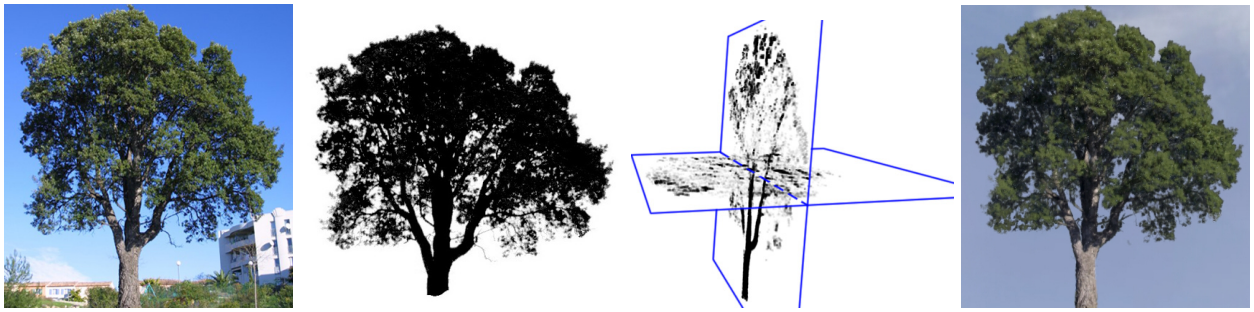
Volumetric Reconstruction and Interactive Rendering of Trees from Photographs

Alex Reche*
REVES/INRIA Sophia-Antipolis &
CSTB, Sophia-Antipolis, France

Ignacio Martin†
GGG, University of Girona, Spain
http://iia.udg.es/IG.html

George Drettakis*
REVES/INRIA Sophia-Antipolis, France
http://www.inria.fr/reves



(a) (b) (c) (d)

Figure 1: Our method captures and renders existing trees from photographs, by estimating opacity in a volume, then generating and displaying view-dependent textures attached to cells of the volume. (a) One of the original photographs of an oak. (b) The α mask used for the opacity estimation. Two cross slices of the resulting opacity are shown in (c). A synthetic image of the original view, using our view-dependent rendering, is shown in (d). Textures are attached to billboards in cells of the volume and are generated based on estimated opacity.

Abstract

Reconstructing and rendering trees is a challenging problem due to the geometric complexity involved, and the inherent difficulties of capture. In this paper we propose a volumetric approach to capture and render trees with relatively sparse foliage. Photographs of such trees typically have single pixels containing the blended projection of numerous leaves/branches and background. We show how we estimate opacity values on a recursive grid, based on alpha-mattes extracted from a small number of calibrated photographs of a tree. This data structure is then used to render billboards attached to the centers of the grid cells. Each billboard is assigned a set of view-dependent textures corresponding to each input view. These textures are generated by approximating coverage masks based on opacity and depth from the camera. Rendering is performed using a view-dependent texturing algorithm. The resulting volumetric tree structure has low polygon count, permitting interactive rendering of realistic 3D trees. We illustrate the implementation of our system on several different real trees, and show that we can insert the resulting model in virtual scenes.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: 3D reconstruction, interactive rendering, visibility estimation

*e-mail: {Alex.Reche|George.Drettakis}@sophia.inria.fr
†e-mail:Ignacio.Martin@udg.es

1 Introduction

Capturing real-world trees for subsequent rendering in virtual environments is an interesting and challenging task. Existing digital capture techniques such as laser range finders, stereo or modelling from images have been extensively developed for solid objects which have a well-defined opaque surface. Once the object is captured, standard computer graphics methods are used for rendering. Trees, and vegetation in general, are a special case. The complex geometric structure of leaves and branches means that traditional scanning methods cannot usually be applied. In particular capturing depth information of such objects is a very difficult task because of the high frequencies implied by the complex, small geometries of the leaves and twigs. Procedural models for trees can be guided by images [Shlyakhter et al. 2001], but result in models with very high polygon counts (hundreds of thousands or even millions of polygons per tree).

Our goals are i) to capture real trees from a small number of photographs, using a relatively simple and fast procedure, without the need for specialized equipment or setup; ii) to create a low polygon count tree model which permits realistic interactive renderings of the captured tree.

To achieve these goals we make a key observation: at a reasonable distance, each pixel in a photograph of a tree is typically the blended projection of a number of leaves/branches and the background. This can be seen as a transparency effect, in which the pixel value corresponds to a mixture of tree/leaf colors and background modulated by the opacity of the "tree volume" along the pyramid defined by the camera and the pixel.

Based on this observation, we adopt a volumetric model for the rendering of trees. We cast the problem in a direct volumetric rendering context [Max 1995]; we consider the tree as a volume, with opacity and color values. We estimate opacity based on a small set of calibrated digital photographs of a tree. We first estimate opacity of the tree volume, by optimizing over a recursive grid of opac-

ity values. Our optimization uses alpha-mattes extracted from the photographs as estimates of cumulative opacity at the pixels (see Fig. 1(b)). Once the recursive grid of opacity values has been estimated, we assign a billboard to each grid cell. For each billboard, a view-dependent texture is generated for each view, using the original image, the number of billboards projected to the image and the opacity of each cell. These textures capture the high frequency detail and are rendered with a view-dependent texturing algorithm.

The main contributions of our approach are:

- The efficient estimation of the opacities for a tree volume on a recursive grid, using the alpha-mattes computed from a small set of calibrated images around a tree.
- The generation of view-dependent textures for each cell of the recursive grid, which allows high-quality view-dependent rendering of the resulting, low polygon tree model using billboards.

This entire process is efficient and compact, creates satisfactory reconstructions of trees, and is particularly suited to those with relatively sparse foliage (see Fig. 1), which are not well handled by previous methods.

2 Related Work

Modelling and rendering of trees has interested computer graphics researchers over the last decades. A large part of previous work concentrated on the generation and rendering of entirely synthetic trees based on procedural methods such as grammars (L-systems) (e.g., [Prusinkiewicz and Lindenmayer 1990; Deussen et al. 1998]), or on rule-based plant growing systems based on codified botanical knowledge such as the AMAP system [de Reffye et al. 1988]. Such approaches have been used to generate stunning images of forests and trees in general, with a high degree of realism. However, they do not capture the aspect of existing, real trees.

The multi-layer z-buffer method uses precomputed synthetic images of trees for rendering [Max and Ohsaki 1995; Max 1996]. Volumetric texture approaches have been used to model and render trees; the complex geometry is represented as an approximation of the reflectance at a distance [Neyret 1998]. An adaptation of this approach to hardware was developed later, using textured slices for interactive rendering [Meyer and Neyret 1998]. Meyer et al. [Meyer et al. 2001], presented a hierarchical bidirectional texture solution for trees at different levels of detail, i.e., leaves, branches, up to the entire tree. This results in very efficient level-of-detail rendering for trees. Efficient rendering of trees can also be achieved using point-based methods [Deussen et al. 2002]. A more recent approach has been developed by [Qin et al. 2003], in which a volumetric approach effects an implicit level-of-detail mechanism, for lighting (both sun and sky) and shadowing, using shadow maps.

With the emergence of augmented reality applications, as well as the generation of virtual objects from real sources, the need for capture of existing trees has become more evident. A simple approach using two textures based on photos has been developed [Katsumi et al. 1992]; rendering is achieved using horizontal slices through the tree, the emphasis being on shading and shadowing. A different approach to capturing and rendering trees was presented by Shlyakhter et al. [Shlyakhter et al. 2001]. This approach first creates a visual hull-like representation of the shape of the tree and then fits an L-system to the resulting model. The photographs are reprojected onto the resulting polygons of the L-system, resulting in very realistic representations of real trees, at the price of high polygon counts for the resulting models.

Image-based methods have also been developed which can handle objects with semi-transparent silhouettes. These include opacity hulls [Matusik et al. 2002], surface light fields [Wood et al. 2000] and microfacet billboards [Yamazaki et al. 2002]. In all of the above cases, a large part of the object is opaque, defining an overall opaque geometric shape, which is recovered. This shape is required since these approaches use either pure standard "surface textures", or, in the case of microfacet billboards, require the underlying geometry to place and cull the billboards used.

In the approach presented here, we have been inspired by volume rendering methods. The rendering model we have adopted is based on the original direct volume rendering approaches (e.g., [Blinn 1982; Levoy 1988; Drebin et al. 1988; Max 1995]). Although our method is closer to image-based rendering than volume rendering, our approach is related to polygon-rendering solutions for volumes (e.g., [Westover 1990; Shirley and Tuchman 1990]).

Our opacity estimation approach is inspired by the approaches used in medical imaging. Our method has similarities to algebraic methods used for computed tomography, such as the SART algorithm (e.g., [Andersen and Kak 1984]). The construction of a voxel representation for the tree has similarities with the voxel coloring [Seitz and Dyer 1997] or space carving [Kutulakos and Seitz 1999] approaches. However, these approaches do not currently treat semi-transparent volumes such as those we construct for trees. The Roxel algorithm [Bonet and Viola 1999] extends space carving to treat opacity using techniques for opacity estimation [Szeliski and Golland 1998]. A related approach, using a hierarchy of cells for fast rendering of complex scenes was presented by [Chamberlain et al. 1996]. Compared to medical imaging or the Roxel approaches, we need to represent high frequency detail with low storage cost, which we achieve using the recursive structure combined with the view-dependent texturing technique.

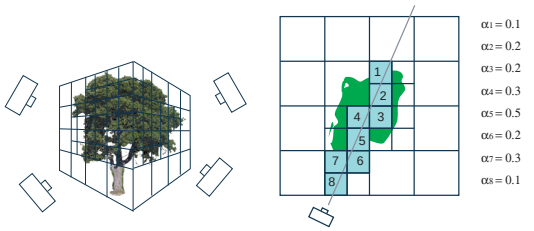


Figure 2: (a) A recursive grid is placed around the tree. (b) A ray emanating at pixel p traverses the grid; a set of cells are collected.

3 A Volumetric Rendering Approach for Trees

As mentioned in the introduction, we have chosen a volumetric rendering model to represent and render trees, given the fact that we are dealing with photographs containing many semi-transparent pixels.

Consider the tree shown in Fig. 2(a); we create a recursive grid enclosing the tree which is surrounded by the cameras. For a large number of pixels in these photographs, each pixel is semi-transparent, and we estimate an alpha value corresponding to the cumulative opacity α_p for each pixel (see Section 4). In typical volume rendering applications (e.g., [Levoy 1988; Drebin et al. 1988]), a sampled opacity field is given at vertices of a grid, typically coming from a scanning device (X-rays etc.). In our case, we will need to estimate the opacity in the grid, based on the alpha values extracted from the photographs. We assign the value α_i to each cell i of the grid. We show how to do this in Section 5.

Given the opacity and color, we can adopt a standard volume rendering equation to generate images of our trees. Consider a ray emanating from a pixel p, which traverses n grid cells which have opacity values α_i (see Fig. 2). For now, we assume that for a given pixel we have assigned a color c_i for each cell. The intensity

Figure 3: Three photographs of typical trees we captured using our method. In each photograph the background color is significantly different from the colors of the leaves, trunk and branches.

I at a pixel is defined using standard volume rendering equations (e.g., [Levoy 1988]):

$$I = \sum_{i=0}^{n} \alpha_i c_i \Pi_{m=i+1}^{n} (1-\alpha_m) \quad (1)$$

which expands to the standard compositing equation for rendering:

$$I = \alpha_n c_n + (1-\alpha_n)(\alpha_{n-1} c_{n-1} + (1-\alpha_{n-1})(..(1-\alpha_1)\alpha_0 c_0)..) \quad (2)$$

To render the tree volume, we will develop a rendering algorithm which approximates Eq. 2. We first need to estimate the α_i values, and we then need to approximate the color.

In reality, both the opacity and color values are anisotropic and inhomogeneous; estimating and storing them accurately would clearly be intractable, except in the simplest cases. We choose the following simplifications: we will estimate inhomogeneous but *isotropic* discrete opacity values defined on a recursive grid. For color, we will not perform an estimation, but we will partially preserve the anisotropic quality of the color by using a view-dependent texturing approach, with textures generated based on the opacities and the original photographs of the tree (see Section 6). We use billboards centered at the cells for the actual rendering.

Our capture method has three main steps. The first step is image capture and alpha matting. In the second step, these alpha values are used to estimate the opacity values in a grid. In the third step, we used estimated opacities to generate billboards for each cell and each image. Rendering is achieved with view-dependent texturing on the billboards to efficiently display the trees. These three steps are described in detail in the following sections.

4 Image Capture and Alpha Matting

The first step in our algorithm requires photographing the tree and generating the alpha maps. The tree chosen for reconstruction currently needs to be in a suitable position so that it can be photographed with a relatively distinct background (see for example Fig. 3, in which we can photograph the tree in a full circle with a sky background). Trees in urban environments often have this property, at least for the majority of views.

Photographs are then taken while moving around the tree. Our experiments have shown that approximately 20-30 photographs are required. We then need to calibrate the cameras of these photos. We accomplish this step using REALVIZ ImageModeler (www.realviz.com), but any standard algorithm can be applied for this step [Faugeras 1993]. The calibrated cameras will be used in subsequent steps. By adding a few markers in the scene (we use sticks with colored play-doh on the top), calibration takes less than 10 minutes for about 20 images.

Next, we extract alpha masks for each photograph, using the algorithm of Ruzon and Tomasi [Ruzon and Tomasi 2000]. This algorithm requires user intervention, in which zones of background and foreground are specified. Consider for example the tree shown in Fig. 4. The user specifies foreground and background, and we can see that the alpha-mask algorithm succeeds in finding semi-transparent regions in the image. The area of the pixel contains a mix of leaves/branches and background. The alpha-mask calculates this visibility coverage using colors. Alpha-masks of 0 correspond to background regions and alpha-masks of 1 to completely opaque regions.

The intuition behind the alpha-masking approach is simple. The pixels in the image are transformed into LAB color space and clustered by color distribution. For each pixel, its color can be seen as a best fit interpolation between the distributions of the clusters for the foreground and background colors. The value of this interpolant is the alpha value of the pixel

We thus consider that for pixels with an alpha mask strictly between 0 and 1, the color of the pixel C_p is a linear interpolation in LAB color space of the background color C_b and the foreground color C_f:

$$C_p = \alpha_p C_f + (1-\alpha_p) C_b \quad (3)$$

C_f and C_b are computed as weighted sums of background and foreground, and perturbed accordingly. We create an alpha-image containing the values α_p for each pixel and an image of "pure foreground" C_f values used later for rendering (see Section 6).

5 Reconstruction

To achieve volumetric rendering of trees, we need to estimate opacities in each grid cell. We clearly do not have enough information *inside* the volume to be able to estimate both color and opacity, since we only have photographs without any depth information. Even if the depth information were available, it could not be precise enough to be useful. For this reason, we will restrict estimation only to opacity, treating color in the rendering process as will be discussed in Section 6.

A complete iteration of the reconstruction process will loop through each pixel of each image. For each pixel, we cast the pyramid defined by the camera and the pixel square into the grid. The cells touched by this pyramid are treated. The algorithm performs an optimization step, trying to choose the best values for the opacities of these cells based on the corresponding opacity value α_p of the pixel. Note that empty (transparent) cells are ignored for all steps of our approach.

Before the first iteration, we initialize the opacity values α_i in each cell, by projecting each cell into each alpha-image, and using the minimum of the average opacity values in the respective images as the value of α_i. Given that the images are the only data we have available, this minimum is the closest we can get to the final solution.

At each iteration, an initial opacity value α_i (or transparency $t_i = 1-\alpha_i$) is stored at each cell, together with a δ_i^t, which is initialized to 0 at each cell. We will use the δ_i^t values to accumulate the movement of transparencies due to the minimization for each pixel in each image. In particular, we accumulate intermediate transparency optimizations for each pixel and cell in δ_i^t, and update all cell transparency values simultaneously at the end of each iteration.

For each pixel of each image we approximate the pyramid by casting a ray through the grid. For each ray we collect the grid cells hit, with initial opacity values α_i, $i = 1...n$. We experimented with a larger number of rays, but the resulting increase in accuracy did not justify the additional expense.

Figure 4: Left: the original image. Middle: the regions specified roughly by the user as foreground and background. Right: the computed alpha-mask (note that we used black for opaque for clarity of the figure).

Using the same reasoning as for the volume rendering equation (Eq. 2), and considering an absorption only model [Max 1995], we assume that the pixel opacity value α_p should be:

$$\alpha_p = \alpha_n + (1-\alpha_n)(\alpha_{n-1} + (1-\alpha_{n-1})(...(1-\alpha_1)\alpha_0)..) \quad (4)$$

where α_i is the opacity in each cell i. If we recast Eq. 4 as a recursive function f^n, we get:

$$f^0 = \alpha_0 \quad (5)$$

$$f^n = \alpha_n + (1-\alpha_n) f^{n-1} \quad (6)$$

It can easily be seen by induction, that:

$$f^n = 1 - \Pi_{i=0}^{n}(1-\alpha_i) \quad (7)$$

If we substitute $t_i = 1-\alpha_i$ (which are the transparency values of the cell), we need to find the values t_i which best fit the equation:

$$t_p = \Pi_{i=0}^{n} t_i \quad (8)$$

where $t_p = 1-\alpha_p$. Clearly, we have an under-constrained problem, since we need to satisfy a system of equations (8). The number of these equations is equal to the number of pixels in all images. Our goal is to obtain a best-fit solution for the t_i values in the cells. One approach would be to use Eq. 8 directly with a standard iterative minimization technique such as a conjugate gradient method.

For reasons of efficiency however, we choose to transform Eq. 8 into a sum by taking logarithms:

$$\log t_p = \log \Pi_{i=0}^{n} t_i = \sum_{i=0}^{n} \log t_i \quad (9)$$

Eq. 9 can be seen as an n-dimensional plane, and desirable values for $\log t_i$ should satisfy this plane equation.

Interestingly, the logarithms have a physical interpretation. Transparency $t(s)$ is defined as [Max 1995]:

$$t(s) = e^{\int_0^s -\tau(t)dt} \quad (10)$$

where τ is the extinction coefficient of the volume. If we approximate the integral using a Riemann sum at intervals Δx, the discrete transparency t_i can be defined as [Max 1995]:

$$t_i = e^{-\tau(i\Delta x)\Delta x} \quad (11)$$

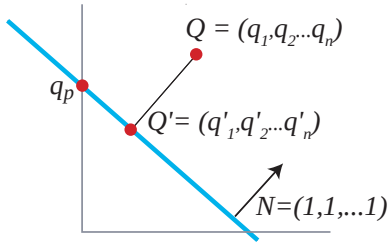


Figure 5: Illustration of the projection step for the optimization of the t_i (or respectively q_i values). The illustration is in 2-D for simplicity, but is actually performed in N-d, where N is the number of cells intersected by the ray.

Taking the logarithms gives:

$$\log t_i = -\tau(i\Delta x)\Delta x \quad (12)$$

Thus the $\log t_i$ values can be seen as a discrete approximation of the extinction coefficients. In what follows we use $q_i = \log t_i$, where q_i is the analog of the discrete extinction coefficient, and $\log t_p = q_p$. We also define the vector $Q = (q_1, q_2, ..., q_n)$.

Recall that the goal is to find a best fit of the t_i values for the cells for the given t_p of the alpha image. The transformation via the logarithm allows us to perform this step by a direct projection onto the $n-$dimensional plane P, which has normal $N = (1,1,...1)$ and with intercept q_p on one of the axes (see Fig. 5). We can thus simply find the projection $Q' = (q'_1, q'_2 ... q'_n)$ of Q onto P.

We then revert to the space of transparencies by taking the exponent $t'_i = e^{q'_i}$. We will weight this projection by the relative coverage of the ray and the voxels being considered. The exact calculation of the weight is illustrated in Appendix A. The weighting can be seen as a linear interpolation between the component values of Q and Q'. Thus, for a given cell i, we have:

$$\delta_i = w_i(t'_i - t_i) \quad (13)$$

The stored δ_i^t value of the cell is then incremented with the δ_i value, representing the optimization of t_i due to this ray.

To complete an iteration we perform this estimation for each pixel of each image.

When the iteration is complete, for each cell we add the δ_i^t values, normalized by the sum w of the weights w_i, into the transparency values t_i to initialize them for the following iteration. We iterate until convergence, which is defined when the values of t_i change less than a threshold in a single iteration. The result is an estimate of transparencies (and thus opacities) for each grid cell. We try and do a "best-fit" for all input images; this approach will tend towards a local minimum. Note that in practice we recover and store the transparency values t_i from the estimated discrete extinction coefficients q_i, passing through the exponential.

Finally, if t_i becomes greater than a *transparency cutoff threshold* when adding in δ_i^t's, we set t_i to 1; the cell thus becomes transparent and is subsequently ignored. The reconstruction process is summarized as follows:

```
while not converged
    foreach cell i
        init δ_i^t,w
    endfor
    foreach image
        foreach pixel
            list-of-cells = cast-ray-into-grid
            create vector Q = (q_1,..q_n) from t_i's
            project Q onto plane P
            foreach cell i
                δ_i = w_i(t_i' − t_i)
                update δ_i^t ; w += w_i
            endfor
        endfor
    endfor
    foreach cell i
        t_i += δ_i^t / w ; if t_i > thres then t_i = 1
    endfor
endwhile
```

A visualization of slices of opacity values are shown in Fig. 6. The process typically takes 3-5 minutes on the models we tested on a Pentium IV 2.7 Ghz PC, and 3-4 iterations are sufficient to achieve convergence.

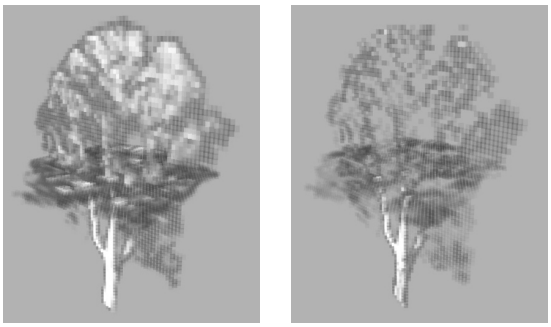


Figure 6: Illustration of slices of opacity values at the initialization step (left) and after the end of the reconstruction process (right).

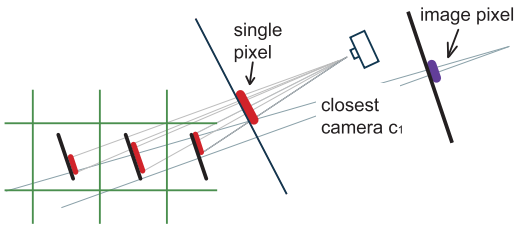


Figure 7: An illustration of the problems of naive rendering using view-dependent texturing: texture repetition results in blurring.

6 Texture Generation and View-Dependent Rendering

At this stage, we have a recursive grid with opacity values which represents the tree. Rendering the tree is a complex task, since we want to preserve high-frequency information existing in the photographs, but at the same time we do not have any additional 3D information other than the low-frequency opacity estimate per cell. Given our goal of low-polygon count for the final representation, we cannot subdivide the recursive grid too deeply, since this would lead to an explosion in the number of primitives required.

Our solution is to attach a billboard (a small polygon which is always oriented towards to viewer) to each cell. A naive approach to rendering would be to project a view-dependent texture from the input images onto each billboard, potentially blending between the n closest cameras, in the spirit of Debevec et al. [Debevec et al. 1996] or the Unstructured Lumigraph [Buehler et al. 2001]. Our tests with this approach gave severe blurring artifacts, since texture is repeated across the billboards corresponding to a given pixel (see Fig. 7). We present a heuristic solution where we use a billboard for each cell and we generate an appropriate texture for each image.

Generation of the billboard textures

We generate billboard textures in a preprocessing step by looping through input images and all cells. For each image i and for each cell c we construct an $M \times M$ billboard texture T. We typically set $M = 8$ or $M = 4$, which is a good compromise; smaller sizes would require higher subdivision. Note that we use the images containing the "pure foreground" values C_f (Eq. 3).

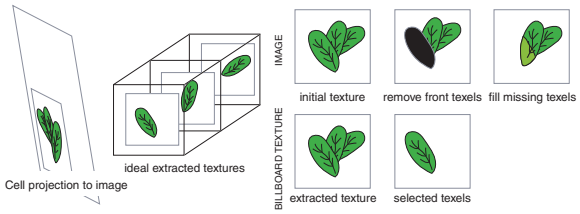


Figure 8: (a) Illustration of the ideal case; (b) Texture generation for the first cell, and update of the image.

The intuition behind this process is illustrated with an artificial ideal case, shown in Fig. 8(a). Three cells in a row project onto a region of the image containing the three leaves. If all information were available, we could generate the three separate textures with each leaf, one for each cell. To do this, we would visit the cells front to back. For each cell, we would project the cell to the image, extract the texture, mask out only the part corresponding to this cell and assign this as a texture for the cell. We would then update the image and proceed to the following cell (Fig. 8(b)). The resulting billboards would avoid the blurring problem mentioned above.

In what follows, we use the term *pixels* for the values in the RGB and alpha images, and *texels* for pixels of the billboard textures. We project from billboard to image space and vice-versa using appropriate resampling for all values. We use a copy C_i of the (RGBA) image since we modify pixel values.

For a given image and a given cell, we have estimated transparencies t_c for each cell, and the α_p (or transparency $t_p = 1-\alpha_p$) value for each pixel. We also compute the total number of billboards rendered at each pixel using the stencil buffer, and keep track of the remaining number N_{curr} of cells not yet rendered as we process alpha image pixels.

We introduce a heuristic approximation to the ideal case given the minimal information at our disposal, i.e., alpha image pixel values, cell transparencies, and the current number of cells (or billboards) rendered at each pixel.

As in the ideal case, for each image, we render cells front-to-back, and find the region R in the image corresponding to the projection of the cell. For each texel, we decide whether it should be selected using the corresponding image pixels. Our metric for selecting these texels is based on an appropriate comparison of cell and pixel transparencies. Our heuristic assumes that coherent pixel/cell transparencies indicate a higher probability that the pixels correspond to geometry in this cell. We will thus prefer pixels with cell/pixel values which are close according to the metric. To complete the method, we update the pixel color and transparency values after treating each cell.

To implement this approach we first create a quantity from the cell transparency t_c which is comparable to the corresponding alpha-image t_p values. This is done by raising t_c to the power of N_{curr} of billboards not yet rendered at this pixel. This approximates the influence of the current cell on the cumulative transparency of all cells projecting to this pixel: We note this value $t_c^{N_{curr}}$. For each pixel we also store a value t_{curr}, which is initialized to t_p.

For each texel we thus compute the importance, H_t, which gives higher values for texels with t_{curr} close to $t_c^{N_{curr}}$:

$$H_t = \frac{1}{t_{curr} - t_c^{N_{curr}}} \tag{14}$$

To complete the approach we update the value of t_{curr} as follows: $t_{curr} = t_{curr}/t_c$, thus "removing" the influence of the current cell as estimated previously using the power computation.

We store an age field with all pixels which is initially 0; recall that texels resample this value. Texels are ordered by age and importance, and the first $\lfloor \alpha_c * M * M \rfloor$ texels are selected to be used on this billboard. For the remaining texels, the age of their corresponding pixels is incremented, thus increasing their chance of being selected later, and spreading out the choice of texels. Some pixels of the image may remain unselected, but this does not appear to cause significant artifacts.

As seen above, we base the number of texels selected for each billboard on the opacity of the cells. Initially, we tried a random choice of texels given this number, and the results were unsatisfactory. Use of these heuristics clearly improved the result.

The pixels used on the billboard are removed from the copy of the original image, and are filled by interpolating neighboring pixels. A more sophisticated texture generation process could be used for this step, e.g., in the spirit of [Efros and Leung 1999].

The final texture for this cell billboard and this image is thus an $M \times M$ RGBA texture, for which the pixels chosen as valid are assigned the RGB values of the (potentially modified) copy of the original image, and the A value is the value of the α image. This process is summarized in the following pseudo-code:

```
foreach RGBα image i
    make a copy C_i of image i
    render billboards to initialize N_curr at each pixel
    foreach cell c front-to-back
        find projected region R of cell c in image C_i
        create resampled M × M texture T from C_i
        sort texels of T by age and by H_t
        select ⌊α_c * M * M⌋ first texels
        for pixels in R corresponding to selected texels
            t_curr = t_curr / t_c
            replace pixels using texture generation
        for remaining pixels in R, increment age
        for all pixels in R, N_curr = N_curr − 1;
    endfor
endfor
```

The uncompressed textures require a significant amount of memory. To reduce the required texture memory, we dissociate alpha from RGB, and pack two RGB textures if their alphas do not intersect. The resulting memory requirements are about 45% of that originally required. For a typical uncompressed tree of 100Mb, the resulting compressed version requires 45Mb.

View-Dependent Rendering

Once an RGBA texture has been generated for each cell and each original camera, we can render in a straightforward manner. We traverse the cells back-to-front, and render the billboard for each cell with multi-texturing, using the appropriately weighted textures corresponding to the two closest cameras [Debevec et al. 1996]. The RGBA billboard textures are simply blended for each cell visited in back-to-front order, in the sense of the **over** operator [Porter and Duff 1984].

7 Implementation and Results

Our current implementation uses a recursive grid, which subdivides into 3x3x3 subcells (an octree could also be used). Currently, we subdivide all non-empty cells down to the maximum subdivision level. Empty interior cells are not subdivided.

We present two examples of real captured trees. Our approach is however better appreciated in an animation or walkthrough. Please see the accompanying video, available on the ACM SIGGRAPH 2004 Full Conference DVD-ROM, to better appreciate the results. The video also shows sequences of trees inserted in virtual scenes. All timings are on a Pentium IV 2.7 Ghz PC.

The first tree we have captured is an oak, shown in Fig. 1. We took 22 pictures of the tree, and generated an equivalent set of masks. The estimation took about 15 mins for a 4-level subdivision of our grid, resulting in 51,000 cells with 8x8 billboards, using a cuttoff transparency threshold of 0.94. The texture generation phase took 30 minutes, and resulted in 150Mb of texture, which after packing (which took 31 min), reduces to 56Mb. We show two new (non input) views of the oak in Fig. 9 (rightmost images); a synthetic rendering of an input view is shown in Fig. 1(d).

The second tree we have captured is a pine, shown in Fig. 9. We took 18 pictures of the tree; estimation took 15 minutes for a level 5 subdivision of our grid, resulting in 110,000 cells, with a cutoff transparency threshold of 0.9. The texture generation for 4x4 billboards took 15 minutes, resulting in 58Mb of texture. After a 17 minute packing process, the final compressed tree requires 22Mb. We show two new (non input) views of the pine in Fig. 9 (left).

Use of the transparency cutoff threshold greatly reduces the number of cells (by a factor of 2 and 8 respectively for the oak and pine), making the resulting trees smaller in memory and faster to display. Initially, some cells incorrectly receive non-zero opacity due to missing information such as the small number of cameras, or occlusion in the images. Use of the threshold makes some of these transparent during reconstruction, improving the visual quality.

8 Discussion

Our method gives very promising results, and we believe that it will be useful as a fast way to capture existing trees and to produce high quality interactive renderings of trees with low polygon count models.

One current difficulty is the need to have a good background to be able to acquire satisfactory alpha mattes. We currently add a uniform color background (a sheet in our case), where possible to help the alpha matte algorithm (see Fig. 4(left)). This helps, but is not always a practical solution. Using the Bayesian approach

Figure 9: From left to right: two new (non-input) views of the captured pine tree and two new views of the captured oak tree.

of [Chuang et al. 2001] may improve the mattes extracted, and we hope that other researchers will develop a more stable alpha extraction approach, perhaps specifically adapted to the case of trees.

Figure 10: Side by side images of the reconstructed oak tree (synthetic view) using thresholds of 0.9, 0.94, 1.0 (left to right).

Currently, the generated textures partially contain the original lighting information. However, since we manipulate the textures in the generation stage, it is clearly possible to develop some approximations which would permit relighting. In particular, the existence of opacity information should probably allow us to approximate light distribution, and perform approximate lighting or relighting calculations. A stochastic approach to shadow removal and normal estimation [Katsumi et al. 1992] could probably be applied. The relatively random nature of the tree leaf distribution should make this easier to achieve than for more structured objects. Exact shadow shapes are harder to identify, and thus the error should be less perceptually evident.

Using a low cutoff threshold makes some cells transparent, but reduces blurring significantly. This is a tradeoff, since depending on the level of subdivision and the number of cameras, existing branches or leaves may be removed. This effect is illustrated in Fig. 10 and in the video. Even when features are removed, for example with threshold 0.9, the tree remains very realistic and remains very similar to the input photographs; however blurring is almost completely eliminated. Investigating heuristics to find the best compromise is a promising avenue of research.

Finally, our approach is best suited for viewing the captured tree at a distance no closer than that of the cameras of the input photographs. Closer viewing would require additional processing. Conversely, when viewing from a distance, multi-level rendering could be used, by computing levels-of-detail for the billboards in the hierarchy.

9 Conclusion

We have presented an approach which allows efficient capture of trees from photographs in three steps. First photographs are taken and alpha-mattes generated; this step also generates the appropriate foreground color to be used for rendering and requires limited user intervention. An automatic reconstruction step creates a recursive grid which contains the opacity values for each cell. We then generate view-dependent textures for each cell, corresponding to the input views, which are attached to billboards placed at the centers of the cells. Rendering is performed using view-dependent texture mapping.

We have presented interactive renderings of captured trees, including the example of a pine tree, which is a particularly hard case to capture and render using other methods, due to the numerous, very fine pine-tree needles.

We believe that this approach, which combines a volumetric opacity estimation with view-dependent texturing will be a promising avenue for future research. Volume rendering solutions for shadows, lighting and relighting will probably be very useful for the development of lighting solutions for our captured trees. We will also develop ways to create varying instances of trees, by modifying certain characteristics, as well as ways to manipulate the resulting models (motion in the wind, scaling etc.). We plan to work actively on investigating the need and methods to improve opacity estimation, as well as improving texture generation techniques.

10 Acknowledgments

Thanks to Frédo Durand for important input on an early version, and Pierre Poulin for comments. This research was supported in part by the CREATE RTD project funded by the 5th Framework Information Society Technologies (IST) Programme of the European Union (IST-2001-34231), http://www.cs.ucl.ac.uk/create/ and by the French Ministry of Research grant ACI MD SHOW. We thank Alias|Wavefront for the generous donation of Maya, and Alexandre Olivier for modeling and capture. Thanks to Audrey Bzeznik for supporting the first author and for feeding the troops.

References

ANDERSEN, A., AND KAK, A. 1984. Simultaneous algebraic reconstruction technique (SART): A superior implementation of the art algorithm. *Ultrasonic Imaging 6*, 81.

BLINN, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. *Proc. SIGGRAPH'82*, 21–29.

BONET, J. S. D., AND VIOLA, P. A. 1999. Roxels: Responsibility weighted 3d volume reconstruction. In *Proc. ICCV-99*, 418–425.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GOTLER, S. J., AND COHEN, M. F. 2001. Unstructured lumigraph rendering. In *Proc. SIGGRAPH 2001*, 425–432.

CHAMBERLAIN, B., DEROSE, T., LISCHINSKI, D., SALESIN, D., AND SNYDER, J. 1996. Faster rendering of complex environments using a spatial hierarchy. In *Proc. Graphics Interface'96*.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2001. A Bayesian approach to digital matting. In *Proc. of IEEE CVPR 2001*, vol. 2, 264–271.

DE REFFYE, P., EDELIN, C., FRANSON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and development. In *Proc. SIGGRAPH 88*, 151–158.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proc. SIGGRAPH'96*, 11–20.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MĚCH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. *Proc. SIGGRAPH'98*, 275–286.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Interactive visualization of complex plant ecosystems. In *Proc. IEEE Visualization 2002*.

DREBIN, R. A., CARPENTER, L., AND HANRAHAN, P. 1988. Volume rendering. In *Proc. SIGGRAPH'88*, 65–74.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proc. IEEE ICCV 1999*, 1033–1038.

FAUGERAS, O. D. 1993. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* The MIT Press, Cambridge, Mass.

KATSUMI, T., KAZUFUMI, K., EIHACHIRO, N., FUJIWA, K., AND TAKAO, N. 1992. A display method of trees by using photo images. *Journal of Information Processing 15*, 4.

KUTULAKOS, K., AND SEITZ, S. 1999. A theory of shape by space carving. In *Proc. ICCV-99*, vol. I, 307–314.

LEVOY, M. 1988. Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (May), 29–37.

MATUSIK, W., PFISTER, H., NGAN, A., BEARDSLEY, P., ZIEGLER, R., AND MCMILLAN, L. 2002. Image-based 3D photography using opacity hulls. *ACM Trans. on Graphics (Proc. SIGGRAPH 2002) 21*, 3 (July), 427–437.

MAX, N., AND OHSAKI, K. 1995. Rendering trees from precomputed Z-buffer views. In *Proc. 6th EG Workshop on Rendering*.

MAX, N. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (June), 99–108.

MAX, N. 1996. Hierarchical rendering of trees from precomputed multi-layer z-buffers. In *Proc. 7th EG Workshop on Rendering*.

MEYER, A., AND NEYRET, F. 1998. Interactive volumetric textures. In *Proc. 9th EG Rendering Workshop 1998*.

MEYER, A., NEYRET, F., AND POULIN, P. 2001. Interactive rendering of trees with shading and shadows. In *Proc. 12th EG Workshop on Rendering, 2001*.

NEYRET, F. 1998. Modeling animating and rendering complex scenes using volumetric textures. *IEEE Trans. on Visualization and Computer Graphics 4*, 1 (Jan.–Mar.), 55–70.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. In *Proc. of SIGGRAPH'84*, 253–259.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. The algorithmic beauty of plants. *Springer, New York*.

QIN, X., NAKAMAE, E., TADAMURA, K., AND NAGAI, Y. 2003. Fast photo-realistic rendering of trees in daylight. In *Proc. of Eurographics 03*, 243–252.

RUZON, M., AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proc. IEEE CVPR'2000*, 18–25.

SEITZ, S., AND DYER, C. 1997. Photorealistic scene reconstruction by voxel coloring. In *Proc. IEEE CVPR 1997*, 1067–1073.

SHIRLEY, P., AND TUCHMAN, A. 1990. A polygonal approximation to direct scalar volume rendering. *Computer Graphics 24*, 5 (Nov.), 63–70.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications 21*, 3 (May/June), 53–61.

SZELISKI, R., AND GOLLAND, P. 1998. Stereo matching with transparency and matting. In *Proc. ICCV-98*, 517–526.

WESTOVER, L. 1990. Footprint evaluation for volume rendering. *Proc. SIGGRAPH'90*, 367–376.

WOOD, D. N., AZUMA, D. I., ALDINGER, K., CURLESS, B., DUCHAMP, T., SALESIN, D. H., AND STUETZLE, W. 2000. Surface light fields for 3D photography. In *Proc. SIGGRAPH 2000*, 287–296.

YAMAZAKI, S., SAGAWA, R., KAWASAKI, H., IKEUCHI, K., AND SAKAUCHI, M. 2002. Microfacet billboarding. In *Proc. 13th EG Workshop on Rendering*.

Appendix A Weight computation for a cell

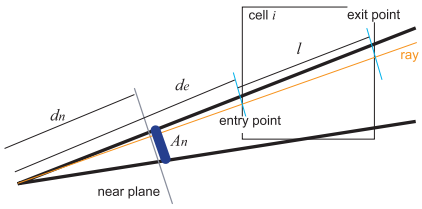


Figure 11: Illustration of the weight computation for a cell.

The four corners of the pixel define a pyramid, which we intersect with the near plane (see Fig. 11, illustrated in 2D for simplicity). This intersection has area A_n; the distance to the near plane is d_n. We shoot N rays through a given pixel. A given ray will intersect cell i at a distance d_e, and the length from the entry to the exit point of the ray in the cell is l. It is assigned an area of $A_s = A_n/N$ at the near plane. We compute the area A'_s of the slice of the pyramid corresponding to the ray using similar triangles: $A'_s/d_n^2 = A_s/d_e^2$ giving $A'_s = A_s(d_e^2/d_n^2)$. We then estimate the volume of the ray/cell intersection as $V = lA'_s$. The weight of the ray will be $w_i = V/V_c$ where V_c is the volume of the cell.