Australian Government
Department of Defence
Defence Science and
Technology Organisation

3D Point Cloud Tree Modelling

Mark Graham and Adam Davies

Intelligence, Surveillance and Reconnaissance Division
Defence Science and Technology Organisation

DSTO-TN-0954

ABSTRACT

Three-dimensional (3D) point cloud models of trees were developed using the software packages Maya and MATLAB. These models are to be assembled into a forest scene that can be used for producing simulated Laser Detection and Ranging (LADAR) imagery of objects obscured beneath the canopy. This document describes the approach for modelling the trees using Maya then importing and completely reconstructing the models in a custom 3D point cloud format in MATLAB.

Published by

Intelligence, Surveillance and Reconnaissance Division
DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 7389 5555
Fax: (08) 7389 6567

AR-014-775
June 2010

3D Point Cloud Tree Modelling

Executive Summary

This document provides an overview of the development of three-dimensional (3D) point cloud models of trees using the software packages Maya and MATLAB. This work was undertaken as part of Task 07 / 105 *Support to Intelligence, Surveillance and Reconnaissance* to investigate the tasking of airborne sensors to detect inconspicuous targets, particularly those partially obscured by forest canopies. An important aspect of this task was to develop a representation of a forest scene that would enable the modelling and simulation of an airborne foliage-penetrating Laser Detection and Ranging (LADAR) system.

The aim of this work was to produce 3D models of Australian forest trees that can then be easily assembled into forest scenes. The forest models and models of target vehicles generated using the same procedure will be used to simulate LADAR imagery in later studies to assist in making recommendations for the military operational use of an airborne foliage-penetrating LADAR imaging system.

The trees were based on photographs from a study site in central Queensland, together with survey data from woodlands near the RAAF Tindal base in the Northern Territory. No further validation of individual tree models was conducted.

Contents

1. Introduction

Tree models were developed and assembled into a representation of a real world forest scene that could subsequently be used for simulating the imagery that is produced by a foliage-penetrating Laser Detection and Ranging (LADAR) system.

The computer graphics and modelling software package Maya can be used for generating tree models, assembling them into a forest scene, and also has the capability to illuminate the sections in the scene that would represent the imagery captured by a single snapshot of a LADAR system. Maya cannot be used to isolate and display only the illuminated sections, nor merge multiple results captured from different viewing angles. Therefore it was decided to export the Maya tree models to MATLAB and then develop a custom software tool for simulating foliage-penetrating LADAR imagery of objects obscured beneath a forest canopy.

This document outlines an approach for modelling the trees using Maya, then importing and completely reconstructing the models in a three-dimensional (3D) point cloud format in MATLAB. This type of model provides a way for representing real world 3D objects, such as trees, with a high level of detail. It is simply a set of voxels (volume elements in 3D space) that describes the visible surface of objects. By producing a 3D point cloud representation of a forest scene in MATLAB, where each point is the centre of a voxel, a simple ray tracing algorithm could be developed for determining the voxels that are within direct line-of-sight to a LADAR sensor.

2. Tree Modelling in Maya

2.1 Development of the tree models

Maya was used to generate the original tree models that were representative of the Australian environment. The dimensions of the tree models were based on a survey of woodlands in the vicinity of RAAF Tindal, Northern Territory [1], while the overall appearance was based on photos of poplar box (*Eucalyptus populnea*) from woodlands in the Injune Landscape Study Site in central Queensland [2].

The tree models (Figure 1) were developed with a subset of Maya's 3D paint effects called 'tubes'. Tubes are able to emulate the creation, growth and behaviour of plant life, controlled by a comprehensive set of parameters that define the shape and growth behaviour of branches, twigs and leaves. The trees were initially based on an example model of a medium-sized birch tree, which was modified to create a set of models resembling poplar box trees.

These tree models could be directly assembled into a forest scene in Maya. With the use of lighting effects, the sections of the forest scene representing the imagery captured by a single snapshot of a LADAR system could be illuminated. While this was the logical solution to implement for modelling a foliage-penetrating LADAR sensor, it was not possible to use Maya to display only the illuminated sections (which represent the simulated LADAR image), combine multiple results from different viewing angles, or to conduct further analysis on the results. It was therefore decided to produce a 3D point cloud model of the Maya trees in MATLAB and develop a customised model specifically for simulating a LADAR sensor to overcome these issues. This approach should also enable real world forest data stored in a point cloud format to be used in the modelling and simulation.

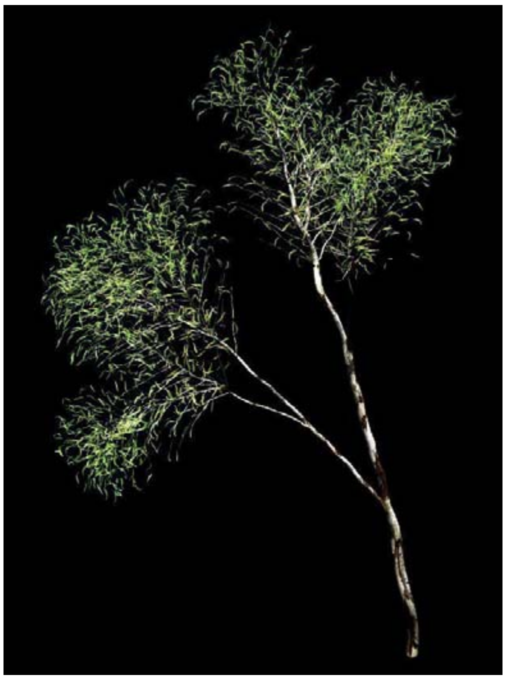


Figure 1: Rendered image of a tree model produced in Maya

2.2 Extracting and exporting the models

To export the Maya models to MATLAB for further manipulation, a procedure was developed which made use of Z-buffer images. The Z-buffer is the component of Maya's native output format (Interchange File Format) which indicates furthest visible depth.

The Z-buffer was extracted and manipulated to export to MATLAB via a plug-in module developed using C++ libraries that are part of the Maya package (see Appendix A for some details). This depth information was saved in a text file that could be easily exported to MATLAB. These text files contain positive floating point values representing distances from a user-defined reference plane (located in front of the camera) to pixels in the rendered scene. The information in this text file can be plotted to produce a depth image of the object (Figure *2*). The pixel resolution of the depth image is limited by the graphics and processing capabilities of the computer. These images were captured at a resolution of 1280 × 1024 pixels with an orthographic projection so that perspective distortion was not introduced.

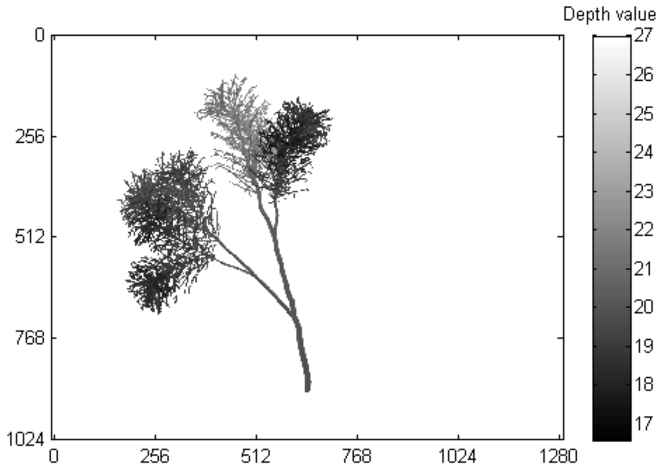


Figure 2: Depth image of the tree in Figure 1. The greyscale indicates the depth value associated with each pixel, where black is closest to the camera and white is furthest away.

Since the depth image from a single viewpoint of the tree does not provide sufficient information to reconstruct a complete 3D point cloud model, it was necessary to capture and combine depth images from multiple views around the tree. For each tree model, a series of four depth images were captured at 90° intervals around the tree (Figure 3). For completeness a fifth depth image was also captured from above the tree to provide a top-down view.

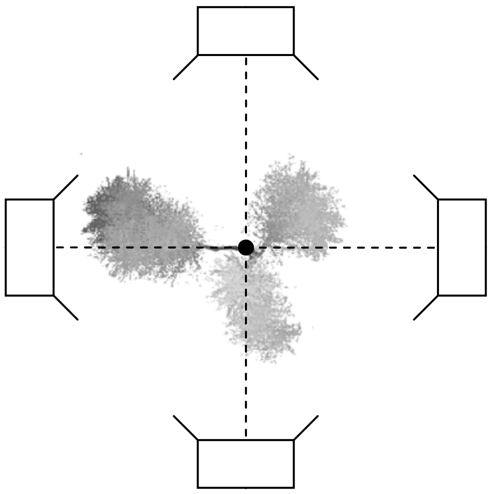


Figure 3: Top view layout of the camera positions for capturing depth images of a tree model. The pivot point was located at the base of the trunk.

A consequence of creating 3D point cloud models is that quantisation errors are introduced such that the model in MATLAB is a slightly coarser representation of the original Maya model. This error could be reduced if a higher resolution depth image was used, but producing such an image would require a computer with greater graphics and processing capabilities.

The distance from the camera to the pivot point, located at the base of the trunk, was the same for all views of each tree and set to the shortest distance such that the largest tree model was contained in the field of view of the camera. This provided the highest-possible resolution image while maintaining a constant distance between the camera and pivot point for all trees.

3. Reconstructing the Tree Models in MATLAB

3.1 Reading and formatting the model images

The text files containing the depth information about the image were read into a two-dimensional (2D) array in MATLAB. The row and column coordinates of each element in this matrix correspond to the position of the pixel in the depth image and its value represents the distance from a user-defined reference plane located in front of the camera (depth value). Based on images of simple geometric test objects, these depth values appear to be evenly spaced. By default, a depth value of 10 000 represents empty space. This value is significantly larger than any value representing a pixel that exists on the tree model. Therefore a search of this matrix was performed to find the row and column coordinates of all depth values less than 10 000. This search returns a list of row, column and depth value coordinates for the tree model, hereafter referred to as (r,c,d) coordinates. Each tree model was formed using the four sets of (r,c,d) coordinates that are generated from the four depth images captured around each tree. The top view was not used in this stage.

3.2 Transforming the depth information in the images

The 3D point cloud model consists of integer (x,y,z) coordinates defining the centre of each voxel. But since the depth images do not contain any information about the size of the pixels in terms of depth values, the sets of (r,c,d) coordinates for each tree model cannot be directly mapped to (x,y,z) coordinates. It is therefore necessary to transform the depth values by applying a conversion factor (which is determined from the depth image resolution) and then round to integer values to generate the (x,y,z) coordinates which represent the voxels in 3D space.

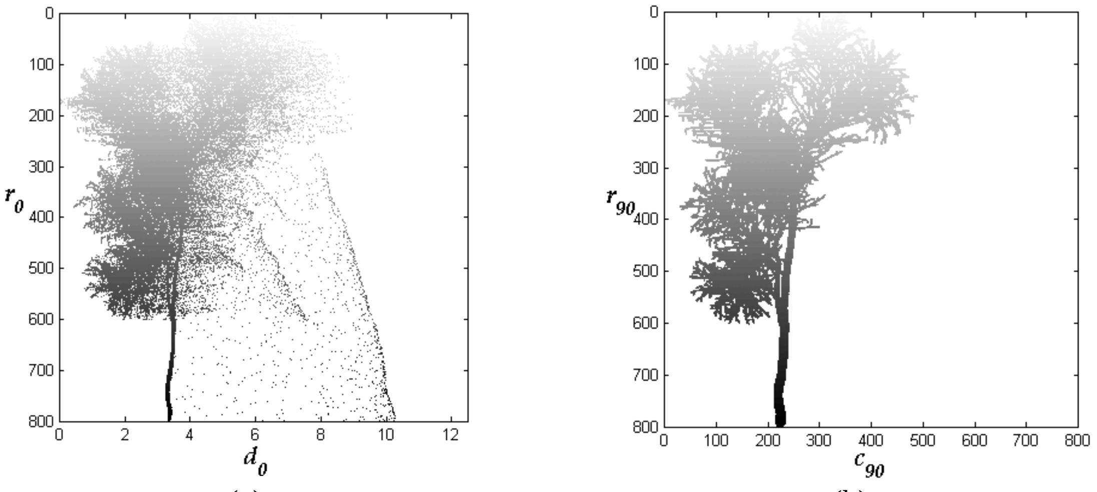
Each depth image was captured at a resolution of 1280 × 1024. Combining four depth images taken at 90° intervals at a fixed distance from a pivot point meant that the tree would be contained inside a space with dimensions of 1280 × 1024 × 1280 voxels. At this resolution, the distance in 3D space from the camera to the pivot point is 1280 / 2 = 640 voxels. To determine the conversion factor to apply to the depth values, let P be the distance from the camera to the pivot point in the depth image, then the depth conversion factor is simply: $D = 640 / P$. However P is not known in terms of depth value units and the depth conversion factor D instead needed to be determined from the depth images.

The depth conversion factor could be determined from any two of the depth images taken of the tree models where the position of the camera was separated by exactly 90°, e.g. the 0° and 90° views of the tree model. For a given row $r = \alpha$ in these two views, the pixel with the minimum depth value in the 0° view, $d_{0,r=\alpha}$, should align with the pixel with the minimum column coordinate in the 90° view, $c_{90,r=\alpha}$ (Figure 4). Therefore the depth conversion factor D for row α is determined as:

$$D(\alpha) = \min(c_{90,r=\alpha}) / \min(d_{0,r=\alpha}) \qquad (1)$$

$D(\alpha)$ was computed for multiple rows and an average taken. Once this average $\overline{D}$ was determined for the tree model, the depth values were multiplied by $\overline{D}$ and rounded to integer values, hereafter referred to as d' coordinates.

Correct alignment of every voxel in the two views is not possible due to an issue with the depth information extracted from Z-buffer channel in the Maya IFF images. This problem is illustrated in Figure 4a where there appeared to be erroneous or misaligned depth values present in the depth images. This issue was ignored at this point and dealt with after an initial 3D point cloud model of the tree was produced (§4.2) since it did not affect the process for determining the depth conversion nor the mapping, rotation and translation of the image reconstruction process presented in §3.3.



(a) (b)

Figure 4: (a) Plot of the (r,d) coordinates of the tree model for the 0° view. (b) Plot of the (r,c) coordinates of the tree model for the 90° view. The units of d (depth values) are different to r and c (pixel positions of the depth image). Ignoring erroneous values, if d_0 is scaled correctly then all voxels in the left plot should align with a corresponding voxel in the right plot.

3.3 Mapping, rotation and translation

The sets of (r,c,d') coordinates from the depth images for the four views taken from around the tree model were each mapped to (x,y,z) coordinates and then rotated and translated to form an overall 3D point cloud that describes the tree model.

Mapping each set of (r,c,d') coordinates onto the 3D space in MATLAB was quite straightforward. For a 3D image, when viewed from the front with a 90° vertical rotation, the horizontal axis is *x*, the axis into the screen is *y* and the vertical axis is *z*. Therefore mapping from (r,c,d') coordinates to (x,y,z) coordinates for each view, ignoring rotation and translation operations which were performed subsequently, is:

$$\begin{aligned} &d' \rightarrow x \\ &c \rightarrow y \\ &R_h - r \rightarrow z \end{aligned} \tag{2}$$

where R_h is the horizontal resolution of the depth image ($R_h = 1024$ for an image with a resolution of 1280 × 1024).

Following mapping onto the 3D space, each set of (x,y,z) coordinates were rotated about the *z* axis so that the four images taken from around the tree model could be merged together. The (x,y,z) coordinates were rotated anticlockwise about the *z* axis through camera position θ:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3}$$

The process of merging the four views of the tree model together (Figure 5) was completed by shifting the *x* and *y* coordinates:

$$x'_\phi = x_\phi - \min(x_\phi), \phi = \{0°, 90°, 180°, 270°\} \tag{4}$$

$$y'_\phi = y_\phi - \min(y_\phi), \phi = \{0°, 90°, 180°, 270°\} \tag{5}$$

$$y''_{180} = y'_{180} + \max(y'_{90}) \tag{6}$$

$$x''_{270} = x'_{270} + \max(x'_0) \tag{7}$$

For each of the four views, the data along all three axes was shifted so that the minimum value in each direction is zero (Eqns 4 and 5). The data from the 0° and 90° views should now align correctly since the closest voxel to the camera in the 0° view, $\min(x_0)$, will merge with the minimum y value in the 90° view, $\min(y_{90})$. Additional translations were required for correct alignment of the 180° and 270° views. For the 180° view, Equation (6) was applied so that the closest voxel to the camera, $\min(y_{180})$, aligns with the maximum *y* value in the 90° view, $\max(y_{90})$. For the 270° view, Equation (7) was applied so that the closest voxel to the camera, $\min(x_{270})$, aligns with the maximum *x* value in the 0° view, $\max(x_0)$.

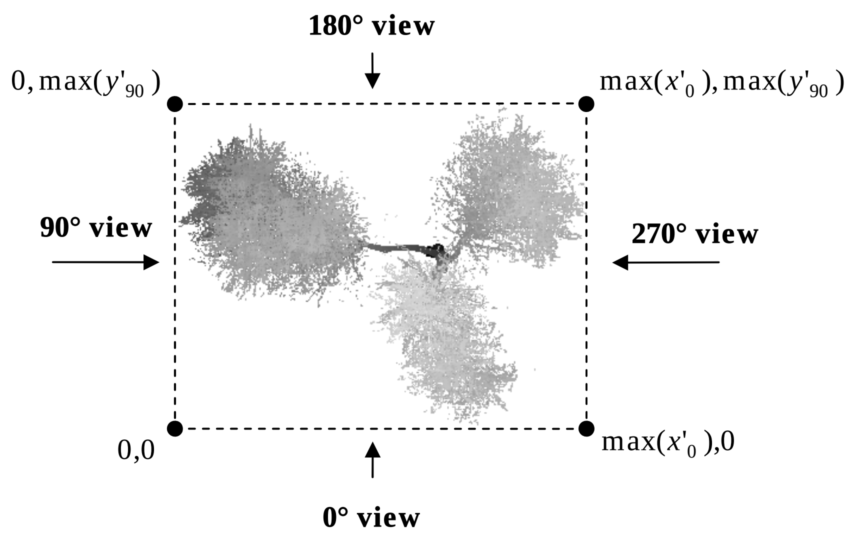


Figure 5: Translating the x and y coordinates to align and merge the four views of the tree model

The mapped, rotated and translated (x,y,z) coordinates for each of the four images were stored in a single list which describes the 3D point cloud representation of the tree model (Figure 6). If the reconstruction of the tree model was accurate then this list of voxels would contain many duplicates, since there is significant overlap when the four views of the tree are merged together. At this point this list is filtered to remove any duplicate voxels, i.e. multiple instance of a voxel with the same (x,y,z) coordinates.

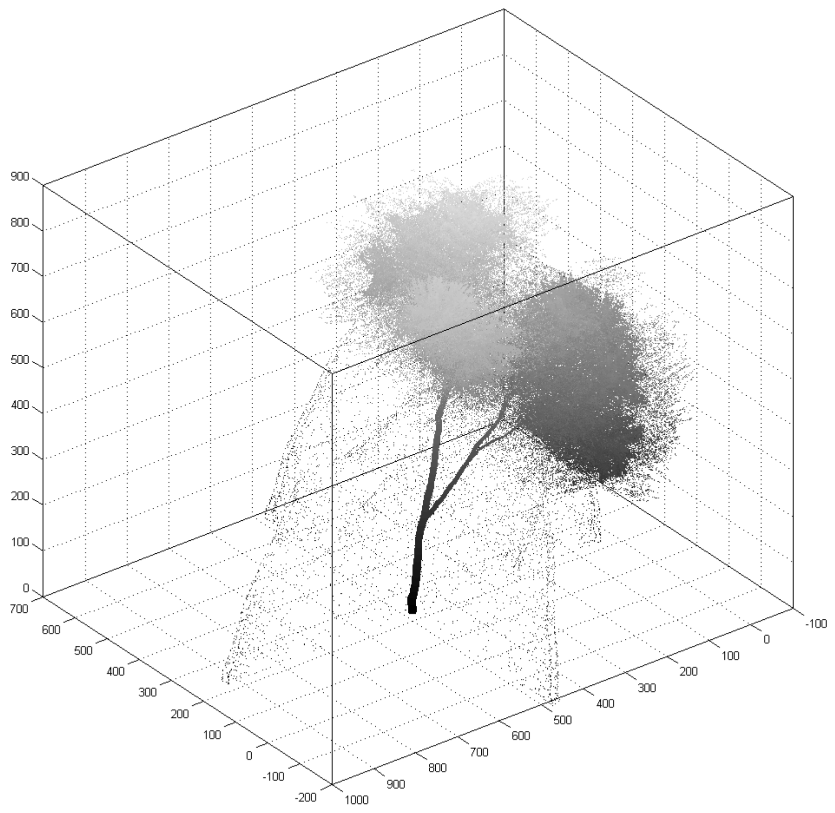


Figure 6: 3D point cloud model of the Maya tree shown in Figure 1 reconstructed in MATLAB

4. Improving the 3D point cloud model

4.1 Problems with the Maya model data

The initial reconstructed 3D point cloud model of the tree (Figure 6) had a significant proportion of voxels that were incorrectly aligned. This appeared to predominantly occur around the trunk and branches and created a spray-like effect in the 3D plot. While it was initially thought that this issue was caused by the process to transform the depth information (§3.2), further analysis revealed that this issue had been caused when images were rendered in Maya. This problem was not encountered with simple geometric test objects such as cubes or spheres, or with models of vehicles.

Figure 7 shows the unmodified depth image for the 0° view of the tree model where the background has been plotted in black. The white pixels that form an outline around the trunk and branches are depth values that are significantly further away from the camera and responsible for creating the spray-like effect observed in Figure 4a and Figure 6.

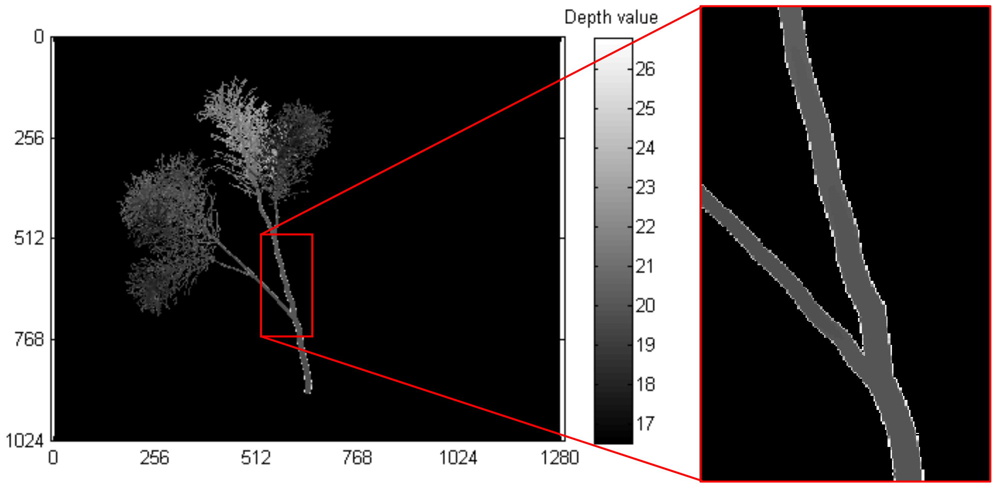


Figure 7: A depth image of the tree shown in Figure 1 with zoomed in section of the first fork. The background is plotted in black to show the erroneous data (1-pixel wide white outline) around the trunk and branches. The greyscale indicates the depth value associated with each pixel of the tree, where black is closest to the camera and white is furthest away.

At present, a solution for solving this issue in Maya has not been found. The erroneous depth values seem to vary from anywhere between the correct depth and some row-dependent maximum. Therefore it is not possible to easily remove them by applying a threshold to limit the depth values. The best-case scenario would be to relocate the erroneous depth values by estimating the correct values, but this is not possible in the crown of the tree which is expected to contain large discontinuities. Instead a template masking technique was developed in MATLAB to remove a significant proportion of the erroneous voxels.

4.2 Removing erroneous voxels from the model

The technique developed for removing the voxels in the MATLAB 3D point cloud model used the information contained in the depth images to perform template masking that removes the majority of the erroneous voxels.

Template masks were defined by extracting the (r,c) coordinates from the depth images of the 0° and 90° views around the tree and the top view of the tree looking directly down. These coordinates were mapped to sets of (y_0, z_0), (x_{90}, z_{90}) and (x_{top}, y_{top}) coordinates:

$$\begin{aligned} c_0 &\to y_0 \\ R_h - r_0 &\to z_0 \end{aligned} \tag{8}$$

$$\begin{aligned} c_{90} &\to x_{90} \\ R_h - r_{90} &\to z_{90} \end{aligned} \tag{9}$$

$$\begin{aligned} c_{top} &\to x_{top} \\ R_h - r_{top} &\to y_{top} \end{aligned} \tag{10}$$

where R_h is the horizontal resolution of the depth image.

Any voxels in the set of (x_m, y_m, z_m) coordinates describing the 3D point cloud representation of the tree model that are outside of the template masks, defined by the sets of (y_0, z_0), (x_{90}, z_{90}) and (x_{top}, y_{top}) coordinate pairs, were removed (Figure 8).

To remove any additional erroneous voxels from within the crown of the tree that were missed by the template masking, an isolated location test was performed. This identified and removed any voxels from the model that were isolated, i.e. a voxel that has no immediate neighbours. Isolated voxels should not exist in the tree model and were therefore deemed to be erroneous.

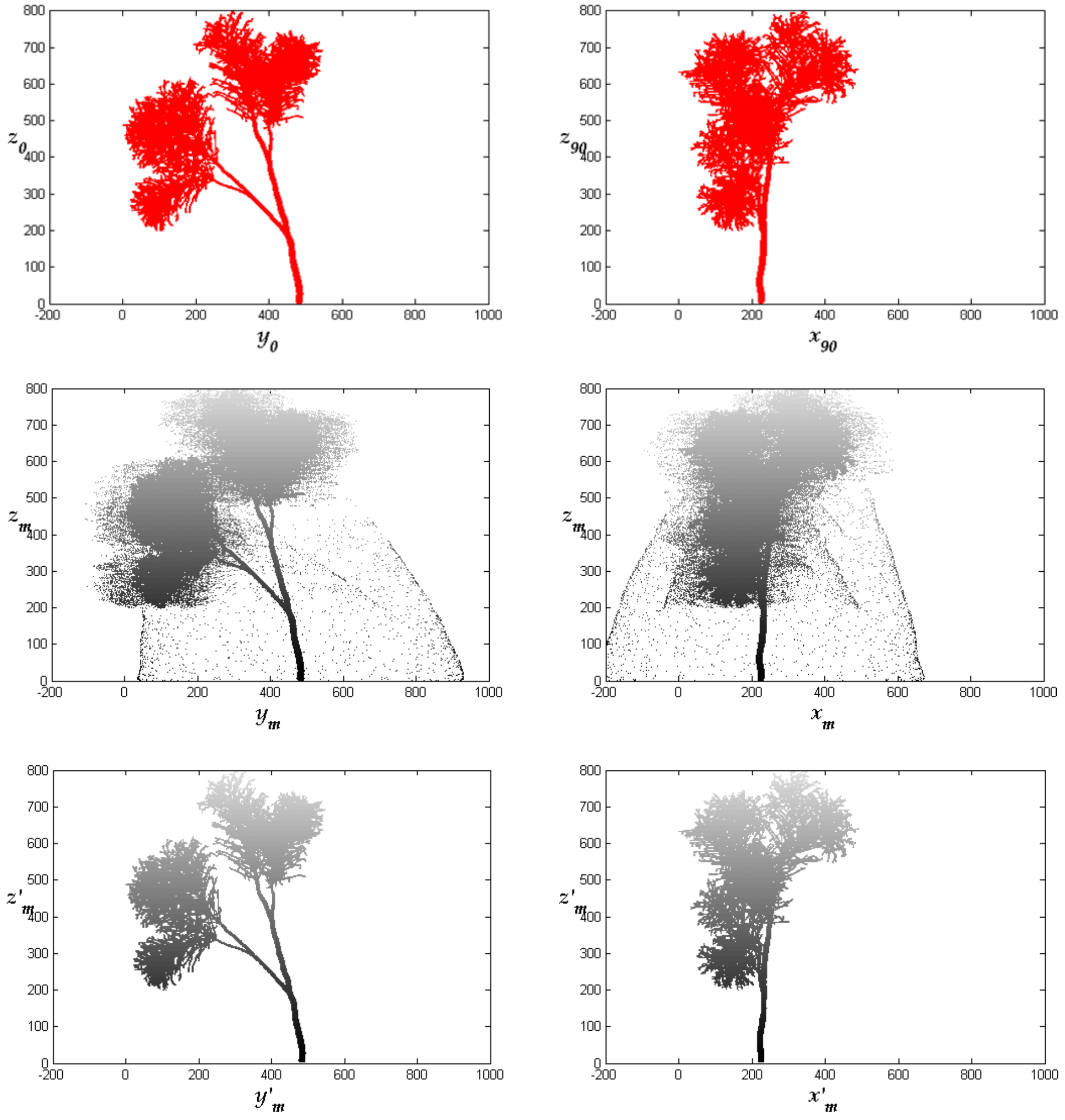


Figure 8: The process for removing erroneous voxels from the 3D point cloud model of the tree. The template images (top) were matched to the unfiltered images of the tree (middle) and non-matching voxels removed (bottom).

4.3 Interpolating missing voxels in the trunk and branches

An unfortunate consequence of the techniques used to reconstruct 3D point cloud models of trees in MATLAB was that several sources of error were introduced resulting in a degradation of the quality of the model. In particular, removing erroneous voxels from the model rather than relocating the voxels to their correct position could potentially cause gaps to appear in the trunk and branches of the tree where the different views should overlap.

Linear interpolation was used to fill in any gaps that existed in the tree trunk or branches from the base of the tree up to the crown. The interpolation was performed on adjacent voxels in every layer (the *x*-*y* plane at each *z* coordinate) of the tree model. For the tree trunk, voxels on each layer were ordered in a clockwise direction. The linear interpolation was performed between pairs of voxels on this ordered list and the algorithm generated all of the missing locations in the trunk on that layer.

However above the first branching, voxels that form each branch must be identified and the interpolation performed as described above but on the branches separately. This process was semi-automated by using the `clusterdata` function in the MATLAB Statistics Toolbox, which performs hierarchical clustering and worked effectively when unique branches were not in close proximity.

Figure 9 illustrates this process performed on a single layer of the tree. In this example, clustering was used to identify the two unique branches that are present on this layer of the tree.

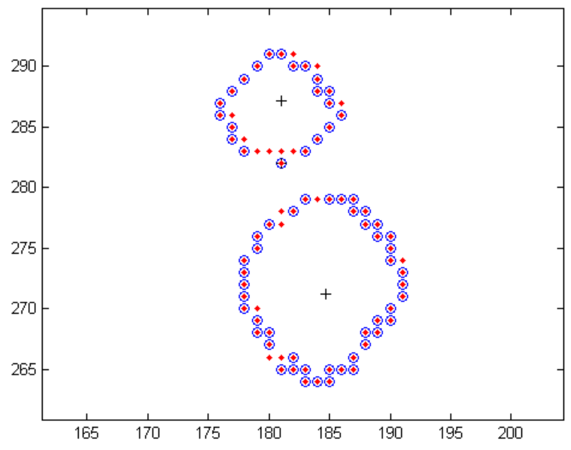


Figure 9: Linear interpolation of missing voxels in the branches on a single layer of the tree model. The blue circles are the original voxels and the red dots are the interpolated voxels.

It is currently not possible to interpolate the tree trunk or branches within the crown of the tree in an automated way due to the difficultly in distinguishing between voxels that form branches or leaves. A manual process could be used to visually check and add any missing voxels to the branches in a layer within the crown of the tree. This is a very tedious and time consuming process that was not undertaken because the improvement in the model was expected to be minimal.

4.4 The final model

The final 3D point cloud model of the tree (Figure 10) was saved to a file for future use. It is also important to note that while this document has focused on the development of 3D point cloud representations of trees in MATLAB, it is feasible that these techniques could be applied to other types of models.

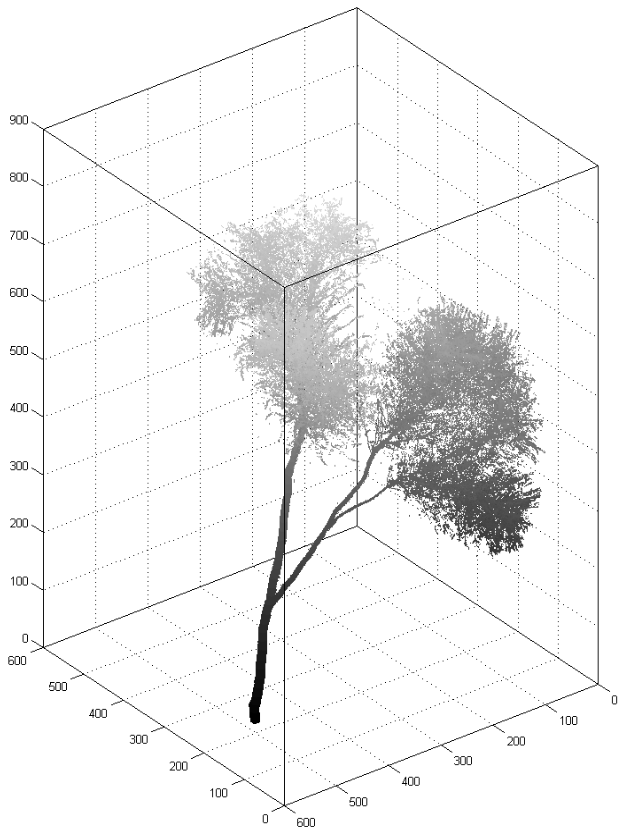


Figure 10: Final MATLAB 3D point cloud model of the Maya tree shown in Figure 1

5. Forest Modelling

5.1 Assembly into a forest scene

A forest scene (Figure 11) was produced using a variety of tree models that were reconstructed in MATLAB using the procedures described in this document. Individual 3D point cloud tree models were placed into the forest scene by shifting their *x* and *y* coordinates such that each tree is located in a desired position. Care needed to be taken to ensure that the trees didn't overlap. Before each shifted tree model was added to an overall list of voxels describing the 3D point cloud forest scene, a visual check was used to confirm it was located correctly. This was done by generating a plot showing the new tree in one colour and all other previously added trees in a different colour.

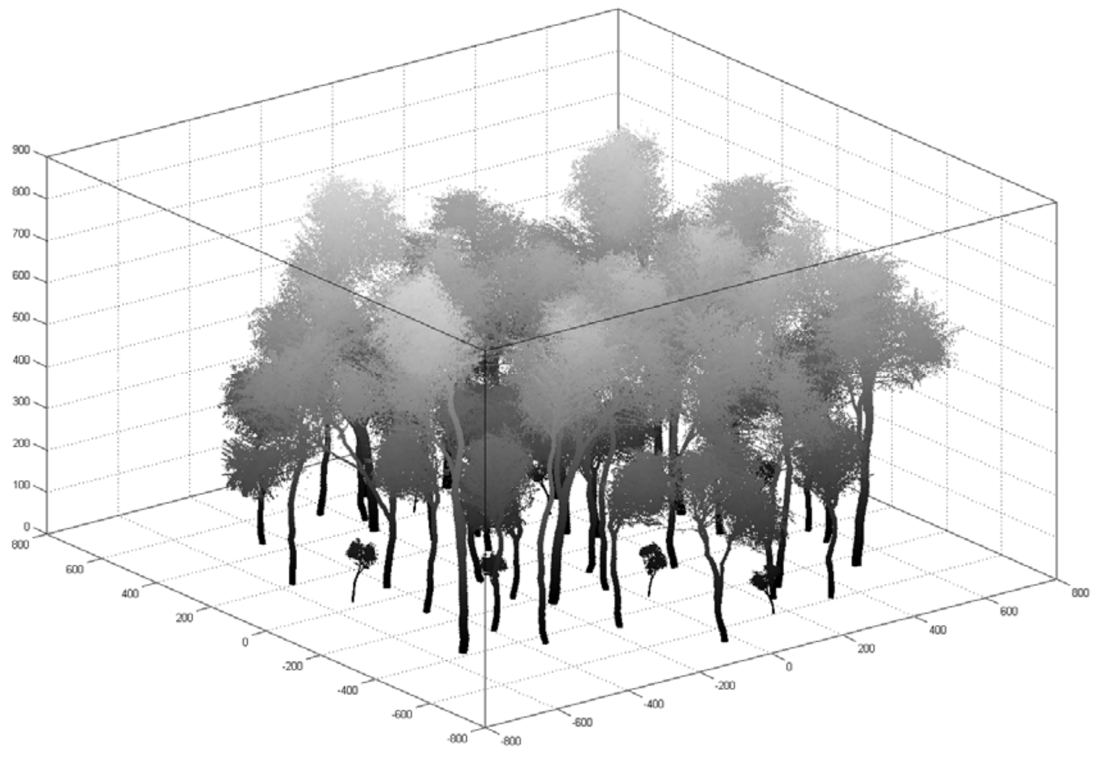


Figure 11: 3D point cloud model of a forest scene generated in MATLAB

5.2 Assessing the forest model

The 3D point cloud representation of the forest scene (Figure 11) had characteristics of an 'open forest', i.e. 10–30 m high trees with 30–70% foliage cover [3]. It consisted of 49 trees within a circular area of approximately 50 m in diameter. There were 18 trees between 20–30 m high, 23 trees between 10–20 m high, and 8 trees less than 5 m high.

The forest model was verified by determining the light penetration through the canopy as a function of depression angle. If the tree crowns can be assumed to be a random uniformly distributed volume, then the light transmission away from vertical is [4]:

$$T_\theta = T_{vert} \cos\theta = T_{vert} \sin(\pi / 2 - \theta) \tag{11}$$

where θ is the angle from vertical, and $(\pi / 2 - \theta)$ is the angle from horizontal, i.e. the sensor depression angle.

Foliage penetration was estimated by ray tracing through the forest for depression angles between 45°–90° at 15° azimuth increments for each of three locations in the forest scene. These locations were in positions that a vehicle could have conceivably driven to between the tree trunks. The software developed in MATLAB to simulate LADAR imagery [5] was used for this calculation, by determining the number of rays to reach a flat target on the ground.

A least-squares fit of Eq. 11 to the results gave T_{vert} = 31.3% (Figure 12), although a straight line was found to give a better fit than the sine function. Since the canopy was of course not a homogenous medium, there was considerable variation in the light penetration at each depression angle.

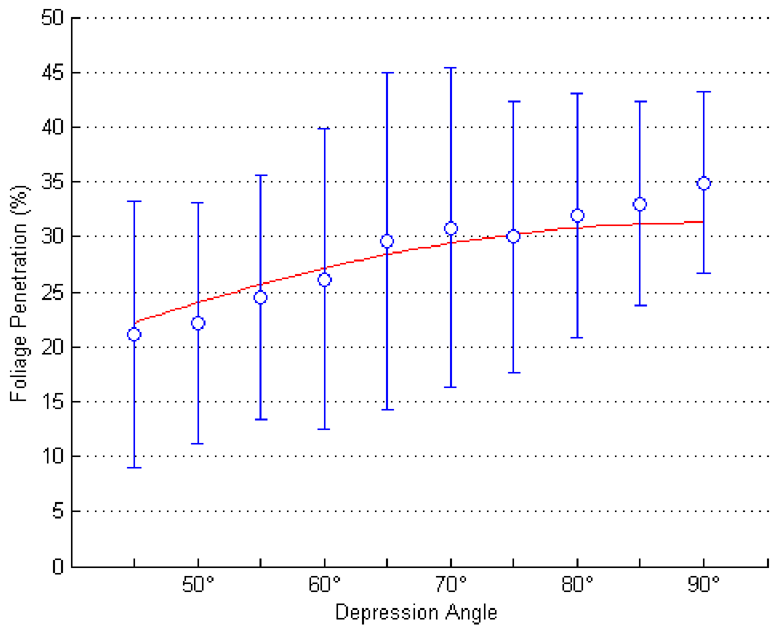


Figure 12 Measured light penetration of the forest canopy model (○ mean ± 1 standard deviation) and predicted penetration (solid red line)

6. Summary

A process was developed to create realistic 3D point cloud models of Australian trees in MATLAB. The software package Maya was used to create the initial tree models and a specially-developed plug-in was used to extract the depth information from images of the trees. This depth information was obtained from multiple views around each tree and exported to MATLAB for reconstructing the trees in a 3D point cloud format. This involved the transformation and mapping of the depth information to form a set of voxels for each view of the tree. These sets of voxels were then rotated, translated and combined to form a complete set of voxels representing the 3D point cloud model of each tree. A final template masking process was applied to each tree model to remove artefacts introduced in the rendering of the tree images in Maya. No solution was found to correct this issue in Maya and it was not encountered with simple geometric test objects such as cubes or spheres, or with a model of a target vehicle which were generated using the same procedures described in this document.

The tree models reconstructed in MATLAB were easily assembled into a representation of a forest scene, which was assessed by light penetrating testing. This 3D point cloud representation of a forest and a model of a target vehicle were successfully used with software developed in MATLAB for modelling and simulating aspects of an airborne foliage-penetrating LADAR imaging system.

This work ultimately enabled realistic 3D LADAR imagery to be produced, which was subsequently used in operations analysis studies. The results of these studies are anticipated to assist in making recommendations for the military operational use of an airborne foliage-penetrating LADAR imaging system.

7. Acknowledgements

The authors would like to thank Dr David Mewett for providing useful guidance and suggestions in preparing this report.

8. References

1. Williams, P.; Stanford, C.; Sanderson, D.; Pash, K.; and Lohmeyer, D. (2003), *A Vegetation Canopy Probability Line of Sight Model*, DSTO technical report DSTO-TR-1402.

2. Lee, A. (Australian National University), photographs supplied by email, 17 November 2005.

3. Specht, R. (1970), Vegetation, in G.W. Leeper (ed), *Australian Environment*, 4th Edition, Melbourne University Press, Melbourne.

4. Chevalier, T.; Steinvall, O.; and Larsson, H. (2007), *Performance of laser penetration through forest vegetation*, Proceedings of SPIE, vol. 6550.

5. Graham, M. (2009), *Design of a Foliage Penetrating LADAR Simulation Tool*, DSTO general document DSTO-GD-0577.

Appendix A: Details of Plug-In to Export Depth Information from Maya

To extract the depth map from an IFF image, a C++ Maya plug-in was created using the `iffreader` library. Each individual pixel depth value may be accessed using the `getDepth` function (given the *x* and *y* coordinates) or the entire map by the `getDepthMap` function.

Additional libraries were used in relation to general plug-in operations:

```
#include <maya/MPxCommand.h>
#include <maya/MStatus.h>
#include <maya/MArgList.h>
#include <maya/MFnPlugin.h>
#include <maya/MObject.h>
#include <maya/MGlobal.h>
#include <maya/MString.h>
#include <maya/MPoint.h>
```

See the Maya API manual for more information.

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING / CAVEAT (OF DOCUMENT)

2. TITLE 3D Point Cloud Tree Modelling	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) Mark Graham and Adam Davies	5. CORPORATE AUTHOR DSTO Defence Science and Technology Organisation PO Box 1500 Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER DSTO-TN-0954	6b. AR NUMBER AR-014-775	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE June 2010

8. FILE NUMBER 2009 / 1106945	9. TASK NUMBER 07 / 105	10. TASK SPONSOR Intelligence and Security Group	11. NO. OF PAGES 17	12. NO. OF REFERENCES 5

13. URL on the World Wide Web http: / / www.dsto.defence.gov.au / corporate / reports / DSTO-TN-0954.pdf	14. RELEASE AUTHORITY Chief, Intelligence, Surveillance and Reconnaissance Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS http: / / web-vic.dsto.defence.gov.au / workareas / library / resources / dsto_thesaurus.shtml

Trees, Simulation, Modelling

19. ABSTRACT

Three-dimensional (3D) point cloud models of trees were developed using the software packages Maya and MATLAB. These models are to be assembled into a forest scene that can be used for producing simulated Laser Detection and Ranging (LADAR) imagery of objects obscured beneath the canopy. This document describes the approach for modelling the trees using Maya then importing and completely reconstructing the models in a custom 3D point cloud format in MATLAB.

Page classification: UNCLASSIFIED