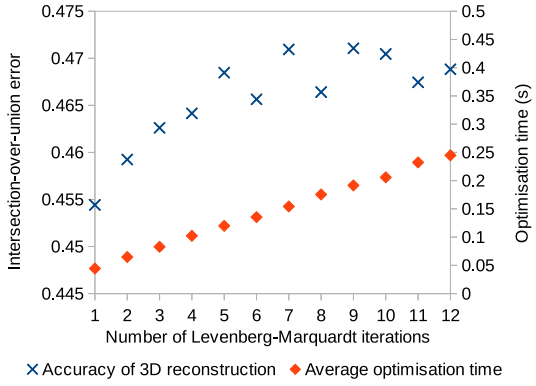
Figure 11 evaluates the Pseudo-Huber robust cost function's t parameter on the vineyard-scale datasets (Appendix A.2). The robust cost gives a small increase in accuracy over least-squares optimisation[3]. A relatively low t parameter of 0.25 standard deviations gives the best performance (lower t values give narrow valleys in the cost function, explaining the poorer performance here).

5 Experimental evaluation

First we evaluate individual modules of the computer vision system in isolation, using the ground-truth labelled test images. Second, we evaluate the computer vision system as a whole on the vineyard-scale datasets. Third, we discuss results from vineyard trials. Finally, we evaluate the computational efficiency of the system. A video showing the different stages of the 3D reconstruction, and the robot pruning vines in the vineyard, is available as supplementary material, and at `http://hilandtom.com/vines.mov`.



(a) Number of camera poses to optimise on each iteration. Camera poses are removed from the optimisation and fixed after this many frames.



(b) Number of Levenberg-Marquardt iterations.

Figure 10: Effect on reconstruction accuracy and computational cost for two optimisation parameters. Both are a trade-off between accuracy and efficiency. The optimisation time includes the constant 28ms for finding a sparse Cholesky decomposition (Kummerle et al., 2011).

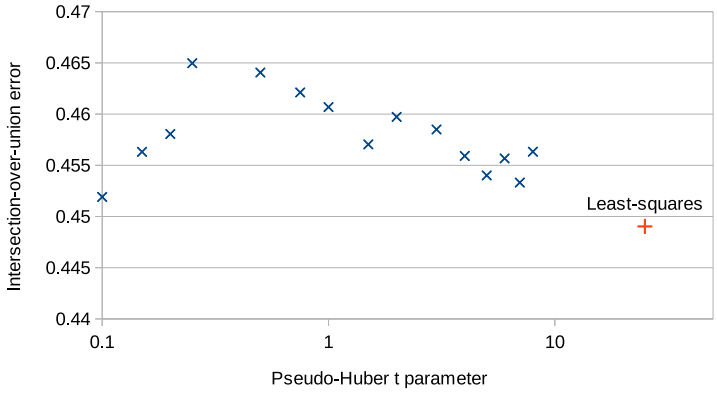


Figure 11: Effect on reconstruction accuracy of the Pseudo-Huber t parameter.

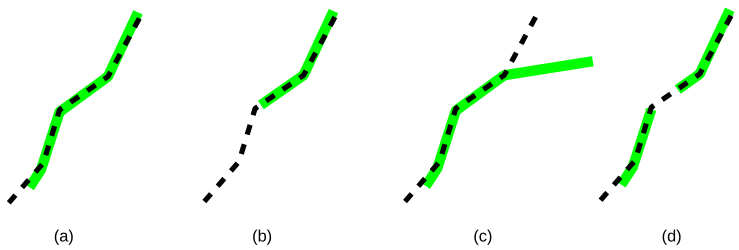


Figure 12: Evaluating precision when comparing labelled ground truth canes (dashed) and detected canes (green). (a) detected cane overlaps 95% of ground truth cane, giving 95% precision. (b), (c) and (d) show cases where canes fail the 'correctly detected' test, giving 0% precision: (b) detected cane overlaps less than 50% of ground truth cane; (c) detected cane goes outside ground truth cane; (d) ground truth cane detected in multiple parts, and each part overlaps less than 50% of the ground truth cane.

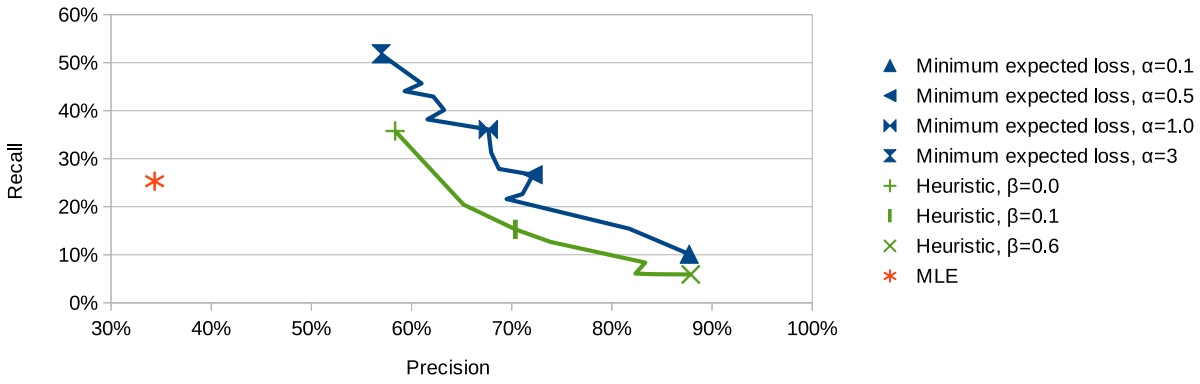


Figure 13: Comparison of different methods of selecting correspondences between canes on test dataset (hand-labelled images of three plants). 1145 matches between 2D canes give a plausible 3D reconstruction; of these 496 are correct.

5.1 Evaluating the computer vision system against ground truth

We evaluated the 2D cane detector on the test dataset (hand-labelled images of three plants; Section 4). We define a cane to be correctly detected if it covers at least 50% of a ground truth cane and is at least 90% within a ground truth cane (Figure 12). The precision is then the total length of correctly-detected canes divided by the total length of detected canes. The recall is the total length of correctly-detected cane divided by the total length of ground truth canes. On the test dataset, the 2D cane detector achieves 58% precision with 49% recall[4]. Most errors are due to canes being detected in several parts, rather than as one complete cane, and these parts are still useful for 3D reconstruction. However, for more complete 3D models it is necessary to combine measurements from many frames.

Next, we use the hand-labelled test dataset to evaluate three methods for selecting a correspondence between canes: first, the minimum expected loss correspondence. Second, the maximum likelihood correspondence (the correspondence found most often by Gibbs sampling). Third, a common heuristic used to reject incorrect matches: reject a match between 2D canes if the likelihood of the second best match to either cane is within some threshold β of the likelihood of the best match. A similar heuristic is commonly used for matching SIFT features (Lowe, 2004; Szeliski, 2010).

Figure 13 shows the precision and recall of these proposed methods. The minimum expected loss correspondence consistently outperforms the other options in terms of precision and recall. It has 72% precision at 27% recall with $\alpha = 0.5$, whereas the maximum likelihood correspondence has precision of only 34% at 25% recall. The maximum likelihood correspondence is selecting ambiguous matches where a better reconstruction would be given by not selecting these matches. By contrast, the heuristic rejects too many correct correspondences as being ambiguous. The correspondences that are missed are predominately matches between partly detected 2D canes (either there are multiple plausible reconstructions, or there is not enough overlap to reconstruct), and incorrectly labelled training data. Botterill et al. (2014a) has further evaluation and analysis of the minimum expected loss correspondence.

[3]Earlier in development, when correspondences and assignments were less reliable, the robust cost function gave a substantial increase in performance, and was essential for reconstructing more than one or two plants. Now that other parts of the system perform better, this is no longer the case.

[4]Interestingly, the test set performance is considerably better than the training set performance (51%/42%). This is due to variation in the complexity of plants, and the person labelling becoming more accurate with time. Reliably labelling training data is known to be challenging (Russakovsky et al., 2015).

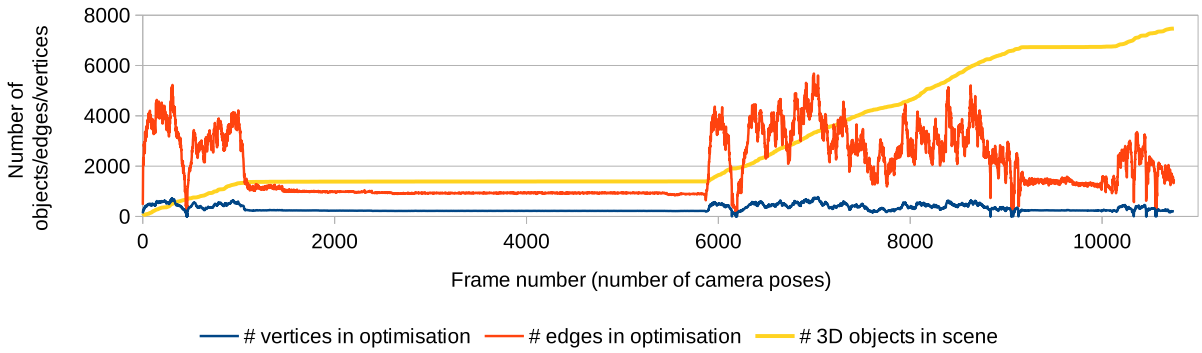


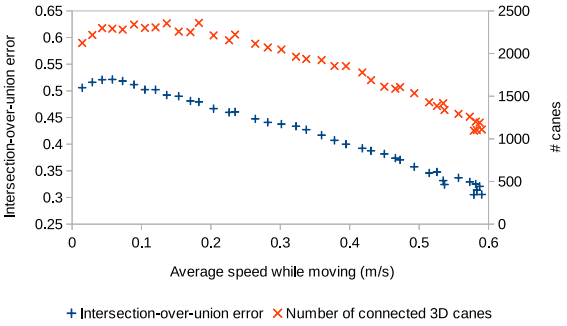
Figure 14: The number of 3D objects in the reconstruction grew as the robot moved along the row and observed more vines, but the incremental optimisation kept the size of the optimisation, measured by the number of vertices and edges, bounded (*Sauvignon Blanc 1* dataset).

5.2 Evaluation on vineyard-scale datasets

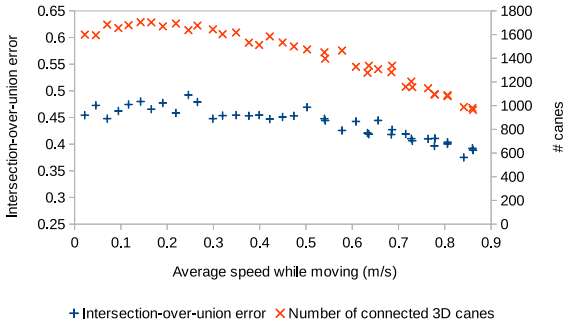
To evaluate the performance and scalability of the 3D reconstruction, we tested the computer vision system on large vineyard-scale datasets showing entire rows of vines. This paper is the first time an evaluation of the entire computer vision system has been presented. Figure 14 shows the size of the optimisation as one row of vines was reconstructed. 59 plants (7198 3D objects) were reconstructed from 10460 stereo triples (i.e. 10460 camera poses). g2o models the optimisation as a graph where the *vertices* are the 3D points and camera poses which are estimated, and the *edges* represent constraints between vertices, e.g. a feature measurement in an image is represented by an edge connecting a camera pose with the feature's 3D point. Full bundle adjustment would require 38000 vertices (145000 dimensions) which is not feasible for real time operation, but the incremental bundle adjustment optimised at most 772 vertices and 5682 edges at any one time.

The vineyard-scale datasets allowed us to evaluate the effects of robot speed and framerate on the 3D reconstruction. Two rows of *Sauvignon Blanc* vines were captured at 30 frames per second. We evaluated accuracy at different speeds by subsampling the frames (from every frame to every 40th frame). The system runs at a framerate of 2.5 frames per second. Figure 15 shows that the accuracy and completeness of the reconstruction are similar for a range of speeds up-to 0.25m/s, as having extra frames showing the same vines provides little additional information (it is generally more accurate and efficient to have few widely-spaced views rather than many nearby views; Nistér, 2001). For higher speeds, the accuracy and completeness decrease as incorrect 2D-to-3D assignments become more common and more 3D structure is missed. To summarise overall performance, we measured the length of the rows (post-to-post distance) at 95.99m and 80.15m, and compared these distances to the 3D model (Figure 16). The estimated row lengths are all within 1% of the measured distance, for subsets of up-to every 20th frame, and have errors of 0.49% on average. This level of accuracy enables the robot arm to cut canes after the platform has moved further along the row. At a speed of 0.25m/s, the platform takes 7s to move between two plants, or just over 6 minutes to reconstruct an entire row. This is fast compared with the time required for the robot arm to execute path plans, which is 55s per plant, or one hour per row. For comparison, a human would take around two hours to prune one of these rows.

Next, we tested the AI pruning algorithm on the reconstructed vines. The AI was developed and trained using simulated images of vines. Corbett-Davies et al. (2012) evaluated the system's performance on 100 simulated vines by getting a viticulture expert to rate the quality of the pruning decisions made by the AI, and also to rate the quality of decisions made by a human pruner. The decisions made by the AI were judged to be better than the human for 30 vines, and at least as good for 89 vines. Figure 17 shows the pruning decisions made for 59 real vines. On each plant the AI selects two long canes to keep, and two to cut at

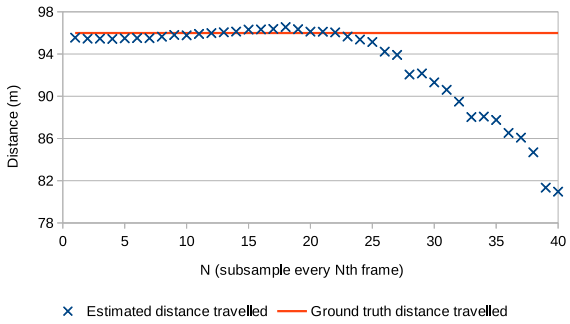


(a) *Sauvignon Blanc 1* dataset (9110 frames/59 plants).

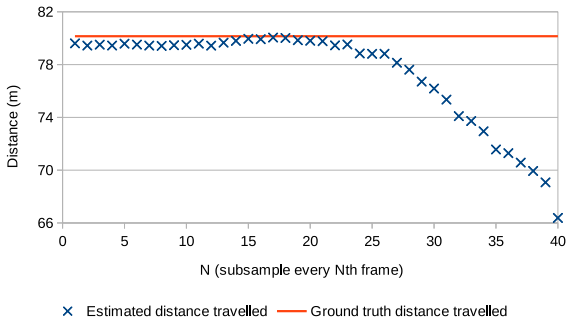


(b) *Sauvignon Blanc 2* dataset (21329 frames/50 plants).

Figure 15: Graph comparing reconstruction accuracy ($\frac{I}{U}$) and completeness (the number of canes reconstructed and connected into the vine models) versus mean speed. The maximum speed is 1.8 times the mean speed for these datasets.



(a) *Sauvignon Blanc 1* dataset.



(b) *Sauvignon Blanc 2* dataset.

Figure 16: Graph showing distance travelled estimated by the computer vision software compared with the measured distance, for different subsets of frames (every frame, every second frame, up to every 40th frame).

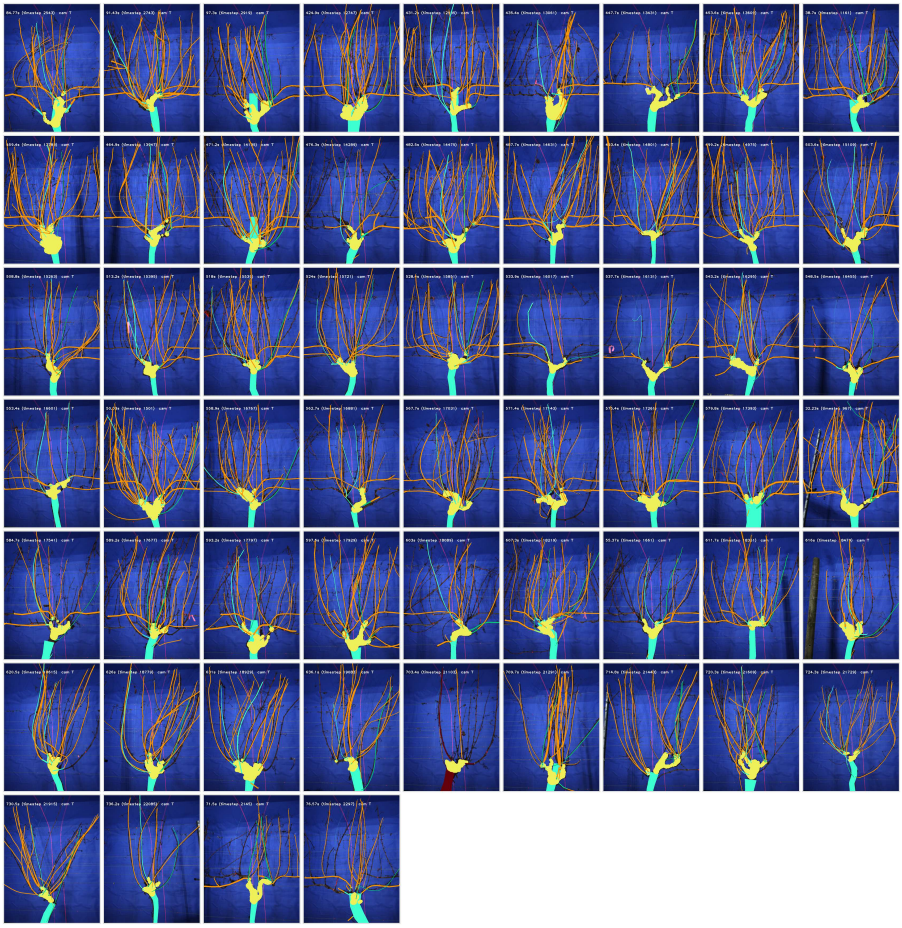


Figure 17: Every *Sauvignon Blanc* vine from one row (58 vines), and the pruning decisions made by the AI. The figure shows the variation in shape and structure of vines along a row. Two canes on each vine should be kept (marked green), the head regions are marked yellow, and canes to remove are marked orange.

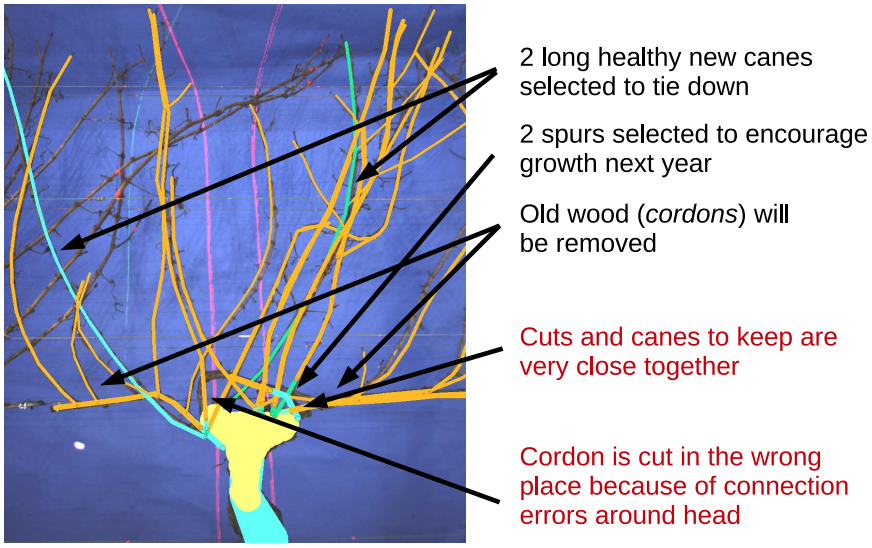


Figure 18: AI pruning decisions for a *Chardonnay* vine. The figure shows the 3D model of the vine back-projected onto one frame. Canes to remove are marked in orange, canes to keep are marked green, and the head region is marked yellow. The red colour is a structured light pattern, and is not currently used.

10cm (to 'spur'). The remaining canes should be removed. A limitation of the present system is that many plant models are missing canes (e.g. partly-reconstructed canes which are not connected to the plant), and many plants have structural errors close to the head, precisely where the cuts should be made (Figure 18). Section 6 discusses future plans for improving the 3D models in order to address these issues.

5.3 Vineyard trials

We tested the pruning robot on a row of *Sauvignon Blanc* vines in the Lincoln University vineyard in July 2015. The computer vision system built up a 3D model of the vines as the winch pulled the platform along the row. The AI selected canes to cut from each vine model. When the arm was within reach of the cutpoints on a vine, the system signalled us to stop the winch. The path planner then planned paths to make all of the cuts, and the arm executed these plans, before retracting. The system then signalled us to restart the winch, to move to the next vine. The trial is recorded using rosbag (from ROS), so that ROS's model of the arm in motion around the vines can be visualised afterwards.

Figure 19, and the supplementary video (`http://hilandtom.com/vines.mov`) show the arm pruning a vine, alongside the ROS's model of the arm pruning the 3D vine model. All six canes which should be cut are hit by the cutting tool, but four canes are pushed aside rather than being cut. The cut motions extend 5cm either side of the target canes, so hitting these canes indicate that in this experiment, the total accumulated error from modelling the vines, calibration, the robot's trajectory and robot positioning accuracy is of a similar magnitude. For this vine, detecting that the platform has stopped and planning paths to all cutpoints took 5s, and executing the paths took 115s. This trial demonstrates that the complete integrated system can cut canes, although it is certainly not reliable: trials on twelve other plants were mostly aborted due to the cable tangling, connection failures, and cameras moving out of calibration.

5.4 Computational performance

The image acquisition, computer vision, and path planning software runs on a desktop PC with a six-core Intel i7 2.93GHz processor. The system runs asynchronously and grabs new frames as soon as the processing of previous frames is complete. Figure 20 shows that foreground/background segmentation and

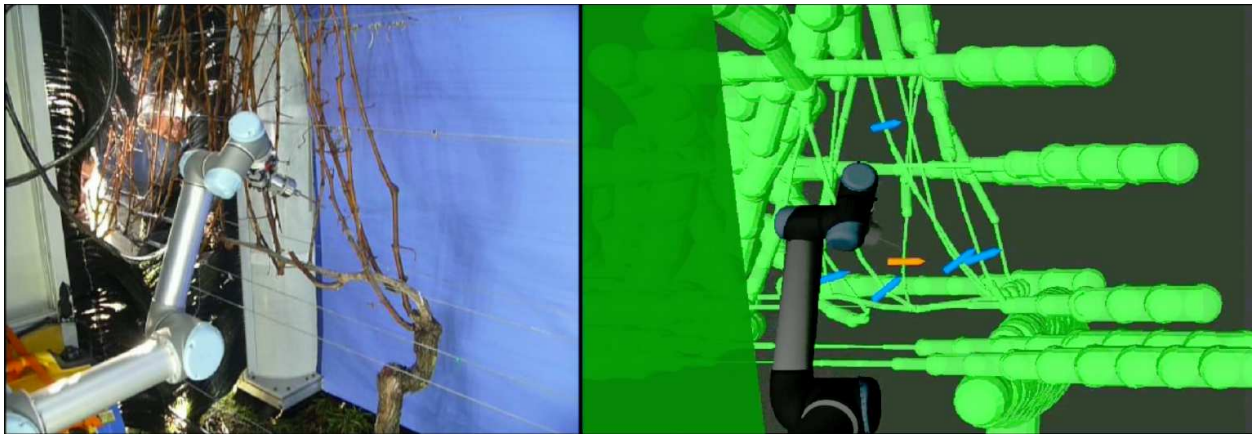


Figure 19: A frame from the supplementary video, showing the robot arm reaching to prune the last of six cutpoints on a *Sauvignon Blanc* vine at Lincoln University vineyard (left). The visualisation on the right shows the model of the robot arm, the vines and the cutpoints.

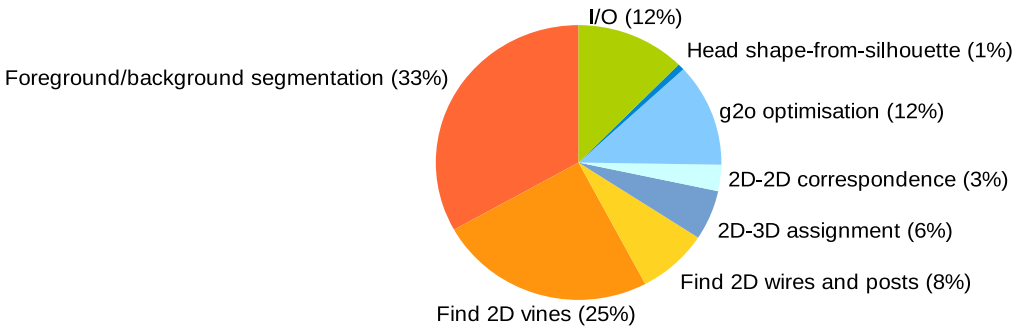


Figure 20: Computer vision system CPU time broken down by task.

Table 1: Impact of different collision detectors on path planning times and success rates. Median time per path for planning and executing plans to-and-from an average 8.4 cutpoints on each of 38 vines, using a single CPU core. Each path moves the arm from one cutpoint to the next.

Collision detector	Planning time (s)	Planning time with safety margin (s)	Execution time (s)
FCL mesh models	1.82	1.98	5.87
Geometric primitives	0.37	0.39	5.91
Geometric primitives, sorted	0.13	0.17	5.98
Geometric self-collisions only	0.012	-	1.55
Comparable published results	0.4-1.0	-	-

2D structure detection are the most expensive computer vision tasks. Foreground/background segmentation is slow because of the large number of pixels to label. 2D vine structure detection is slow because of the large number of decisions to make when joining primitives. These use SVM classification, and both could be sped up with more use of boosting (i.e. simple classifiers that can classify most examples, with the SVM applied only to the remainder). Other parts of the system are relatively efficient because they operate on a small number of higher-level parts, such as canes. By contrast, other plant modelling systems require minutes per frame for either dense stereo processing (Dey et al., 2012) or to compute high-resolution volumetric models (Kumar et al., 2012; Paproki et al., 2011).

Simple multithreading throughout the system (e.g. segmenting the three frames from each camera concurrently) distributes the computational load over an average of four cores. On the complex *Sauvignon Blanc* vines, the system runs at an average of 2.5 frames per second.

We evaluated the effects of the customised collision detector on the time required for planning paths. We compared it with the FCL collision detector, which uses mesh intersections to detect collisions, and octree spatial partitioning to find these intersections efficiently (Pan et al., 2012). Table 1 shows the time required for path planning and execution. To plan one path requires an average of 14000 collision checks, and in this trial the path planner always succeeded in finding a collision-free path. Planning times are 12 times faster when using the customised collision checker with the objects sorted by their distance from the robot's base, compared with the FCL planner. The collision checker based on geometric primitives is fast enough that the time required for planning is insignificant compared with the time required for execution (2.5s per plant, compared to 53s for execution). These planning times are considerably faster than than previous results for agricultural robots using the same UR5 arm (Nguyen et al., 2013; Lee et al., 2014).

We also tested our collision detector without obstacles, so that it is only checking self collisions and collisions with the platform walls. Paths are one quarter of the length on average, indicating that much of the time required to prune a vine is because of the extra movement required to avoid other vines.

6 Discussion and lessons learnt

We developed a complete robot system for pruning vines. The system is a prototype, and doesn't work well enough to replace human pruners, but it does demonstrate many technologies that will enable pruning to be fully automated. This section summarises the lessons learnt from the project, and makes recommendations for how to design a commercially-viable pruning robot.

6.1 Computer vision system

The computer vision system works by breaking the problem up into a cascade of smaller components. Every component is tolerant to errors from earlier stage, and even though many feature matches are missed, a near-complete structure is built up over multiple frames. When canes are initially only partly visible, they are extended when more complete detections come into view. The 3D model is a high-level parametric model, where vines are represented by polylines. This model is essential for the AI to decide which canes to prune, and is also useful for efficient path planning.

The accuracy of each component can be evaluated on ground truth labelled images, in isolation of later components. This design enables us to parametrise each component in turn, and avoids the challenging problem of parametrising many components jointly[5]. Developing and parametrising each component individually to match ground truth was a breakthrough which gave a large performance increase for the entire system.

The most significant limitation of the computer vision system is incorrect 3D structure around the vine head, which originates from errors in 2D cane detection. The bottom-up parse used to find the 2D canes makes local 'join' decisions independently of the large scale structure of the plant. To address this we are researching ways to parse the images using both local and global knowledge of the vine's structure. A further limitation is that moving canes are not modelled. Modelling dynamic objects from moving cameras is challenging in general (Sheikh and Simon, 2014), but canes are a special case (one end stays still while the other sways). This knowledge could be used to model their motion within the optimisation.

One way in which the completeness of the 3D models could be improved is by fusing additional sensors. Preliminary work shows that laser-line structured light gives high-resolution depths, even on thin wires, which can be used in conjunction with vision for vine modelling (Botterill et al., 2013a).

6.2 System architecture

The system was designed so that the platform moves continuously while a 3D model of the vines is built, and while the robot arm makes the cuts. At present, the platform stops while the arm makes cuts. To prune while the platform is in motion would require the arm to react dynamically to the platform's movement. This dynamic path planning is challenging, and it is unclear whether a significant speedup over a start-stop system could be obtained. A consequence of the continuously-moving design is that a long chain of hardware and software components must all work reliably over multiple frames for a vine to be modelled and pruned. Even when stopping to cut, the accuracy in cutting a cane is limited by the combination of errors in the 3D model (including approximation errors and errors in camera calibration), movement of the canes, errors in the robot's estimated trajectory, errors in the robot-to-camera calibration and errors from the robot's positioning accuracy. By contrast, a robot that stops at every vine for both imaging and cutting could make more accurate cuts and would be more robust, because it has a shorter chain of components where errors can accumulate, and has fewer points of failure. Stopping to model each vine would also give the option of 3D reconstruction methods including voxel-based shape-from-silhouette and coded light, and would avoid the need to reconstruct 3D models incrementally or to reconstruct the platform's trajectory. However, one application for which there is much industry demand (Hall et al., 2002; Araus and Cairns, 2014) and for which the current architecture is ideal, is for measuring every vine in the vineyard for crop monitoring and for phenomics research.

6.3 Robot motion planning and cutting technology

The main bottleneck in the current system is the time required for the robot arm to execute paths, and we are currently researching ways to optimise execution times. These optimisations will include improved

[5]Similarly, the lower layers in deep neural networks are parametrised sequentially from the bottom layer up, because training all layers jointly gives poor results (Hinton et al., 2006).

ordering of cutpoints (an example of the travelling salesman problem, which is small enough to solve exactly), and finding nearby robot configurations to cut nearby canes. These optimisations will be possible because of our relatively fast planning times. There is also potential to further improve planning times by choosing a different path planning algorithm (many are available in the Open Motion Planning Library).

One reason for slow path execution times is that the articulated robot arm sometimes requires large joint rotations in order to move the cutting tool a short distance (see supplementary video). Future simulation studies will investigate whether an arm with more degrees of freedom, or with a different joint configuration (e.g. a robot with parallel manipulators), will enable faster movements.

The current cutting tool requires 10cm-long cut motions to cut the canes, and often fails to make a separation cut. The long cut motions required limit which canes can be cut without hitting other canes. Quality pruning requires many cuts that are close to the canes' bases (about 1cm from the base for *Sauvignon Blanc*). In future, we will use a compact secateur-based cutter, which will allow cuts to be made closer to the head and closer to other canes. We will also eliminate the tangle-prone external power cable.

7 Conclusions

This paper described a system for robotically pruning grape vines. The robot images the vines with a trinocular stereo rig, then uses a customised feature-matching and incremental bundle adjustment pipeline to build a 3D model of the vines, while estimating the robot's trajectory. To overcome the challenges of modelling the self-occluding and self-similar canes we customised the 3D reconstruction system to exploit knowledge of the vines' structure.

The system reconstructed 3D models of entire rows of vines while the robot moved at 0.25m/s, and the robot's trajectory estimate was accurate to within 1% of ground truth. The 3D models are accurate enough that the robot arm can reach and make cuts. The 3D models are parametric models of the vine's structure, and are used both for deciding which canes to cut, and for efficient collision-free path planning for the robot arm. To our knowledge, this is the first attempt to automate cane pruning, where a complete and connected model of the plant is required.

The current system was designed to model the vines as the platform moves, but many of the techniques developed would also be suitable for a robot that stops at every plant. In addition, the ability to build 3D vine models in real-time from a moving platform is useful for crop monitoring and phenomics.

A Appendix

A.1 Support Vector Machines

Many different tasks in the computer vision system are formulated as classification problems and solved using Support Vector Machines (SVMs; Schölkopf et al., 2000). SVMs are a standard machine learning tool for binary classification of feature vectors. An SVM uses a decision function to classify each feature vector as positive or negative (given by the sign of the decision function). The parameters of the decision function are fitted to minimise classification error on a labelled training set of feature vectors, then this decision function is used to classify future examples. A simple example is deciding whether a pixel is foreground or background based on its colour vector, and a more complex example would be deciding whether two detected 2D cane parts are part of the same cane, based on a feature vector including the relative positions, thicknesses and angles between them.

A linear SVM finds a hyperplane in feature vector space which separates positive and negative feature vectors,

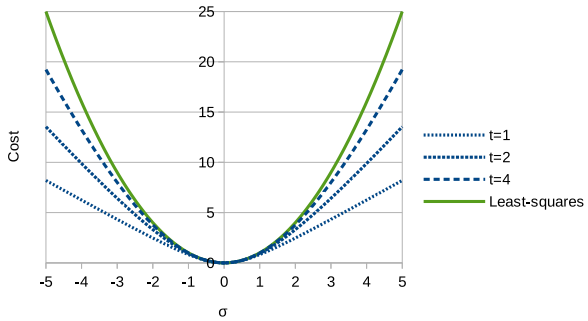


Figure 21: Pseudo-Huber cost function.

and classification with linear SVMs is computationally cheap. Radial Basis Function SVMs (RBF-SVMs) are more powerful as they can represent more complex decision surfaces. SVMs have hyperparameters which control the penalty for misclassified training examples (ν) and the complexity of the decision surface (γ), these should be set so that the decision surface is complex enough to represent the trends present in the data, but without overfitting. Often the SVM will generalise better to new examples if it is trained on a minimal subset of the features; this reduces the complexity and dimensionality of the model. We use standard best practice to choose hyperparameters and a feature subset without overfitting: we normalise feature vectors, then select hyperparameters and a feature subset which maximise the six-fold cross-validation score (Hsu et al., 2003). Hyperparameters are selected by grid search, and the feature subset is selected by backward feature selection (repeatedly removing features from the feature subset until the score no longer improves; Jain and Zongker, 1997).

For classification, class weights control the trade-off between the rates of false positives and true negatives. Alternatively we can fit a sigmoid curve to the SVM's decision function to map the SVM response to class membership probability (Zadrozny and Elkan, 2002). For multi-class classification, we train one SVM per class (against every other class), then select the class with the highest decision function response. Many alternative machine learning algorithms could be used instead of SVMs, e.g. Neural Network or Random Forests. These methods all have similar performance on low-dimensional standard machine learning datasets (Meyer et al., 2003; Caruana and Niculescu-Mizil, 2006).

A.2 Robust optimisation

Optimisation methods like Levenberg-Marquardt find models minimising the sum of squared residual errors, $\sum_i r_i^2$. This is a maximum likelihood estimate of the model if the residuals $\{r_i\}$ are independent and normally-distributed. In practice reprojection errors are not approximately normally-distributed—large residuals often occur due to incorrect 2D-to-3D assignments and moving canes. A more appropriate error term to minimise is the Pseudo-Huber robust cost (Hartley and Zisserman, 2003, Appendix 6.8):

$$\text{cost} = \sum_i 2t^2 \left(\sqrt{1 + \left(\frac{r_i}{t}\right)^2} - 1 \right) \tag{3}$$

The Pseudo-Huber cost is approximately quadratic when residual errors are low, and linear when they are high, with t controlling where this transition occurs (Figure 21). The Pseudo-Huber cost is minimised in the g2o Levenberg-Marquardt framework by the Iteratively-Reweighted Least Squares method (Hartley and Zisserman, 2003, Appendix 6.8): residuals are weighted so that their square is the Pseudo-Huber cost.

A.3 Supplementary material

A video showing the different stages of the 3D reconstruction, and the robot pruning a vine in the vineyard (alongside the corresponding 3D vine and arm model in ROS), is available as supplementary material, and at `http://hilandtom.com/vines.mov`. It is best played with VLC media player.

Acknowledgements

Thanks to all the technicians and support staff at University of Canterbury and Lincoln University who helped this project to succeed. Thanks to Hilary McMillan for help revising this manuscript.

References

J. L. Araus and J. E. Cairns. Field high-throughput phenotyping: the new crop breeding frontier. *Trends in Plant Science*, 19(1):52–61, 2014.

C. W. Bac, E. J. van Henten, J. Hemming, and Y. Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, pages 888–911, 2014. doi: 10.1002/rob.21525.

T. Botterill, R. Green, and S. Mills. Reconstructing partially visible models using stereo vision, structured light, and the g2o framework. In *Proc. Image and Vision Computing New Zealand*, 2012a. doi: 10.1145/2425836.2425908.

T. Botterill, S. Mills, R. Green, and T. Lotz. Optimising light source positions to minimise illumination variation for 3D vision. In *3DIMPVT*, pages 1–8, 2012b. doi: 10.1109/3DIMPVT.2012.13.

T. Botterill, R. Green, and S. Mills. Detecting structured light patterns in colour images using a support vector machine. In *Proc. Int. Conf. Image Processing*, pages 1–5, Melbourne, Australia, September 2013a. doi: 10.1109/ICIP.2013.6738248.

T. Botterill, R. Green, and S. Mills. Finding a vine's structure by bottom-up parsing of cane edges. In *Proc. Image and Vision Computing New Zealand*, pages 1–6, Wellington, NZ, November 2013b. doi: 10.1109/IVCNZ.2013.6727001.

T. Botterill, R. Green, and S. Mills. A decision theoretic formulation for sparse stereo correspondence problems. In *Proc. International Conference on 3D Vision (3DV)*, 2014a. doi: 10.1109/3DV.2014.34.

T. Botterill, R. Green, and S. Mills. Voxel carving for collision avoidance for a vine pruning robot arm. In *Proc. IVCNZ*, pages 1–6, 2014b. doi: 10.1145/2683405.2683419.

T. Botterill, M. Signal, S. Mills, and R. Green. Design and calibration of multi-camera systems for 3D computer vision: lessons learnt from two case studies. In *Proc. Pacific-rim Symposium on Image and Video Technology workshop on Robot Vision (PSIVT-RV)*, pages 1–14, 2015.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

T. Brogårdh. Present and future robot control development—an industrial perspective. *Annual Reviews in Control*, 31(1):69–79, 2007.

R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proc. Int. Conf. Machine learning*, pages 161–168, 2006.

Y. Chéné, D. Rousseau, P. Lucidarme, J. Bertheloot, V. Caffier, P. Morel, É. Belin, and F. Chapeau-Blondeau. On the use of depth camera for 3D phenotyping of entire plants. *Computers and Electronics in Agriculture*, 82:122–127, 2012.

L. Christensen. *Raisin Production Manual*, chapter Vine Pruning, pages 97–101. UCANR Publications, 2000.

S. Corbett-Davies, T. Botterill, R. Green, and V. Saxton. An expert system for automatically pruning vines. In *Proc. Image and Vision Computing New Zealand*, 2012. doi: 10.1145/2425836.2425849.

J. Davis, B. Jähne, A. Kolb, R. Raskar, and C. Theobalt. Time-of-flight imaging: Algorithms, sensors and applications. Technical Report 10, Dagstuhl Seminar Report, 2013.

Z. De-An, L. Jidong, J. Wei, Z. Ying, and C. Yu. Design and control of an apple harvesting robot. *Biosystems Engineering*, 110(2):112 – 122, 2011. ISSN 1537-5110. doi: http://dx.doi.org/10.1016/j.biosystemseng.2011.07.005.

D. Dey, L. Mummert, and R. Sukthankar. Classification of plant structures from uncalibrated image sequences. In *Workshop on the Applications of Computer Vision*, 2012.

G. Dryden. 2014 viticulture monitoring report. Technical Report ISBN 978-0-478-43760-7, Ministry for Primary Industries and New Zealand Winegrowers, 2014.

R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/361237.361242.

Y. Edan, D. Rogozin, T. Flash, and G. E. Miles. Robotic melon harvesting. *IEEE Transactions on Robotics and Automation*, 16(6):831–835, 2000.

N. M. Elfiky, S. A. Akbar, J. Sun, J. Park, and A. Kak. Automation of dormant pruning in specialty crop production: An adaptive framework for automatic reconstruction and modeling of apple trees. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 65–73, 2015.

F. Fahimi. *Autonomous Robots*. Springer, 2009.

FAO. Faostat: Crops, national production. online, 2013. URL `http://faostat.fao.org`. retrieved September 2013.

J.-S. Franco and E. Boyer. Efficient polyhedral modeling from silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):414–427, 2009.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741, 1984.

J. Geng. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011.

G. Gimel'farb. Probabilistic regularisation and symmetry in binocular dynamic programming stereo. *Pattern Recognition Letters*, 23(4):431–442, 2002.

A. Hall, D. Lamb, B. Holzapfel, and J. Louis. Optical remote sensing applications in viticulture-a review. *Australian Journal of Grape and Wine Research*, 8(1):36–47, 2002.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, second edition, 2003. ISBN 0-521-54051-8.

G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

M. Hofer, M. Maurer, and H. Bischof. Improving sparse 3D models for man-made environments using line-based 3D reconstruction. In *Proc. International Conference on 3D Vision (3DV)*, volume 1, pages 535–542, 2014.

C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification, 2003.

A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *Trans. Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, F. Dellaert, M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. Ben-Shahar. Computer vision for fruit harvesting robots–state of the art and challenges ahead. *International Journal of Computational Vision and Robotics*, 3(1): 4–34, 2012.

M. Karkee, B. Adhikari, S. Amatya, and Q. Zhang. Identification of pruning branches in tall spindle apple trees for automated pruning. *Computers and Electronics in Agriculture*, 103:127–135, 2014.

M. W. Kilby. Pruning methods affect yield and fruit quality of 'merlot' and 'sauvignon blanc' grapevines. Technical Report AZ1051; Series P-113, College of Agriculture, University of Arizona, 1999.

J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995–1001, 2000.

P. Kumar, J. Cai, and S. Miklavcic. High-throughput 3D modelling of plants for phenotypic analysis. In *Proc. IVCNZ*, pages 301–306. ACM, 2012.

R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.

A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

J. J. H. Lee, K. Frey, R. Fitch, and S. Sukkarieh. Fast path planning for precision weeding. In *Australasian Conference on Robotics and Automation*, 2014.

P. Li, S.-h. Lee, and H.-Y. Hsu. Review on fruit harvesting method for potential use of automatic fruit harvesting systems. *Procedia Engineering*, 23:351–366, 2011.

X. Li, B. Gunturk, and L. Zhang. Image demosaicing: A systematic survey. In *Electronic Imaging*, page 68221J, 2008.

W. Liu, G. Kantor, F. De la Torre, and N. Zheng. Image-based tree pruning. In *International Conference on Robotics and Biomimetics (ROBIO)*, pages 2072–2077, 2012.

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

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55(1):169–186, 2003.

B. Micusik and H. Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. In *Proc. International Conference on 3D Vision (3DV)*, volume 1, pages 13–19, 2014.

T. T. Nguyen, E. Kayacan, J. De Baerdemaeker, and W. Saeys. Task and motion planning for apple-harvesting robot. In *Agricontrol*, volume 4, pages 247–252, 2013.

Z. Ni and T. F. Burks. Plant or tree reconstruction based on stereo vision. In *Annual meeting of the American Society of Agricultural and Biological Engineers*, 2013.

O. Nir and R. Linker. Tree skeleton extraction by close-range laser telemetry as first step toward robotic pruning. In *Proceedings EurAgEng*, 2014.

D. Nistér. Frame decimation for structure and motion. In *Revised Papers from Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pages 17–34, London, UK, 2001.

C. Nock, O. Taugourdeau, S. Delagrange, and C. Messier. Assessing the potential of low-cost 3d cameras for the rapid measurement of plant woody structure. *Sensors*, 13(12):16216–16233, 2013.

S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh. Automated visual yield estimation in vineyards. *Journal of Field Robotics*, 31(5):837–860, 2014.

OpenCV Computer Vision Library, n.d. http://opencv.org/.

J. Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *International Conference on Robotics and Automation (ICRA)*, pages 3859–3866, 2012.

A. Paproki, J. Fripp, O. Salvado, X. Sirault, S. Berry, and R. Furbank. Automated 3D segmentation and analysis of cotton plants. In *Int. Conf. Digital Image Computing Techniques and Applications (DICTA)*, pages 555–560, 2011.

S. Paulin, T. Botterill, X. Chen, and R. Green. A specialised collision detector for grape vines. In *Proc. Australasian Conference on Robotics and Automation*, pages 1–5, 2015.

U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244 – 256, 1972.

Robot Operating System (ROS), n.d. http://ros.org/.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. doi: 10.1007/s11263-015-0816-y.

B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

Y. Sheikh and T. Simon. Multi-view geometry of dynamic 3D reconstruction. In *International Conference on 3D Vision*, December 2014. URL `http://www.3dimpvt.org/tutorials/`.

R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image understanding*, 58(1):23–32, 1993.

R. Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

A. Tabb. Shape from silhouette probability maps: Reconstruction of thin objects in the presence of silhouette extraction and calibration error. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 161–168, 2013.

D. Tingdahl and L. Van Gool. A public system for image based 3D model generation. In *Computer Vision/Computer Graphics Collaboration Techniques*, pages 262–273. Springer, 2011.

B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Proc. International Workshop on Vision Algorithms: Theory and Practice*, volume 1883/2000, pages 1–71, Corfu, Greece, 1999.

Vision Robotics Corporation. http://visionrobotics.com/. online, 2015. Retrieved July 2015.

Q. Wang, S. Nuske, M. Bergerman, and S. Singh. Automated crop yield estimation for apple orchards. In *Experimental Robotics*, pages 745–758, 2013.

O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2115–2128, 2009.

B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the International conference on Knowledge discovery and data mining*, pages 694–699, 2002.

Q. Zhang and F. J. Pierce. *Agricultural automation: fundamentals and practices*. CRC Press, 2013.

Y. Zheng, S. Gu, H. Edelsbrunner, C. Tomasi, and P. Benfey. Detailed reconstruction of 3D plant root shape. In *ICCV*, pages 2026–2033, 2011.

S. C. Zhu and D. Mumford. *A stochastic grammar of images*, volume 2. Now Publishers Inc, 2007.