Voxel carving for collision avoidance for a vine pruning robot arm

Tom Botterill and Richard Green
Department of Computer Science
University of Canterbury, Christchurch, NZ
tom.botterill@canterbury.ac.nz

Steven Mills
Department of Computer Science
University of Otago
Dunedin, NZ

ABSTRACT

A vine pruning robot builds a 3D model of vines using a model-based feature matching and bundle adjustment pipeline. It is important to model the entire structure of each vine in order to plan collision-free paths for a robot arm, and to generate a connected vine structure. Each vine has a complex lump of old growth known as the head, which is challenging to model within the feature-based pipeline, so we use voxel carving to find the *visual hull* of the head from many views. The visual hull is a volumetric model containing the head, so is ideal for collision avoidance.

This paper describes how the voxel carving algorithm is integrated into the 3D vine modelling system. For our application it is convenient to model the vine head as a set of spheres, so we modify a standard voxel carving pipeline to to fit a set of spheres to images of the vine head. We evaluated the system on 300 stereo images of 15 vines: the voxel carve is computationally efficient, and the models found are over 90% complete.

Categories and Subject Descriptors

Computing methodologies [**Artificial intelligence**]: Computer vision—*Vision for robotics, Shape representations, 3D imaging*; Applied computing [**Computers in other domains**]: *Agriculture*

Keywords

Voxel carving, collision avoidance, 3D modelling, agricultural robotics

1. INTRODUCTION

We are developing a robotic system for automatically pruning vines (Figure 1). The vines are imaged with three colour cameras, then a computer vision system builds a 3D model of the vines (Figure 2). An AI algorithm decides which canes to cut, a path planner plans a collision-free path to reach the cuts, then a robot arm reaches and cuts the vines. The system is mounted on a mobile platform which straddles the row of vines, so that lighting is controlled and the vines are imaged against a blue background.

Figure 1: The vine pruning robot. The robot arm and cameras are inside.

The computer vision system for building 3D models is based on a standard 3D reconstruction pipeline where feature locations are extracted from each image, a correspondence between sets of features in different frames is found, matched features are triangulated to make a 3D model, and then the model is optimised using incremental bundle adjustment [8, 17]. Building a 3D model of vines by computer vision is challenging however, because each image shows a tangled network of overlapping canes, so to reconstruct a complete and accurate model we customise each stage of the 3D reconstruction pipeline to work well for vines. The features extracted from each image are the positions of individual canes, with the knowledge that each cane is a smooth curve of uniform thickness used to guide the feature extraction [2]. The correspondence and 3D reconstruction of canes is guided by the fact that the vines have an underlying tree structure [3]. The resulting 3D model is a connected net-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. IVCNZ '14, November 19 - 21 2014, Hamilton, New Zealand Copyright is held by the owner/author(s). Publication rights licensed to ACM. http://dx.doi.org/10.1145/2683405.2683419
.

work of 3D *polylines*, a polyline is a sequence of straight line segments defined by a sequence of control points.

The head of a vine is a complex lump of wood which grows around the top of the trunk over many years (Figure 2). The system needs to model the head so that it can infer the connections between canes and the trunk, and so that the robot arm doesn't collide with it (collisions with the head and trunk could cause expensive damage to the arm). To model the head we use *voxel carving*, a method for building volumetric 3D models from many views, and which is ideal for modelling complex-shaped obstacles for collision avoidance. The following section reviews different 3D reconstruction methods, and explains why voxel carving is the method most suitable for reconstructing the vines' heads.

2. 3D RECONSTRUCTION METHODS

There are three common methods for estimating 3D models from 2D images: feature-based methods, dense-stereo methods, and voxel carving, or silhouette-based, methods. General-purpose feature-based methods for 3D reconstruction match point features (e.g. blobs described by SIFT descriptors [12]) between views, triangulate them into 3D, then use bundle adjustment to optimise the 3D structure [8, 17]. These methods generate sparse sets of 3D points from multiple views, and perform best when objects have visually distinctive features.

Another common method for 3D reconstruction is dense stereo. Images from a pair of calibrated cameras are rectified so that matching scanlines in the two images show the same objects [16]. Dense stereo methods find a *disparity map*, a dense mapping from pixels in one image to pixels on corresponding scanlines in the other image. The disparity map is optimised to jointly maximise a data term describing the similarity of matching pixels, and a smoothness term imposing regularisation constraints on the 3D structure (e.g. that the disparity map should be piecewise smooth). From the disparity map, a depth map of the surface of the 3D object visible to the cameras is computed.

```
//Initialise: Fill space with 3D grid of voxels

//Carve:
foreach frame I with camera P do
    foreach voxel X do
        Project into frame: P : X → x
        if I(x) is background then
            delete X
        end
    end
end

Result: Set of voxels defining visual hull of the
        foreground segment.
```

Algorithm 1: Overview of standard voxel-carving algorithm. Efficient implementations normally use octree representations.

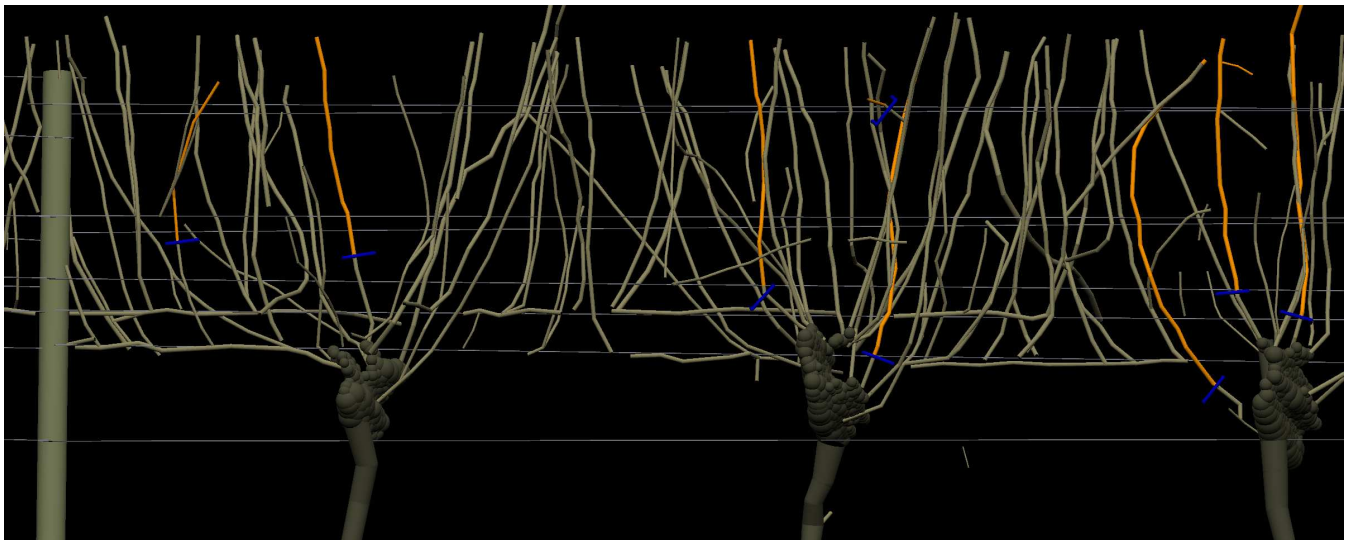
Voxel carving methods discretise 3D space into a set (e.g. a regular grid) of small 3D regions known as voxels, analogous to pixels. Initially, 3D space is filled with voxels. In its simplest form, voxel carving involves considering each view in turn, projecting every voxel into that view, then removing (carving) voxels inconsistent with that view, e.g. because they project to the background (Algorithm 1). There are numerous variations and extensions for different applications: many approaches carve voxels which are not *photo-consistent*, i.e. the voxel's colour is not consistent with the colour seen when projected into different images [10, 16], other systems use octree representations of space to reduce the number of voxels needed to model large objects [15]. An alternative approach is to approximate object boundaries in the image with polylines, and edit a mesh enclosing occupied space to match all of the boundaries [13]. All of these formulations lead to a 3D volume which encloses the actual 3D shape; this volume is known as a *visual hull* [10, 11].

Voxel carving methods work well when the foreground can easily be segmented from the background [10, 9], and with images from a large number of viewpoints. Kumar et al. [9] use voxel carving to model plants imaged against a white background. This allows the use of multiple views to resolve the shape of similar-looking branches, although accurate calibration is essential to ensure that these fine structures are not lost. Voxel carving methods can be computationally expensive if a high resolution (i.e. a large number of voxels) is required, and using knowledge of the 3D structure to regularise the reconstruction is challenging.

For our problem of modelling vines' heads, we have multiple images of each head against a blue background. Camera positions are available from the incremental bundle adjustment used to reconstruct the rest of the vines [1]. The most reliable visual information about the head is its silhouette, as much of the texture on the head is caused by occlusion by canes, shadows from canes, self-occlusions, and because lighting and shadows vary as the mobile platform moves. Reconstruction methods based on feature matching often perform poorly in conditions like these, and unlike voxel carving, provide only a sparse reconstruction, with no guarantee that the entire object is modelled. Dense stereo methods could be parametrised to work around the head regions, but it is challenging to make efficient use of multiple views, and to ensure accurate reconstruction around occlusions. In addition, dense stereo methods constrain the shape of the 3D surface (e.g. force it to be smooth [16] or piecewise fronto-parallel-planar [18]), so there is no guarantee that it will enclose the actual head, leading to a risk of collision [10]. As only the visible surface of the object is reconstructed, there is also the problem of how to avoid collisions with parts which were never viewed. The visual hull computed by voxel carving avoids both of these problems: any regions which are not observed to be obstacle free are part of the visual hull, which is exactly the property needed for computing collision free paths for a robot arm.

The intended uses of the head model are firstly, path planning for the robot arm; secondly, resolving joins between canes and the head; and thirdly, visualising the vines. An efficient representation of the head for all of these tasks is a collection of spheres, each with a position and radius (Figure 2). Spheres are ideal as input for the path planner, as computing collisions with spheres is simple, and because it is trivial to add a "safety margin" [6] around spheres by increasing their radius. Spheres are also suitable for connecting the vine structure for the same reason. Spheres produce "realistic-looking" vine models when rendered (Figure 2); important as the vine models are shown to human pruners who 'prune' digital vine models in order to label training data for

(a) One set of stereo frames from the robot's three cameras



(b) 3D vine model reconstructed from many frames

Figure 2: A vine pruning robot uses a model-based feature matching and bundle adjustment pipeline to reconstruct vines. The models are used for deciding which canes to prune (highlighted in orange), and for planning collision-free paths for the robot arm. The complex head regions are challenging to model, so we use voxel carving to fit volumetric models.

the AI [5]. In the following section we describe our modification to voxel carving so that the resulting model is a set of spheres.

3. VOXEL CARVING FOR VINE HEAD MODELLING

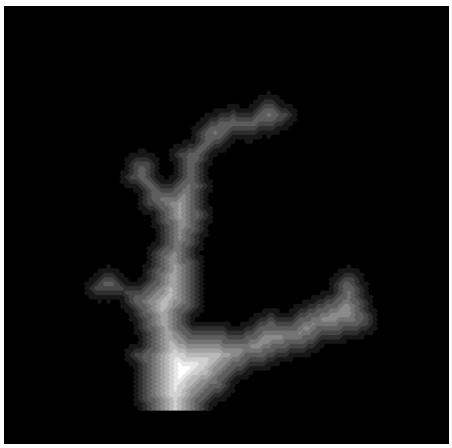
The robot arm and cameras are mounted on a mobile platform which completely straddles the row of vines, so that the vines are viewed against a blue background (Figure 3a). The foreground vine pixels are segmented from the background, then the 2D structure of the canes and trunks are found in each image [2]. The head is the large region of foreground pixels which is not explained by the canes or trunk. We identify this region by masking out the vines and trunk, then applying morphological opening [4, 16], a standard computer vision technique which removes small features (remaining canes and junk) from the images (Figure 3b).

Modifying voxel carving to fit a set of spheres ('spherical voxels') to the head regions is simple: initialise 3D spheres throughout a volume of space, then for every background pixel in every image, intersect its ray with every sphere. If the ray intersects the sphere reduce the sphere's radius (Algorithm 2). This approach is too inefficient for practical use, as it has complexity $O(INV)$, where I is the number of images, N the number of pixels in each image, and V the number of voxels.

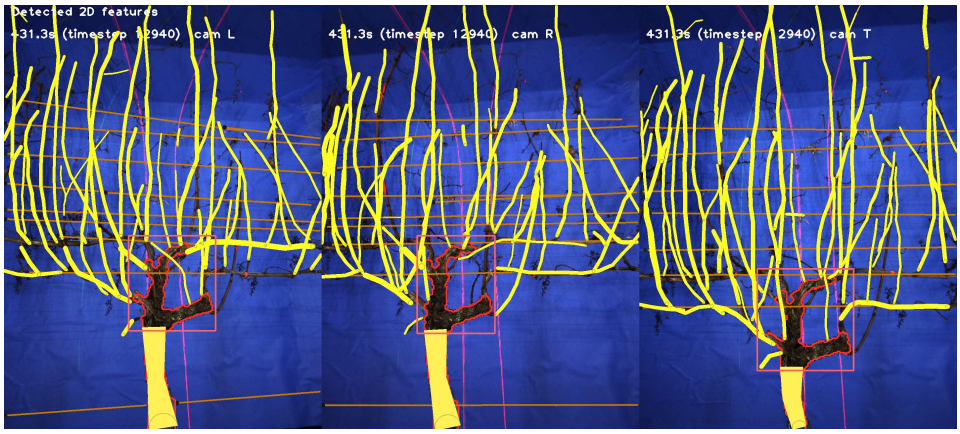
We use the fact that spheres project to circles to reduce the complexity of the spherical-voxel carve, as outlined in Algorithm 3. First, we compute a *distance transform* (Figure 3c) of the each foreground/background segmented image. The distance transform, D, is an image where every pixel value is the minimum Euclidean distance to the nearest background pixel in the original image, and it can be computed in time linear in the number of pixels [7]. Each spherical voxel $(\mathbf{X}, R)$ is projected to a 2D point and radius

(a) Vine head region in original image

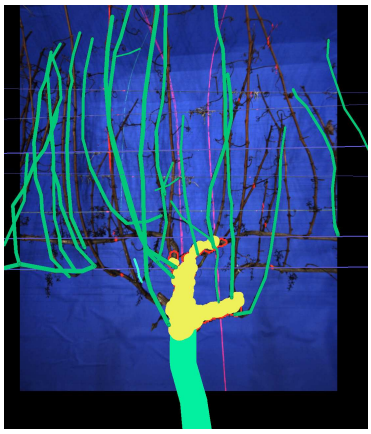
(b) Vine head region identified by morphological opening



(c) Distance transform of head region



(d) 2D canes and trunks (yellow) and wires (orange) are extracted from every frame, and the head regions are identified (red)



(e) 3D model backprojected onto 2D frame

Figure 3: Standard 2D image processing methods are used to identify the head region in each image.

```
//Initialise spheres:
Fill space with spheres centred at regular 3D grid.

//Carve:
foreach frame I with camera P do
    foreach background pixel x do
        foreach voxel (X, R) do
            //R(x) is the ray through x
            //Compute ray-voxel distance d:
            d ← distance(R(x), X)
            R ← min(R, d)
        end
    end
end

Result: Set of spheres defining visual hull of the
        foreground segment.
```

Algorithm 2: Modification of standard voxel-carving algorithm to fit spheres (simple but inefficient).

$(\mathbf{x}, r)$ in each image. The distance transform $D(\mathbf{x})$ gives the closest distance to the background, and forward-projecting this distance back to the spherical voxel's centre gives the distance between the spherical voxel and the closest ray from this image. As previously, if the distance is less than the sphere's radius then we reduce this radius. This approach has complexity $O(IN + IV)$ for the distance transforms and voxel carves respectively.

Two further optimisations reduce the computational cost considerably, but do not reduce the complexity. Firstly, we initialise only the voxels consistent with one image: voxels are initialised along a ray through each foreground pixel at a suitable range of depths, with radii given by the distance transform. Secondly, we remove voxels smaller than a threshold.

The region defined by the set of voxels has the property that every sphere meets the outside of the region (this is true when voxels are initialised because they meet the ray through the nearest background pixel, then remains unchanged when either the radius is reduced, or other voxels are deleted). However, most small spheres are almost entirely contained within a larger sphere. These can be deleted

foreach *frame* **do**
  Identify head region (morphological opening)
  Compute distance transform D
end

//Initialise spheres from one image I with camera P:
foreach *pixel* $\mathbf{x}$ *where* $D(\mathbf{x}) > 0$ **do**
  for *depths* d *in* $d_{min}...d_{max}$ **do**
    Initialise new spherical voxels along ray
    $(\mathbf{X}, R) \leftarrow P^{-1}(\mathbf{x}, D(\mathbf{x}), d)$
  end
end

//Carve:
foreach *frame with camera projection matrix* P **do**
  foreach *spherical voxel* $(\mathbf{X}, R)$ **do**
    Project spherical voxel into frame:
    $P : (\mathbf{X}, R) \rightarrow (\mathbf{x}, r)$
    if $r < D(\mathbf{x})$ **then**
      $R \leftarrow P^{-1}(D(\mathbf{x}))$
    end
  end
end

//Model simplification heuristic:
foreach *spherical voxel* $(\mathbf{X}_1, R_1)$ **do**
  foreach *smaller* $(\mathbf{X}_2, R_2)$ **do**
    if $\|\mathbf{X}_1 - \mathbf{X}_2\| + R_2 - R_1 > thresh$ **then**
      Delete $(\mathbf{X}_2, R_2)$
    end
  end
end

Result: Set of spheres defining visual hull of the foreground segment.

Algorithm 3: Overview of sphere-carving algorithm.

with minimal change to the 3D model (worst case complexity $O(V^2)$ but fast in practice).

4. RESULTS

We integrated the spherical voxel carve into the vine pruning robot's 3D vine reconstruction system. The system runs online and asynchronously, with each frame captured after the previous frame has been processed. Currently the system takes 600ms per frame, corresponding to a robot speed of up to 0.1m/s. The voxel carve takes an average of 20ms per frame, and runs concurrently with other processes, so does not increase the computational cost of the system substantially. 3D models containing both vines and spherical-voxel head models are shown in Figure 2.

We ran the system on 300 images of *Sauvignon Blanc* vines, reconstructing a total of 15 head models. We use a 10mm voxel resolution, so that on average 7142 voxels are initialised and 2090 remain after voxel carving. Figure 4 shows how the volume, and number of voxels, decrease as more images are used in the carve. Each head models is reconstructed from an average of 10.3 viewpoints, with a range of 6 to 17. The volume of the reconstruction and number of voxels falls as more viewpoints are used to carve voxels, and as expected, most voxels are removed after the first few viewpoints.

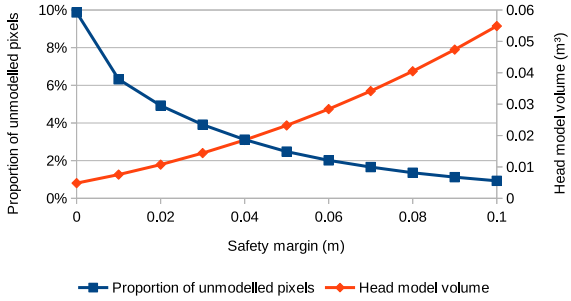


Figure 5: Increasing the 'safety margin' around the head region ensures that it covers all head pixels.

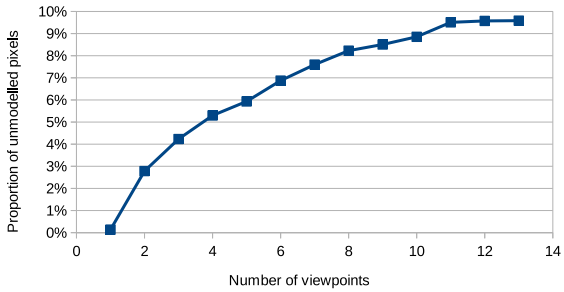


Figure 6: The average number of foreground pixels missed increases as the number of viewpoints increases, due to errors in camera pose.

The heuristic model simplification (deleting spheres almost entirely within others, with a 5mm threshold) reduces the number of spheres from an average of 2090 to an average of 284, while reducing the model's volume by only 2.0%. The main sources of error are quantisation errors, errors from model simplification, errors in segmenting the head model, and errors in camera pose. All of these errors can cause the model to be smaller than the actual head region. A measure of these errors is the proportion of head pixels which are not explained by the model ('unmodelled pixels'). On average, 9.8% of head pixels lie outside the projection of the head model. Without a heuristic model simplification this is 5.9%, indicating that the model simplification is an area for future improvement. For path planning, we add a safety margin to all models to minimise the chance of collisions. Figure 5 shows that safety margin of 50mm reduces the proportion of unmodelled pixels to 2.5%, but with a 380% increase in model volume. Figure 4 shows that the average number of missing pixels increases as more viewpoints are used, due to variations in segmentation and errors in camera pose. Figure 4 showed that the volume decreases slowly after the first few models are added; together these figures suggest that a safe (and efficient) strategy for avoiding incomplete models is to use a smaller subset of all available viewpoints for each head model. We also plan to investigate more robust voxel carving criteria, e.g. reducing sphere size to avoid the nth closest background pixel rather than the closest, which has the potential to mitigate all of these sources of error.

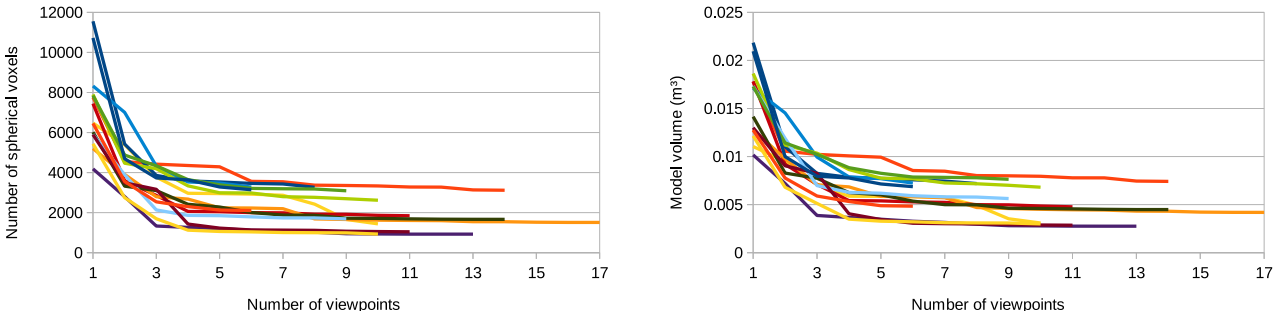


Figure 4: 15 head models (different colours) are reconstructed from between 6 and 17 viewpoints each. The number of voxels, and volume of the head model, fall as more images are added.

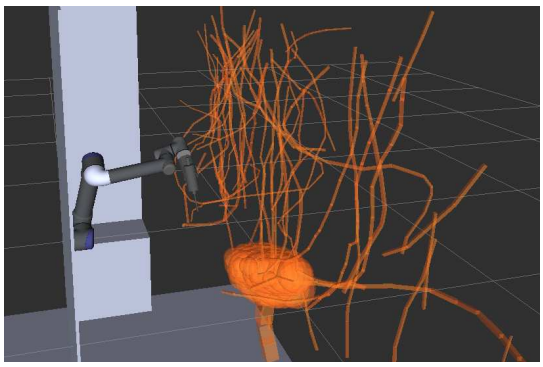


Figure 7: The ROS MoveIt path planner's visualisation of the 3D head model and robot arm.

The 3D vine and head models are used to plan paths for a six degree-of-freedom robot arm, using the MoveIt path planner library in Robot Operating System (ROS) [14]. Figure 7 shows ROS's visualisation of the robot arm and vines.

5. CONCLUSIONS

This paper described how voxel carving is used by a vine pruning robot to model complex 'head' parts of vines, in order to generate complete 3D models for robot collision avoidance. Voxel carving is ideal for this application as it uses multiple views to fit a volumetric model which contains the vine head (a 'visual hull').

For our application, it is convenient to model the head as a set of spheres, so we modify a standard voxel carving pipeline to do this. The system is evaluated on 300 stereo images of 15 vines: the 3D models generated are over 90% compatible with the images of the vines, and for path planning this can be increased to 97.5% with a 50mm safety margin. The voxel carve takes 20ms to fit a head model, compared with 600ms per frame for the rest of the vision pipeline.

6. REFERENCES

[1] T. Botterill, R. Green, and S. Mills. Reconstructing partially visible models using stereo vision, structured light, and the g2o framework. In *Proc. IVCNZ*, 2012.

[2] T. Botterill, R. Green, and S. Mills. Finding a vine's structure by bottom-up parsing of cane edges. In *Proc. IVCNZ*, pages 1–6, 2013.

[3] T. Botterill, R. Green, and S. Mills. A decision theoretic formulation for sparse stereo correspondence problems. In *To appear in Proc. 3DV*, 2014.

[4] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[5] S. Corbett-Davies, T. Botterill, R. Green, and V. Saxton. An expert system for automatically pruning vines. In *Proc. IVCNZ*, 2012.

[6] F. Fahimi. *Autonomous Robots*. Springer, 2009.

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell University, 2004.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. CUP, second edition, 2003.

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

[10] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *IJCV*, 38(3):199–218, 2000.

[11] A. Laurentini. The visual hull concept for silhouette-based image understanding. *T. PAMI*, 16(2):150–162, 1994.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[13] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proc. Conf. Computer graphics and interactive techniques*, pages 369–374, 2000.

[14] I. A. Sucan and S. Chitta. Robot Operating System (ROS) MoveIt path planning library. Retrieved September 2014.

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

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

[17] 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.

[18] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *Trans. PAMI*, 31(12):2115–2128, 2009.