Interactive Point-based Modeling of Complex Objects from Images

Pierre Poulin[a,b] Marc Stamminger[b,c] François Duranleau[a] Marie-Claude Frasson[b]
George Drettakis[b]

[a]Dept. I.R.O., Université de Montréal [b]REVES/INRIA Sophia-Antipolis [c]University of Erlangen

Abstract

Modeling complex realistic objects is a difficult and time consuming process. Nevertheless, with improvements in rendering speed and quality, more and more applications require such realistic complex 3D objects. We present an interactive modeling system that extracts 3D objects from photographs. Our key contribution lies in the tight integration of a point-based representation and user interactivity, by introducing a set of interactive tools to guide reconstruction. 3D color points are a flexible and effective representation for very complex objects; adding, moving, or removing points is fast and simple, facilitating easy improvement of object quality. Because images and depths maps can be very rapidly generated from points, testing validity of point projections in several images is efficient and simple. These properties allow our system to rapidly generate a first approximate model, and allow the user to continuously and interactively guide the generation of points, both locally and globally. A set of interactive tools and optimizations are introduced to help the user improve the extracted objects.

1 Introduction

In the last decade, the realism of images and 3D objects has greatly improved, but it is still quite difficult and time consuming to create models of complex realistic objects, such as those often appearing in nature. Flower beds, plants, trees, tangled rope, crumpled paper, junkyards, etc., are all examples of very complex objects because of the combined presence of numerous surface details, textures, tiny structures, occlusions, holes, discontinuities, specularities, etc. There is however an important need in computer graphics applications (games, film, virtual/augmented reality) for convincing models of realistic complex 3D objects.

For certain classes of such complex objects, specialised modeling methods have been designed with great success; *e.g.*, mountains, waves, plants, cities, etc. However, each method is restricted to its class of objects, and cannot easily be extended to other classes. Image-based methods [16, 10, 3, 17] are quite general, and can produce objects that are visually very complex. They reparametrize radiance information from a large number of input images to display new views of a 3D object with little or no underlying geometry. Unfortunately these representations require lots of memory, and most have difficulty handling mutual occlusion and illumination with other objects. This makes their complete integration in a typical rendering system cumbersome and sometimes impossible.

Reconstruction of 3D objects from photographs is a particularly powerful way to model real objects. With known camera parameters, correspondences between image pixels are used to compute the 3D location and color of a surface. Even though good results have been obtained with these methods (see Section 2), reconstruction of complex and detailed geometry is still cumbersome.

To improve the quality of the extracted objects, we believe it is very important to appropriately involve the *user* in the reconstruction process to guide the spatial and spectral quality of the reconstructed objects. To support such user interaction we need to be able to *interactively* display and manipulate the (partially) reconstructed objects during the modeling process. Good support for interactivity is the main design choice in the system we present here.

Reconstructing complex objects from images also calls for a flexible object representation, which can be *interactively rendered*. 3D points augmented with color or texture attributes are such a general representation that do not require the computation of an intermediate representation such as a triangulation for display. They offer good visual quality when displayed from an appropriate distance, and are also well adapted for efficient visibility culling and levels of detail, facilitating interactivity. Several point-based rendering systems have recently been introduced [18, 21, 26]; we use an adaptation of [26].

Based on these two requirements, user interaction for modeling and efficient display of complex objects, we present an interactive system to extract a *plausible* set of 3D color points from images. The method and the point-based representation are general enough to model any kind of object, but their power is better exploited with complex textured objects. The modeling process is quite

simple: a random 3D point is generated in a 3D zone of interest and projected into all the images to gather its colors. If a sufficient number of these colors confirm that the point is plausible, it is kept. Depth maps from this plausible set of 3D points are constructed at certain intervals in time to ignore images where the generated 3D point would be hidden.

We provide a set of *interactive tools*, which allow the user to guide the iterative solution in critical zones of interest, to modify various quality thresholds and recheck the validity of the set of 3D points, to eliminate outlier 3D points, to fill undesired holes, etc. This user control is a key element of our approach. It is made possible at every stage of the reconstruction process, because we use our point-based representation for efficient rendering of (possibly partially reconstructed) complex objects.

After a brief survey of methods to reconstruct surfaces from images (Section 2), we describe the various elements of our interactive system (Section 3). Results and statistics follow (Section 4). We finally conclude and present directions for further research.

2 Previous Work

In this paper, we address the problem of interactively reconstructing complex objects from images. We will thus not discuss techniques based on 3D scanners (lasers and structured light) or on other sensors. We will also restrict our research to already calibrated images, *i.e.*, images for which the camera parameters have been computed. A vast literature exists on exploiting various cues to reconstruct objects, such as silhouettes, shadows, focus/defocus, motion, shading, etc. We will not specifically use these cues, even though they could improve the reconstruction process by providing additional information. The reader interested in the numerous methods developed over the years should consult books [1, 9, 14, 28], surveys [34, 2], and collections of papers [13] dedicated to these topics.

Reconstructing 3D geometry from one, two, or more photos is a fundamental problem that has received extensive attention in computer vision. The more traditional approach is based on shape-from-stereo [7]. It consists in identifying a feature in one image and searching for a correspondence amongst a number of probable candidates in another image along its epipolar line (projection of the line of sight of the feature onto the other image). Very good results have been obtained in controlled conditions, such as the proximity of viewpoints, spatial or temporal smoothness of adjacent features, or particular shapes. Unfortunately, a good and complete solution still remains a very difficult problem since occlusions, holes, sharp edges, noise, shading, etc., complicate the matching process. In a general 3D scene, there exists an infinity of equivalent shapes that can reproduce the original photos.

An alternative of particular interest to us is volumetric object reconstruction [23, 15, 8, 25]. The approach is based on building a 3D volumetric object that is consistent with the images, rather than directly extracting 3D information from the images. In a typical process, 3D space is voxelized and voxels considered invalid (*e.g.*, outside the silhouette in one image) are *carved out* from the volumetric representation. Besides testing silhouettes, information in each voxel about visibility, transparency, shading, etc., can all be tested with the images for *photo-consistency*. The set of color voxels or a surface fitted to these voxels can then be projected to form new images of the object; impressive results have thus been obtained.

Most of the above techniques are designed to be fully automatic, because the major applications of reconstruction from images have been in object recognition or collision avoidance in robotics. Furthermore, partial, and therefore non-photorealistic results are often quite satisfactory for these applications. Photorealism of the extracted surfaces has only received more attention over the last decade.

Since it is difficult to determine the appropriate quality-level of photorealism for various applications, the elusive additional requirements of photorealism are often better served by integrating the user in the reconstruction process. In fact, reconstruction of architectural 3D models with simple polygonal shapes and high quality textures has greatly benefited in realism from direct interventions of the user. A number of research [5, 22, 19, 12, 11, 24, 6] and commercial systems [30, 32, 33] have been developed on this premise. Unfortunately, only simple polygonal surfaces are extracted from these systems. Manually positioning polygons is clearly a time consuming and sometimes impossible task for complex objects. Even with the help of stereo-based algorithms [5, 12], ambiguities in these complex objects create severe difficulties that a user might be able to correct.

In the next section, we describe how we integrate user intervention at different stages of our interactive system for point-based reconstruction from images. It was inspired and has several similarities with the more traditional algorithms discussed above, but by tightly integrating our point-based representation with the interactive extraction process, we produce a more flexible tool to create improved models of complex objects. Its design also allows easy integration of most past and future improvements in the related algorithms. As an example, we show in Section 4.4 how our limited implementation of voxel

coloring [23] can be integrated in our system.

Another interactive point-based modeling system was recently presented [20]. It uses structured light to generate 3D points in regions of interest by letting the user manipulate the 3D real-world model with respect to a fixed camera. While many goals are identical, their system suffers from the inherent limitations of structured light, *e.g.*, hard to use in large outdoor scenes and in presence of highly-specular surfaces.

3 Point Modeling

Our point modeling basic process starts with an initialization step followed by three main loops (point generation, interactive cleanup, and user-driven refinement). These are executed to produce a set of 3D color points approximating the 3D objects displayed in the images. The algorithm is summarized next. In what follows *sufficient number* is a user-controlled parameter.

1. **Initialization**:
   (a) Take photos (images)
   (b) Calibrate the images
   (c) Define a 3D zone of interest (envelope)
   (d) Draw a mask for each image (optional)
2. **Generation Loop**:
   (a) Generate a random 3D point within the zone of interest
   (b) Validation of 3D points:
      (i) Project the 3D point into each image; gather its associated colors from every *visible* image
      (ii) Keep the 3D point if the color distances are below a *threshold* in a (user controlled) *sufficient number* of images
   (c) Visibility: From time to time, recompute a depth map for each image from the set of 3D points
3. **Interactive Cleanup**:
   - Cleanup by reevaluation
      (a) Interactively increase the value of *sufficient number*, recompute depth maps
      (b) Discard the 3D points failing the new criteria (*sufficient number* and depth maps)
   - Cleanup by rendering:
     For each view (*i.e.*, original image):
      (a) Generate an image of the set of 3D points from this view
      (b) Compare the pixel colors in the original image with those in the generated image
      (c) Discard the 3D points resulting in incorrect image pixels
4. **User-directed Refinement Loop**:
   - Interactively define a smaller zone of interest and restrict point generation, jittering, etc., to this zone, *or*
   - Interactively select a 3D point and use it as a source for 3D point filling, *or*
   - Jitter the 3D points to increase their plausibility, *or*
   - Merge nearby 3D points
5. Display the resulting set of 3D points.

A typical work session is depicted in Fig. 1 for the reconstruction of an intricate rope. An associated web site [31] provides higher resolution images and more detailed statistics. An accompanying video available on this web site demonstrates the interactivity of our system in action.

Note that steps 2 to 4 of our algorithm do not need to be executed sequentially. In practice the user starts with step 2, and then decides which of steps 2 to 4 is the next appropriate step. Since our underlying model, the 3D point set, can be rendered immediately and interactively, the user always has direct feedback about his actions and can quickly react and control the process efficiently. This feedback is possible after only a few seconds of interaction, even of very partial reconstructions of the objects (see Fig. 1).

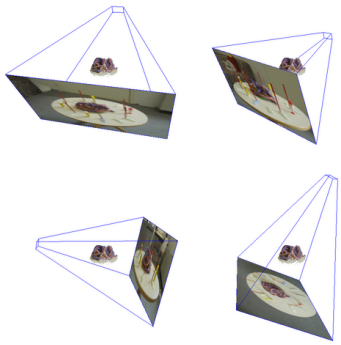
In the rest of this section, we describe each individual step in detail.

3.1 Initialization

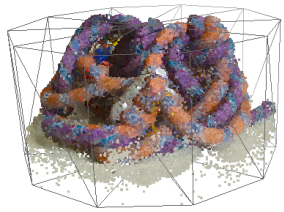
The input of our system is a set of calibrated digital images. We use *ImageModeler*™ from *RealViz* [33] to calibrate a relatively small (2-20) set of photos of resolutions between 1080×720 and 2160×1440 taken from a Canon EOS-DS30 digital camera. We try to preserve object colors as much as possible in the different photos by keeping the same picture exposure and zoom factors. In our examples, the photos are at approximately the same distance from the object, reducing distortion problems. This way, a 3D point from a mostly-lambertian surface appears similar in all photos. *ImageModeler*™ is an interactive calibration system exploiting feature points and epipolar geometry; it has proven quite robust in our scenes. Any other standard camera calibration algorithm could also be used (*e.g.*, [29]).

Once the images have been calibrated, we select a primitive (any closed polyhedron) and transform it in 3D space with *ImageModeler*™ to define an *envelope* around the object to reconstruct. The projections of this 3D envelope in all images must enclose the object to reconstruct. The user can also draw a mask (with *gimp* or *Photoshop*) for each image to indicate pixels clearly outside the objects to reconstruct.

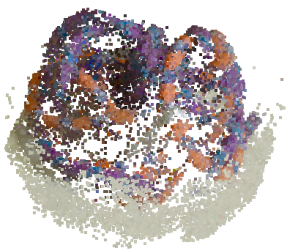
a) 4 of the 14 input images (resolution 2160 $\times$ 1440); pencils help the one hour manual calibration with *ImageModeler™* [33]



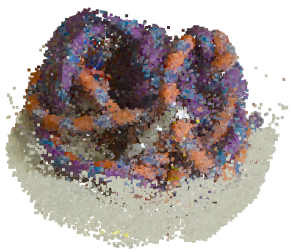
b) Views of four calibrated images with the reconstructed 3D object



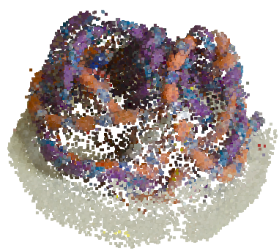
c) The 3D envelope enclosing the object, and thus all the views of the object in each image



d) Initial set of 3D points without depth information; 17,000 3D points generated in 45s, lying mostly on the hull of the object



e) After depth map computation for each image, more points are generated in interior regions of the object (10s), improving color matching (total 45,000 points)



f) Reducing the color threshold and reevaluating all 3D points with new visibility removes weaker (lower quality) 3D points in 3s

g) Automatically generating a total of 86,000 3D points in 45s

h) Manual generation of more 3D points in user indicated regions during 5 min (total of 115,000 points)

i) Applying jittering on the 3D points initially improves the positions of 15% of the points, converging to 2% after 3 min

Figure 1: Typical interactive modeling session, using our system.

3.2 Generation of 3D Points

The basic process generates 3D points uniformly distributed within the envelope. To achieve this distribution, we randomly generate a 3D point in the bounding box of the envelope. To determine if it is interior to the envelope, we count the intersections of the envelope with a ray in a random direction. More systematic generation patterns, such as jittered points in a regular 3D grid (similar to the space carving approach [15, 8, 25]), did not improve the result significantly; random sampling proved much faster (in order of seconds) to produce an initial set of 3D points. As we shall show later (Section 3.4), more goal-directed point distributions give better results.

Validation of 3D points

We must next validate whether each new candidate 3D point can possibly be part of the model. The validation step consists in projecting the point on each image and gathering the colors at the corresponding pixels. An image can be augmented by a mask, automatically or manually painted by the user, to indicate each pixel clearly outside the object to reconstruct. If the image has such a mask and the 3D point projects onto an invalid pixel, the 3D point is immediately rejected. A *depth map* contains a distance (depth) at each pixel. If the image has such an associated depth map and the distance contained in its pixel is shorter than the distance to the candidate 3D point, the 3D point is considered occluded by earlier 3D points in this image, and the image and its pixel color are not considered in the validation.

This results in a list of images that *see* the 3D point and the colors the 3D point exhibits in these images. This list is used to compute a *plausibility* value for the 3D point in order to decide if we should keep the 3D point or not. In general, a point is plausible if one color clearly dominates in the color list. In order to find this dominant color, we compare colors by computing the Euclidean distance in a color space and compare it with a user-defined color threshold. We experimented with the color spaces RGB, CIE xy, CIE Luv, and CIE Lab, but for our purpose we could not detect that one color space outperformed the others. Comparisons in RGB are the fastest, since no color conversion is required.

An alternative is to color quantize all images to a common color palette, and then to compare colors by testing for equality. In a modeling session we typically test a very large number (millions) of candidate 3D points; since the equality test is much more efficient (2-4 times), this choice speeds up the overall process significantly. Moreover because the quantization is done at a preprocessing stage, it can be quite sophisticated, dealing with colors from adjacent pixels, perceptual criteria, etc. This quantization process has a general tendency to reduce shading variations in the images.

In our system, the user selects the maximum number of desired colors to represent all the quantized images. We use *ppmquantall* from the UNIX *ppm tools*. As a general rule of thumb, fewer colors will produce more color matches, and therefore more 3D points will qualify as valid. However, some of these 3D points considered as valid would have been rejected if their original colors had been evaluated. The loss of accuracy in color comparison is therefore compensated by an important gain in generation speed. Many of these false 3D points will be removed at the cleanup stage (Section 3.3). In our tests, we usually work in color quantization mode and select between 32 and 128 colors.

Testing Plausibility

To test whether one color dominates in the color list of a 3D candidate, we compare each color in the list with all others. The color with the most matches is considered the *dominant color*; the ratio of the number of matches over the list size is called *plausibility*. Finally, a 3D point is considered *plausible*, if the plausibility of its color list is above a user-defined plausibility threshold. Other more sophisticated standard variance measures could be applied instead, but this simple test gave satisfying results.

Consider the example in Fig. 2(left), showing four cameras looking at the rope scene of Fig. 1. Point A on the object is visible in all images. Consequently, all four images have the same color in the direction to A. The plausibility of A is thus 100%. Point B, also on the object, is hidden in two images. Nevertheless, the dominant color is still the same, but only with a plausibility of 50%. C is a completely wrong candidate, thus it delivers four different colors, resulting in a plausibility of 25%.

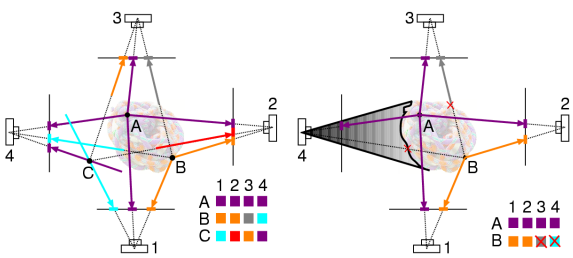


Figure 2: Plausibility tests for points A, B, and C.

Visibility

Point B in the previous example has a plausibility of only 50% because occlusion is not considered properly. If we determine that cameras 3 and 4 do not see point B, we can remove the false colors from the list, so that it only contains two colors. The new color list then has a plausi-

bility of 100%.

To handle occlusions, once a first rough approximate point set has been generated and validated, we compute the depth map of the point set for each image. This is similar to the approach of Szeliski and Golland [27]. Then in the plausibility test, any 3D candidate that is further from the camera than its corresponding depth map value, is considered hidden in the corresponding image and its color is not added to the color list for the point. This is depicted in Fig. 2(right). With a depth map for camera 4, point B cannot be visible from camera 4, the same is true for camera 3. Thus the resulting list only contains two elements, both of the same, correct color.

3.3 Interactive Cleanup

In early phases of a modeling session, false 3D points can be generated. As soon as better depth information is available, a large fraction of these points can be removed automatically. During an interactive session, the user modifies the thresholds and the system reevaluates the plausibility of all previously accepted 3D points, using the most recent depth maps. The system then removes all points below the current thresholds (color, fraction of visible images, etc.). The effect is illustrated in Fig. 1(d)-(g). This iterative process is directly controlled by the user who chooses the thresholds, when to generate new depth information, and when to clean up, according to the results displayed.

We also clean up the point set by rendering the points from the viewpoint of each input image and comparing the rendered images with the original images. All 3D points that result in a significantly different color in an image pixel are discarded. An item buffer with a unique number per 3D point identifies which 3D point is visible in a pixel of one input image. The rendering verification step works best with our texture reprojection method, which will be explained in Section 3.5.

3.4 User-directed Refinement of the 3D Model

The two more automatic techniques described above can significantly improve the set of 3D points and can also be run as a background process when computing power is available. However, a more interactive approach is usually better to refine the set of 3D points where the user feels details are needed. We present a set of four tools, which allow the user to interactively improve the quality of the model.

Interactive Modeling Tool

Our first technique lets the user position a smaller primitive in 3D space, and restrict most of the above described operations to this more confined zone of interest. In our implementation, this is a sphere, which has as center the intersection of the eye ray through the current mouse pointer position with the point set (see accompanying video [31]). The user can move this sphere intuitively, and increase or decrease its size. Usually, the sphere is used to restrict point generation, in order to add points and thus detail to undersampled regions. However, it can also be used to clean up certain regions, or remove large incorrectly classified object parts.

3D Filling Tool

Another type of interactive manipulation lets the user position special sources of 3D point generation, similarly to the approach of Chen and Medioni [4]. These sources generate 3D points according to certain rules, such as following branching structures in trees, or sampling the surface of a polygon. The same manipulation as above is used to select the 3D location of a source, but we also consider the color of this 3D point (or colors of 3D points around it) as a color criteria to *flood fill* the 3D space. The location of 3D points are therefore generated by the flood fill algorithm. A direction will only be visited recursively if its generated 3D point is considered plausible.

Another filling tool consists in displaying the current point set from the virtual camera, and then to try to generate a 3D point in each empty pixel. 3D points are generated along the line of sight of the pixel until a validated point is found (up to a maximum number of attempts).

Jittering Tool

In a loop, we randomly select a 3D point and apply a small user-specified random perturbation (jittering) on its location. We reevaluate its plausibility and keep its new location if its plausibility increases. In practice, this process improves the point set. In the example of Fig. 1, we applied jittering (about 0.1% of the object size) to the point set (h) for 3 min to result in the new point set (i). In the beginning, 15% of the perturbations improve the plausibility, after 3 min the fraction decreased to 2%, illustrating that the set of 3D points quickly settles to a stable configuration, and it becomes less interesting to use jittering after this point.

Merging Tool

The random nature of our 3D point generation as well as the interactive refinements can result in a large number of 3D points within a very constrained neighborhood. The user can reduce this set by clustering 3D points within a specified distance and selecting the best representative(s) according to the color distances and distribution within each cluster. This is implemented by placing a voxel grid of specified resolution into the envelope and keeping the *best* 3D point(s) inside each voxel, or merging the 3D points inside the same voxel according to various criteria (color and spatial distances). This allows us to reduce the

point set to different point densities in different portions of the 3D model.

3.5 Display

To display the reconstructed object, we use OpenGL's `GL_POINTS` primitive. With vertex arrays, we achieve a rate of 6 million colored 3D points per second on an nVIDIA GeForce3, which is enough to render complex objects in real-time, giving the user immediate feedback on the current progress of the reconstruction.

Texture reprojection produces higher quality results. We render the point set and read back the depth buffer. For each pixel P, we then reconstruct the corresponding 3D position p and look for the camera that sees p from the angle closest to the current view. We then project p into this view, read the corresponding pixel color and write it into the frame buffer pixel P. This is a simplified version of previous approaches [5, 3], using only the angular criterion, which is reevaluated per pixel. This step could be optimized, but with our current implementation we already achieve about 2 frames per second, depending on the image size.

4 Results

Various results of our system are provided in this section, as well as on an accompanying video available from the web site associated with this paper [31].

4.1 Synthetic Objects

Synthetic objects can help better understand the behavior and limitations of our algorithms because of the more controlled environment. In Fig. 4A, purely diffuse textured surfaces are rendered with a standard ray tracer at resolution 512×512; we know the *exact* camera parameters. Fig. 4A(bottom) shows a novel view not contained in the original images: (left) a ray traced image of the original objects, (center) the directly rendered point set (more than 30 fps on a GeForce3), and (right) the point set rendered with texture reprojection (software solution at 1.5 fps).

4.2 Soldier and Snack

Figs. 4B and 4C show two further test objects from real scenes. The soldier (Fig. 4B) exhibits smooth surfaces, but also difficult thin structures (flag post) for the automatic generation of our system. The snack (Fig. 4C) has complex (ham) and smooth (plate, cheese, mushroom) surfaces with limited shading, while the grapes are glossier.

4.3 Synthetic Trees

To illustrate the efficiency of our simple algorithm to *fill* 3D samples according to special structures, we produced 10 images of a synthetic tree from our implementation of an *L-system*. In Fig. 3(left), selecting one of the 62 randomly generated points as the seed to our 3D space filling algorithm, produces 15,562 points of the selected color filling the branches in 10s. Because of remaining variations in quantized colors of the images (resulting from shading and antialiasing in the original images), the space filling does not reach the tip of the branches. Selecting a set of colors (instead of a single one) easily helps to fill the rest of the branches. The resulting points are displayed on top of a semi-transparent input image.

In Fig. 3(right), we show the impact of occlusions by leaves. In the rightmost image, after 8,553 points were generated in 7s with our filling algorithm, 11,081 additional points were randomly distributed to fill mostly leaves in about 7 min.

4.4 A Comparison with Voxel Coloring

The flexibility of our interactive system and our point-based representation allows easy integration with other algorithms. We implemented a simplified version of voxel coloring [23, 15]. Voxel coloring requires the reconstructed 3D objects to lie outside the convex hull formed by the cameras, which is the case in the scene of Fig. 1. Instead of carving out voxels, we simply generate points in a rectangle, and process several such rectangles in front-to-back order, thus filling a 3D grid of points. Our depth maps are computed after each such rectangle filling through the 3D grid, and we use our color comparisons. We compare the results in Fig. 4D.

Voxel coloring demonstrates its efficiency due to coherence. The example shows that our approach can lead to similar results, but we have the advantage of being able to interact with the generation process at any point in time.

However, our interactive tools permit direct integration of other generation algorithms. In the accompanying video [31], we show how we use voxel coloring to generate an initial point set and then improve the result using our interactive tools.

Additional advantages of our random-point generation compared to a regular approach are its flexibility and the fact that randomly distributed points can deliver a more desirable visual effect for point-based rendering.

5 Conclusions

In this paper, we have presented our *interactive* system to extract complex objects from images. One major contribution of this system is the central control offered to the user. A set of interactive tools allow the user to guide more automatic reconstruction techniques where details are needed, or neglect regions of unnecessary difficulties. A typical interactive session using the system allows the user to stop a process, modify a value, select another al-

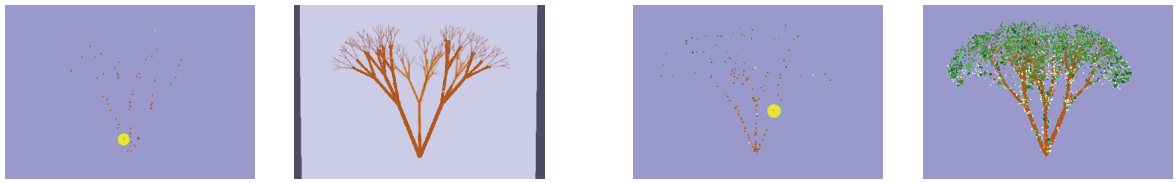


Figure 3: **Synthetic tree with/without leaves.** *10 original images of resolution* 640×443 *quantized to 50 colors; (left) 62 random points, the seed of our filling algorithm (yellow sphere); 15,562 points filled in 10s. (right) 147 random points, seed of filling algorithm (yellow sphere); 19,639 points filled more occluded branches with leaves in 7 min.*

gorithm, merge/refine/revalidate the current result, undo an unsatisfying partial result, etc., each time visualizing the objects being reconstructed in 3D at interactive rates.

The second essential contribution of our system is the use of a point-based representation, which can efficiently display complex and partially reconstructed complex objects. Thanks to this efficient rendering process the user can interactively visualize the behavior of the extraction process, and modify or guide it when necessary. A 3D point is simple and its reconstruction does not require any additional constraint on surfaces, neighborhood, or smoothness. Occlusions between 3D points are also well treated with simple depth maps. Finally, synthetic objects composed of 3D points are easy to integrate within most rendering systems, up to the quality limit of the actual point representation.

By tightly integrating these two key concepts (user interaction and point representation) into our system, it offers a very flexible platform to extract complex objects. Its flexibility also allows interactive experimentation to understand the limits of certain algorithms, and to design new solutions to the challenging problems of modeling from images.

We designed our system with this simplicity in mind. Many of our solutions are built on mature techniques, such as color comparisons, camera calibration, 3D space filling, depth maps, 3D point jittering and clustering, texture reprojection, etc. We did not implement more sophisticated techniques because our emphasis was on implicating the user in the process. The advantage is that any improvement to the techniques used could be easily integrated in our system, leading to equivalent improvements in the results. One important source of improvements, as demonstrated in our limited implementation, should come from a tighter integration of our interaction techniques with the space carving literature [23, 15, 8, 25].

6 Future Work

In its current state, the realism of the objects modeled by our system is very promising, but it is not entirely satisfactory for photorealistic applications. Often, our objects are reconstructed to a given level of quality, but automatic algorithms also seem to settle near this level. User intervention can further improve the model quality but achieving full photorealism is very challenging. In fact in general, we are fundamentally limited by the quality and resolution of the original images. Moreover many other factors other than color and geometry must be extracted from the images to achieve full photorealism.

Nevertheless, there is much room for improvement. We plan to improve the two following aspects in our system. Firstly, our color matching assumes very limited BRDFs, close to lambertian. Shadows and shading variations should be integrated within our color matching test. Other criteria, such as mirror reflection and transparency, could also be integrated into the matching process. Secondly, the information of the original images could be increased with video sequences and high dynamic range images.

Acknowledgements

Financial support for this work was provided by an INRIA invited professorship (P. Poulin), a Marie-Curie postdoctoral fellowship (M. Stamminger), and FCAR graduate (F. Duranleau) and doctoral (M.-C. Frasson) scholarships.

References

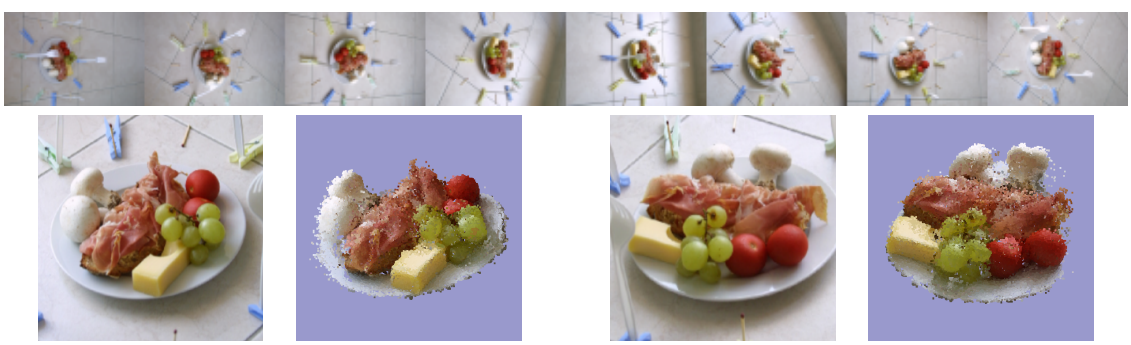
[1] N. Ayache. *Artificial Vision for Mobile Robots – Stereo-vision and Multisensor Perception*. MIT Press, 1991.

[2] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. In *Eurographics 2000: STAR*, September 2000.

[3] C. Buehler, M. Bosse, L. McMillan, S.J. Gortler, and M.F. Cohen. Unstructured lumigraph rendering.

A: Fruit bowl. Top: 13 original images of resolution 512×512. Bottom: A novel view of the scene, (left) original data rendered by a ray tracer, the point set (center) rendered by our system, and (right) rendered by our system with texture reprojection.

B: Toy soldier. Top: 12 photos of resolution 2160×1440; quantized to 64 colors. Bottom: (left and center) The point set rendered from two novel views by our system, and (right) the point set rendered by our system with texture reprojection from the same viewpoint than the center image.



C: Snack. Top: 8 photos of resolution 1440×960; quantized to 64 colors. Bottom: two pairs of images: (left) a photo and (right) the point set rendered by our system.

D: Comparing results with voxel coloring [23]. (left) 305,393 points randomly generated in 4 hours; (center) 200,968 points interactively generated in 45 min; (right) 517,307 points generated by our implementation of voxel coloring in 10 min.

Figure 4: Results

In *Proc. SIGGRAPH 2001*, pages 425–432, August 2001.

[4] Q. Chen and G. Medioni. A volumetric stereo matching method: Application to image-based modeling. In *Proc. of Computer Vision and Pattern Recognition*, pages 29–34, June 1999.

[5] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. SIGGRAPH 96*, pages 11–20, August 1996.

[6] S. Dedieu. *Adaptation of a 3D Model Reconstruction System from Photos to User Knowledge*. PhD thesis, Université Bordeaux I, December 2001.

[7] U.R. Dhong and J.K. Aggarwal. Structure from stereo — a review. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(6):1489–1510, 1989.

[8] C.R. Dyer. Volumetric scene reconstruction from multiple views. In *Foundations of Image Understanding*, pages 469–489. Kluwer, 2001.

[9] O. Faugeras. *Three-Dimensional Computer Vision — A Geometric Viewpoint*. MIT Press, 1993.

[10] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Proc. SIGGRAPH 96*, pages 43–54, August 1996.

[11] E. Guillou, D. Meneveaux, E. Maisel, and K. Bouatouch. Using vanishing points for camera calibration and coarse 3D reconstruction from a single image. *The Visual Computer*, 16(7):396–410, 2000.

[12] A. Hertzmann. Interactive 3D scene reconstruction from images. Technical Report TR 1999-783, Computer Science, New York University, April 1999.

[13] http://iris.usc.edu/Vision-Notes/bibliography/ contents.html 9: 3-D Shape from X and 10: Stereo.

[14] R. Klette, K. Schluns, and A. Koschan. *Computer Vision: Three-Dimensional Data from Images*. Springer, 1998.

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

[16] M. Levoy and P.M. Hanrahan. Light field rendering. In *Proc. SIGGRAPH 96*, pages 31–42, August 1996.

[17] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. Image-based 3D photography using opacity hulls. *ACM Trans. on Graphics*, 21(3):427–437, July 2002.

[18] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proc. SIGGRAPH 2000*, pages 335–342, July 2000.

[19] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Eurographics Workshop on Rendering 1998*, pages 93–104, June 1998.

[20] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, July 2002.

[21] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH 2000*, pages 343–352, July 2000.

[22] Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *Proc. SIGGRAPH 97*, pages 379–388, August 1997.

[23] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[24] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics & Applications*, 21(3):53–61, May / June 2001.

[25] G. Slabaugh, W.B. Culbertson, T. Malzbender, and R. Schafer. A survey of volumetric scene reconstruction methods from photographs. In *Volume Graphics 2001, Proc. of Joint IEEE TCVG and Eurographics Workshop*, pages 81–100, June 2001.

[26] M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry. In *Eurographics Workshop on Rendering 2001*, pages 151–162, June 2001.

[27] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61, August 1999.

[28] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[29] R. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Robotics and Automation*, 3(4):323–344, August 1987.

[30] www.canoma.com.

[31] www.iro.umontreal.ca/labs/infographie/papers.

[32] www.photomodeler.com.

[33] www.realviz.com.

[34] R. Zhang, P.S. Tsai, J. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Trans. on PAMI*, 21(8):690–706, August 1999.