Semi-Automated Landscape Feature Extraction and Modeling

Tony Wasilewski*, Nickolas Faust, Matthew Grimes, and William Ribarsky
Center for GIS and Spatial Analysis Technologies
Graphics, Visualization, and Usability Center
Georgia Institute of Technology

ABSTRACT

We have developed a semi-automated procedure for generating correctly located 3D tree objects from overhead imagery. Cross-platform software partitions arbitrarily large, geocorrected and geolocated imagery into manageable sub-images. The user manually selects tree areas from one or more of these sub-images. Samples are taken from these areas, and color statistics are computed. Tree areas are detected in subsequent images. Tree group blobs are then narrowed to lines using a special thinning algorithm which retains the topology of the blobs, and also stores the thickness of the parent blob. Maxima along these thinned tree groups are found, and used as individual tree locations within the tree group. Magnitudes of the local maxima are used to scale the radii of the tree objects. Grossly overlapping trees are culled based on a comparison of tree-tree distance to combined radii. Tree color is randomly selected based on the distribution of sample tree pixels, and height is estimated from tree radius. The final tree objects (perpendicular intersecting tree cutouts) are then inserted into a terrain database which can be navigated by VGIS[1], a high-resolution global terrain visualization system developed at Georgia Tech.

Keywords: feature extraction, terrain visualization

1. INTRODUCTION

In recent years, much effort has been directed at extraction and modeling of man-made features, such as buildings and roads, from overhead imagery. Close-in imagery is often not available, nor is it practical to collect. Features gleaned from remotely collected imagery can then be used to orient people who are unfamiliar with the terrain in question to the "lay of the land." This exposure to terrain features before deployment can be critical to rapid completion of the tasks at hand, with less time being taken orienting oneself. We have previously developed tools for interactive creation of buildings for terrain visualization, but buildings and terrain are not sufficient for overall scene fidelity. It is necessary to also render some of the largest and most numerous ground features: the trees. The process of manually scanning hundreds of images to indicate location and radius of trees goes beyond being merely tedious.

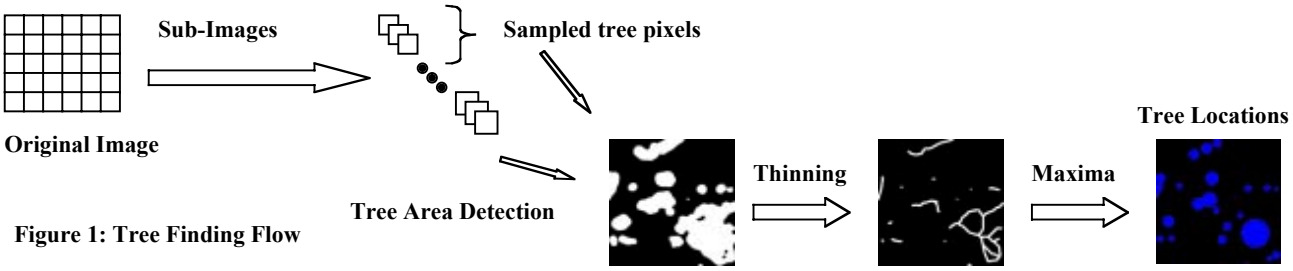

Figure 1: Tree Finding Flow

A procedure was developed to populate our scenes with trees in a fairly quick and automated manner. A geolocated and geocorrected image is split into smaller (512x512) images. The user selects tree areas by creating tree "masks": white tree areas on black background. The masks can be created with any image manipulation software the user has at hand. Given this training data, pixels are sampled, and average color values are extracted. The software uses alternating stages of color selection and blurring to choose areas likely to be trees. Once we have tree areas, we need to derive locations for individual trees. A thinning technique is used to find "skeletons" representing the essential topology of the tree areas, but located in the center of tree groups (where the trunks are likely to be). Tree group width is preserved in this thinning step. Tree group width maxima are used as tree position indicators. Overlapping trees are culled, and tree coordinates are derived from pixel

* Correspondence: tony.wasilewski@gtri.gatech.edu; phone: (404) 894-0133

coordinates in the geolocated subimages. Finally, 3D tree objects (height scaled according to width, colored based on sample tree statistics) are placed at these coordinates in VGIS for exploration. Examples of tree detection are show for a 6-inch true-color ariel photo of the Georgia Tech campus.

2. TREE AREA DETECTION

The first task in building a 3D representation of tree coverage is finding the areas in the overhead imagery which represent tree footprints. The supplied overhead imagery needs to be properly geolocated and geocorrected. Also, images taken in the winter over areas with predominant deciduous growth are not likely to give good results. The tree canopy presents a better contrast with the background in most available image wavebands than does the bare tree branches. Once the user has generated images which represent tree pixels, blurring and color selection / thresholding are used to extract tree area footprints from the remaining images to be analyzed. Large images (> 512/512 pixels) are first split into 512x512 subimages. Files are named according to their upper-left-corner pixel coordinates in the original image.

2.1 Training

The user must select one or more subimages and create "tree masks," which consist of white tree areas on a black background (Figure 2). The software then accepts these masks in a training pass, and outputs average and standard deviation for each color band.

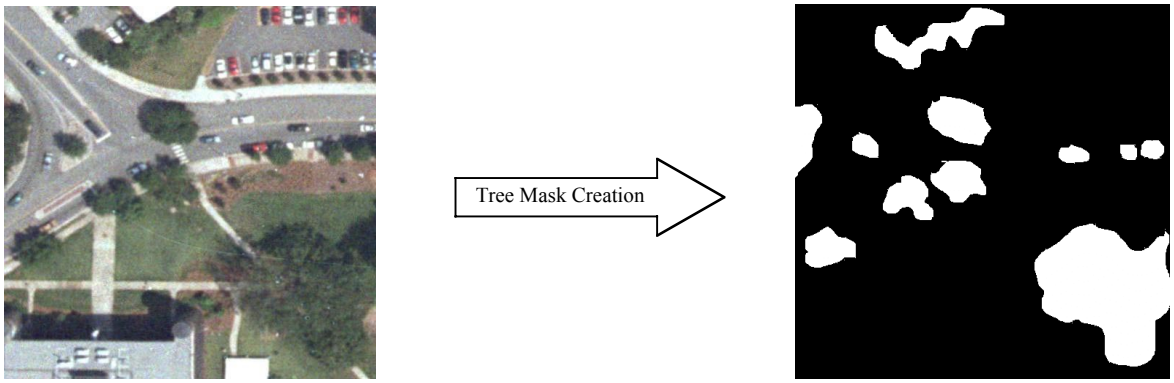

Figure 2: Creating the tree mask

2.2 Detection

The program is run in detection mode on the overhead imagery after training is completed. Each image is first smoothed with a Gaussian filter (kernel size 15, standard deviation 4). This step serves to eliminate, to some degree, color differences within tree canopies, while preserving canopy/non-canopy edges. Note that the standard deviation of the smoothing operator must be scaled down with decreaing pixels/degree image resolution. At some point, this smoothing operator becomes unnecessary, as input resolution is low enough that smoothing of canopy pixels is implicit. We have yet to implement tree-finding with multiple image resolutions, so the precise formulation of this scaling is yet to be determined.

Next, pixels within 1 standard deviation of the sampled color mean are set to white, while the rest of the image pixels are set to black. This may or may not select all pixels which are part of a given canopy, but at least provides one or more "seed" areas within canopies of the same general color statistics as the tree types sampled. Note that this method works poorly if trees of widely varying color are selected in training. Multiple tree species can be detected by separate sampling / detection runs for each species.

A second Gaussian smoothing is performed on the tree area seed blobs. Standard deviation is increased to 6 pixels such that seeds are "grown out" to include canopy pixels which were missed in the color selection process. After smoothing, The image is thresholded to map all non-black pixels to pure white. This smoothing/thresholding step tends to map small, irregular color selection shapes into slightly larger and roughly circular blobs, as expected for actual trees. See Figure 3 for an example tree detection output result.

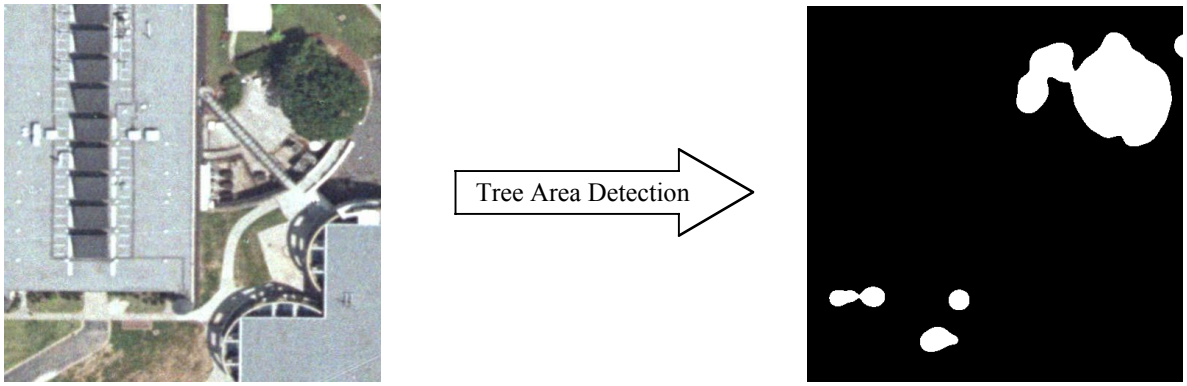

Figure 3: Tree Area Detection Sample

3. TREE LOCATION IDENTIFICATION

Given blobs which represent tree canopy areas, we a presented with the problem of deriving reasonable individual tree locations for placing 3D tree models. Our approach is to "thin" the tree areas down to skeletonized representations, which gives us anchor lines for tree placement. We want to preserve blob region endpoints, while making sure region connectedness is retained. The anchor lines should be located midway between the nearest adjacent blob boundary. Note that this includes inner boundaries, or holes, in blob regions. We added a simple augmentation to the traditional thinning mechanism, which preserves rough line distance from the nearest blob border. Maxima along these lines provide reasonable tree location estimates. Finally, trees which overlap excessively are removed.

3.1 Standard Thinning

The basic idea of thinning[2, 3] is to reduce an image full of objects to and image containing lines which represent these objects. Objects are then iteratively "widdled away" until a skeletal representation is left. Pixels are removed (white→black) from the original image on each iteration unless they meet the following criteria:

- Pixel has no white neighbors (neighborhood is 8 surrounding pixels)
- Pixel is not a single-pixel-wide line endpoint
- Removal of pixel would change local connectivity

Here "connectivity" refers to the relationship between image regions (areas of all-white or all-black). The algorithm procedes as follows, where P0→P7 refer to neighbors of pixel P, with 0 being directly above and successively numbering as we proceed clockwise around the pixel:

```
while (pixel deletion occurs)
        for (each pixel P)
                if (2 <=number of white neighbors <= 6) AND
                (number of black→white transitions from P0→clockwise back to P0) AND
                ((P0 AND P6 AND P2 are black) OR (number of black→white transitions surrounding P0))  AND
                ((P0 AND P6 AND P4 are black) OR (number of black→white transitions surrounding P6))
                        delete pixel P
                end if
        end for
end while
```

3.2 Modified Thinning

Standard will produce a simple thinned image, with white pixels representing blob skeletons and black pixels as background. However, we wish to preserve the thickness information of the blobs, as bulges in tree area blobs are likely to represent tree locations. For instance, if our tree detection process finds two trees which are next to each other, the blurring and thresholding process is likely to "melt" the two trees together, resulting in a dumbell-shaped blob, as seen in Figure 4. Simple

thinning gives an elongated line representing the midway line of the blob, but otherwise contains no clue that two trees are present.

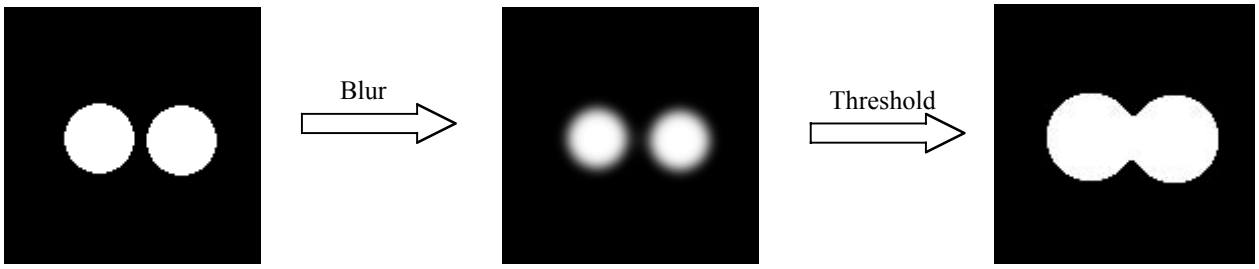

Figure 4: Blending Artifact of Tree Detection

We have modified the standard thinning algorithm with an extra procedure which encodes distance of the thinned midline from the pre-thinned blob boundary. The resulting skeleton is equivalent in shape to the standard skeleton, but with pixel value reflecting distance from boundary instead of pure white. The process can be imagined as the length of the path a boundary pixel follows as the blob is widdled away, travelling from it's initial boundary position to a final resting place on the resultant skeleton.

A "scale" image is initialized to be identical to the binary white/black tree area blob image. Black pixels are given value 0, and white pixels value 1. This is a floating-point image, so fractional pixel values are allowed. As the thinning algorithm proceeds, pixels in the scale image are examined in unison with pixels from the binary valued image. If a scale image pixel is not being deleted (and it hasn't yet "absorbed" any neighbor values), its neighbors are inspected. If any of these are being deleted, the scale image pixel "absorbs" the average pixel value of the deleted pixels, by adding their average pixel value to it's value of 1. It's as if the blob is imploding, and the infalling shock front retains a running sum of its distance traveled. The modified algorithm is as follows:

Image B is the binary (0/1) tree detection image
Initialize floating-point image S with values from image B

while (pixel deletion occurs)
for (each pixel P_B)
determine if pixel P_B will be deleted on this iteration via standard thinning algorithm
end for
for (each pixel P_S and each pixel P_B)
if ((P_S == 1) AND (P_B not getting deleted this iteration))
if (P_S has any neighbors getting deleted)
P_S = Average (deleted neighbors + 1)
end if
end if
end for
delete pixels from B and corresponding pixels S if they are flagged for deletion this iteration
end while

The result is two images, image B being the binary thinned result and image S containing scaled skeleton pixels. Figure 5 shows modified thinning on the threshold image of Figure 4. Note that resultant image gamma was increased to show pixel value differences more clearly.

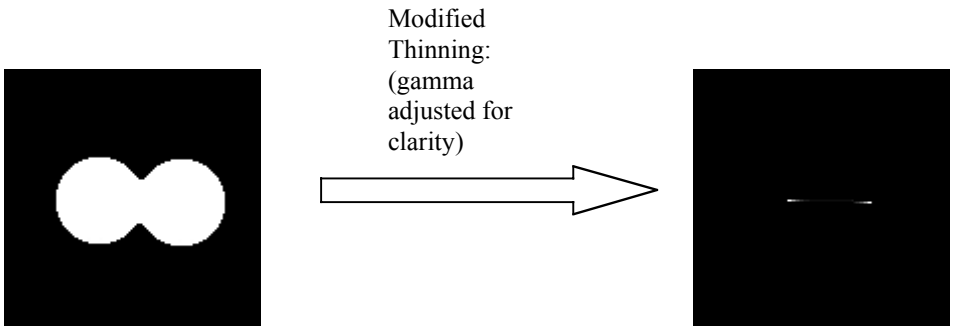

Figure 5: Blending Artifact Blob After Modified Thinning

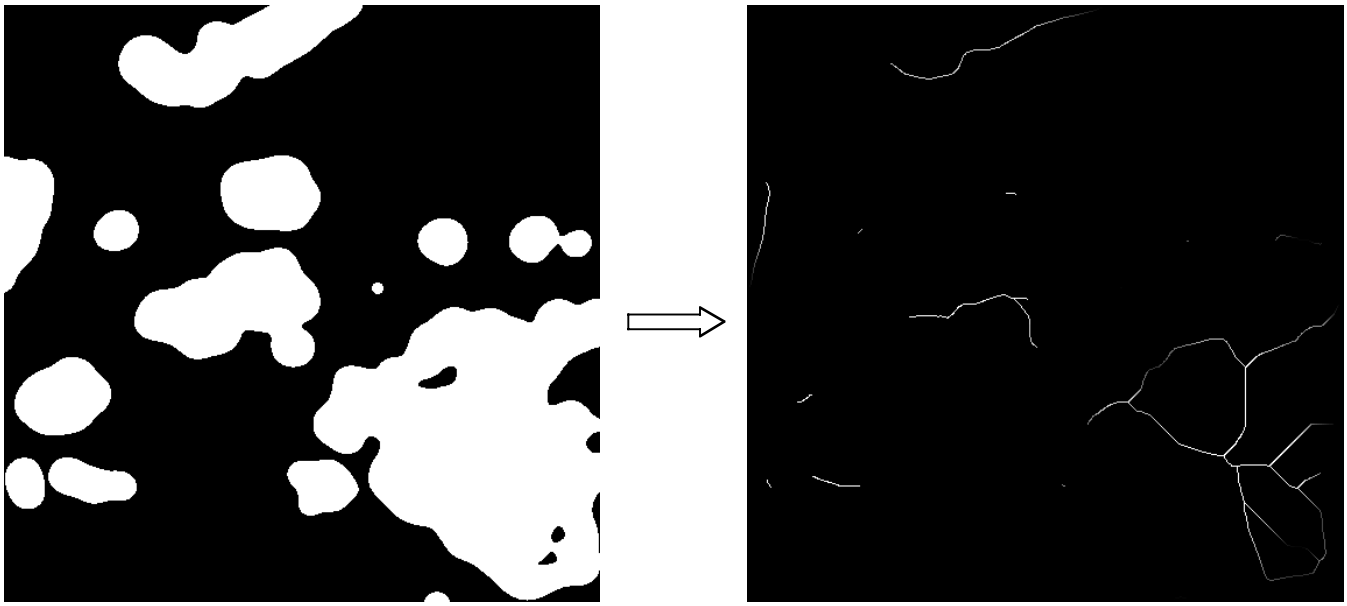

Figure 6: A More Complicated Example of Modified Thinning

3.3 Maxima Detection / Plateau Correction

Once thinned versions of tree area blobs have been created, local maxima in tree skeletons are used as approximate indicators of actual tree positions, and the value at the thinned maxima location is used to indicate tree radii. Maxima retrieval is performed using the following algorithm:

Initialize image S to all 1's

for (each pixel in thinned image T)
if (at least one neighbor is greater in value V than current pixel)
Recursively set adjacent (touching) pixels in S to 0 if corresponding pixels in T have same value V
end if
end for

The resulting image is equal to 1 for local maxima in the scaled thinned image, and 0 otherwise. It preserves "runs" of local maxima, however, which are plateaus of maximal value that may occur if sequential skeleton pixels are equidistant from blob boundaries. A corrective pass through the maxima image detects such plateaus, and replaces them with the average coordinates of the equi-value pixel runs. In this manner, we get individual "island" pixels suitable for using as tree locations. These pixel locations, as well as their corresponding scale value, are recorded and used as the locations and radii for tree objects.

We run into a problem using this technique if the resulting trees are grossly overlapping. As this is physically impossible among trees which are competing amongst one another for available sunlight, we retain the larger tree and discard the smaller. We have found that culling is a good idea if the tree→tree distance is less than 4/5 of the sum of the tree radii.

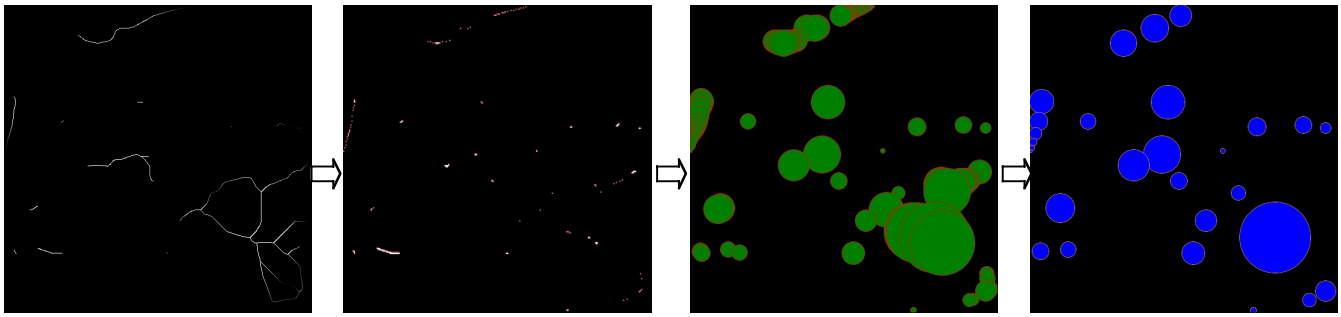

Figure 7: Thinned → Maxima (with plateaus) → Overlapping Trees → Final Trees (Slighty Overlapping)

Since the subimages are named based on their upper-left-hand-corner pixel locations, it is a simple matter to relate pixel coordinates in the subimage to geolocation. This location is used to place the trees in a terrain database.

Figure 8: Trees Found in Area Around Our Building

4. TREE OBJECT CREATION

Tree objects are created by "splattering" a square tree cutout with colored spots. The cutout is initialized to black, then hit randomly with gaussian-shaped color spots. Splat position is uniformly distributed across the image cutout in X and Y. Splat color is varied based on sample tree color mean and standard deviation, and splat shape is a gaussian with standard deviation set to 1/30th of the tree cutout dimension. Two such cutouts (Figure 9) are place perpendicular to one another. The object is scaled in width based on tree radius as described previously, and height is also varied based on width.

Figure 9: Tree Object Seen From Above (And Slighly Off-Axis)

5. CONCLUSIONS

We have developed a systematic procedure for large-scale semi-automated extraction of trees from overhead photos. Tree locations and size compare favorably to overhead imagery.We are in the process of writing a conversion utility to import our tree objects into VGIS and rendering on the terrain surface. Impovements and additions to this procedure are underway. Among these: using hyperspectral image information for color selection, including a texture measure (spatial frequency statistics for all bands over tree areas) for better discrimination of tree/nontree areas, and adding a GUI front end (program is command-line driven at this point).

ACKNOWLEDGMENTS

The tree extraction software was developed with support of the Office of Naval Research and the Naval Research Labs.

REFERENCES

1. .D. Koller, P. Lindstrom, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner. *Virtual GIS: A Real-Time 3D Geographic Information System* Proceedings of Visualization'95

2. T. Y. Zhang, C. Y. Suen, *A Fast Algorithm for Thinning Digital Pattern*, Comm. ACM, vol. 27 nº3, pp. 236-239, 1984

3. Jain, Anil K., *Fundamentals of Image Processing*, Prentice-Hall pp. 382-385, 1989