Abstract

We present a new approach to volumetric scene reconstruction which can produce accurate models from turntable image sequences. Instead of an epipolar plane image (EPI) volume, we consider a function on the 4D spatiotemporal volume valued with the intensity back projection of the camera (time) to the particular voxel. Using an optical flow technique we compute the local orientation of this spatiotemporal image and decide on occupancy based on the relative orientation between the viewing ray of the voxel at the particular time and the local image structure. Our method does not require a background compensation like the silhouette-based methods and is comparable in performance with space carving. The visual quality in rendering of the established model is dramatically increased by using the number of occupancy candidates for a voxel along all cameras as a transparency measure.

1. Introduction

Acquisition of 3D models from scenes has been a well known problem with already many successful solutions already reaching the commercial market. Such solutions usually rely on active illumination with structured patterns and with one exception [6], they assume stationarity of the scenes. Among the passive visual methods, we differentiate between bi-(tri-)nocular stereo methods (see survey in [12]) and volumetric approaches (see [16] and survey in [4]) based on surround camera clusters. The latter are of particular interest since they can produce full 3D point-clouds unlike the 2+1/2D maps obtained by conventional stereo. Voxel coloring [13] and space carving [9, 3] seek to reconstruct scenes based on photo consistency. Both of these methods iteratively calculate voxel visibility and use a color consistency criterion on the back projections of visible voxels to corresponding images. The appearance of the resulting models depends on the quality of calibration and scene compliance with the lighting model (which is typically Lambertian). If the scene is stationary, instead of using a camera cluster we can controllably move the camera or the object. An example of such a motion is a single axis rotation like a turntable motion. Turntable motions have been mainly used for silhouette-based reconstruction [17, 2, 15] where the result is a convex approximation of the real shape, called the visual hull. These techniques can create either a 3D model or employ image-based rendering [11] to produce interpolated views from a set of distinct viewpoints. These methods do not reconstruct concave objects and rely on controlled scene backgrounds [10]. In case of continuous camera motion, such a sequence becomes the circular version of an epipolar plane image collection [1, 8]. As we know, the principle of epipolar plane image analysis is the computation of depth from the slope of edges detected the spatiotemporal image. The underlying principle of either camera clusters or dense image sequences is the assumption about color constancy of the appearance of scene points, expressed as photo-consistency or as brightness constancy – the latter is a usual assumption in optical flow estimation.

In this paper, we introduce a novel algorithm for volumetric reconstruction from turntable sequences. Our domain is a rectangular volume discretized in voxels. Each voxel is assigned with as many gray-values as the number of cameras by computing the intensity of pixels it projects onto. We can think of the world as a collection of parallel (for example vertical) planes and for each plane we compute the collineation of every camera image. Thus we build a 4D spatiotemporal domain which at first sight is a highly redundant representation. However, voxels with the same intensity lie at the same position among the temporal (camera domain) and this brightness constancy along an a priori known direction indicates correspondence.

With this spatiotemporal representation, the reconstruction problem becomes a signal detection problem. In particular, it becomes a test of whether the projection of the local orientation at a specific direction vanishes or not. Local orientation of multidimensional signals has been defined in the past [5, 7] to be the eigenvector corresponding to the maximal eigenvalue of the local inertial tensor in the space domain. Translated to the spatiotemporal domain, it becomes a matrix summarizing the distribution of the spatiotemporal gradients in a neighborhood. The directions we are interested in are tangent planes at a surface swept by a ray to the same scene point over time, which in the case of a turntable sequence turns out to be of helicoidal shape.

The local orientation test indicates whether at the particular voxel and at the specific timepoint there is a match, meaning that this voxel is occupied. To finally decide the occupancy of a voxel, we inspect how many times the local orientation test is successful. A very useful by-product is that this “vote” count can be used as a transparency value...
during rendering.

Our approach resembles several approaches with respect to different aspects. In the sense that it is volumetric and uses the back projected images, the data structures are similar to space carving with which we also experimentally compare. The set-up is also ideal for space carving [9] or generalized voxel coloring [14] since it is expected that photo consistency holds because the images are taken by the same camera. However, because we use the local distribution of gradients, brightness constancy can be only approximated. Because we inspect voxels locally in time we do not have to reason about visibility. The disadvantage of our approach is that we need a dense sequence to guarantee a meaningful differentiability (thirty poses for a full circular motion). In the sense that we use a local technique the approach is similar to epipolar plane image analysis but we work on the voxel space and make use only of a signal detection test while epipolar plane image analysis works in the image domain and uses the exact value of the flow (or local orientation) to compute depth.

Summarizing, our approach is the first that makes use of the spatiotemporal orientation in voxel space and thus combines the advantages of working volumetrically with the algorithmic simplicity of signal detection tests.

2. Spatiotemporal Representation

At the heart of our approach to shape recovery lies a spatiotemporal map which contains the intensity of each world voxel when viewed from each camera. We call this map $\Phi$. To construct $\Phi$, the 3D world is extended by an extra dimension which contains the gray values of projections of each voxel into each image. For the purposes of this paper, we only consider imaging systems in which the camera moves in a circle at constant angular velocity. We will later suggest a way of extending this method to other motions as well.

2.1. Notation

We define the function $\Phi : D \times \mathbb{R} \rightarrow \mathbb{R}$, where $D$ is the bounded volume in $\mathbb{R}^3$, to map the world point $(x, y, z)$ at time $t$ to image intensity. We also define the image sequence $I : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$ to be the map from image coordinates $(p_x, p_y)$ at time $t$ to intensity. We restrict the movement of the camera to a circle $C$ located on the $xy$ plane in the world with center at the origin. The camera is calibrated with respect to this coordinate system and calibration is given as follows: $M_t = M_{int}(t)M_{ext}(t) = [M_t \hat{m}_t]$.

2.2. Properties of $\Phi$

The world function can now be computed as follows: for all world points $v = (x, y, z, 1)^\top$ and the image sequence $I$ (with corresponding calibration information) taken from positions on $C$,

\[
\lambda \begin{pmatrix} p_x(t) \\ p_y(t) \\ 1 \end{pmatrix} = M_{int}(t)M_{ext}(t) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
\]

\[
\Phi(x, y, z, t) = I(p_x, p_y, t).
\]

Let us first assume continuous image and world coordinates and examine the internal structure of $\Phi$. Each point of an image $I_t$ projects a ray

\[
\begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \lambda M_t^{-1} \begin{pmatrix} p_x(t) \\ p_y(t) \\ 1 \end{pmatrix} - \hat{M}_t \hat{m}_t
\]

through $\Phi(x, y, z, t_1)$ from the optical center of the camera at $t$. This ray will have a uniform color which is the color of the original image point. The ray will intersect the object surface or the image background at a point $v_s \in D$. Assuming a Lambertian object surface, all other cameras from which the surface point $v_s$ is visible, will produce rays of identical color through the same point.

Let us assume for a moment that $v' \in D$ is the only point belonging to the object. What does the projection of this point onto $\Phi$ look like over time? Since each camera will produce a ray through $\Phi$, and the camera is rotating, the resulting surface will be a ruled helicoid embedded in four dimensional domain of $\Phi$. The precise structure of $\Phi$ is easier to visualize if we consider the case when $v' = (a, b, 0)$. In this case, the ray projected by each camera will lie on the $xy$ plane, and the helicoid formed in the subspace $\Phi(x, y, 0, t)$ is shown in Figure 1. The helicoids exhibited by the world points are unique since they differ in either the axis of rotation (defined by the $xy$ coordinate) or the generating line slant (defined by the $z$ coordinate). Finding the axis of rotation of each helicoid is equivalent to recovering the 3D point. As other points populate the scene, the internal surfaces will be interrupted due to occlusion, but the inherent local structure of the helicoid sheets should be preserved given a dense enough image sequence.

3. Surface Voxel Detection Criteria

We are looking for a local property of $\Phi$ which will allow us to decide if a voxel belongs to the object surface. Let us consider ways of detecting a surface point $p$ given $\Phi$. The space carving approaches look for color consistency of a voxel among a set of cameras and progressively discard any voxels that do not fit this criterion. This amounts to limiting the color variation of each voxel, given its visibility.

But if we consider the helicoidal structure of surface point projections, we can establish a criterion for surface detection which is local to the voxel neighborhood, not just
Figure 1: Plot of an \( x, y, t \) subspace of \( \Phi \) showing a helicoid produced by occupied voxel at \((x = 0, y = 0, z = 0)\).

one voxel. In essence, we want to find the axis of rotation of each helicoid that corresponds to an object point. As we travel from the camera origin towards the surface point, we note that the orientation of the helicoid sheet changes from being almost level with the \( xy \) plane close to the camera, to being orthogonal to it at the object surface point. For an oriented helicoid, the direction of the \( xy \) orientation changes direction from positive to negative (or vice versa, depending on the direction of camera movement) at the point belonging to the surface.

Figure 2: Two slices through \( \Phi \): (a) shows the slice \( \Phi(x, y = 0, z = 0, t) \), and (b) shows \( \Phi(x, y = -10, z = 0, t) \).

If we estimate the local orientation of the helicoid sheets everywhere in \( \Phi \), we can then follow a ray from each camera origin and determine the axis of rotation corresponding to this ray by looking for a point with orientation perpendicular to the \( xy \) plane and that changes direction of the orientation.

Since the \( z \) component of the orientation is determined entirely by the camera calibration, and does not depend on motion of the camera or the object location, we do not need to compute it.

Local orientation estimation requires texture to be present on the object surface, but, since there is likely to be texture on the occluding interface between the object and the background, it is possible to reconstruct uniformly colored convex objects.

4. Surface Detection

The surface detection method can be split into two stages: local and global.

The local stage involves computing the local orientation of a neighborhood around each point \( w \in \Phi \). This can be computed by eigenvalue analysis of the inertia tensor

\[
T = \sum_{\text{neighborhood of } w} \begin{pmatrix}
\Phi_x^2 & \Phi_x \Phi_y & \Phi_x \Phi_t \\
\Phi_y \Phi_x & \Phi_y^2 & \Phi_y \Phi_t \\
\Phi_t \Phi_x & \Phi_t \Phi_y & \Phi_t^2
\end{pmatrix}
\]

(3)

where \( \Phi_q = \frac{\partial \Phi}{\partial q} \).

Let \( \lambda_1, \lambda_2 \) and \( \lambda_3 \) be the eigenvalues of \( T \). We are looking for the neighborhoods in which the helicoid sheets exhibit locally planar structure which corresponds to the case \( \lambda_1 \gg \lambda_2 \approx \lambda_3 \). In this case, the eigenvector corresponding to \( \lambda_1 \) will give us the normal to the planes through the neighborhood.

The partial derivatives \( \Phi_x, \Phi_y \) and \( \Phi_t \) are computed by convolution of \( \Phi \) with Gaussian derivative operators \( D_x, D_y \) and \( D_t \). The neighborhood sums are also found by convolution with a smoothing operator \( G \).

Once the local structure has been determined, we can find for each ray a point in the world most likely to be occupied. To accomplish this we scan along each ray to find the 180° orientation direction change for the greatest vector magnitude. It is clear from the helicoidal structure of \( \Phi \) that the \( xy \) orientation on a ray will be normal to the ray. The exact phase of the orientation is determined by the known motion of the camera and the location of a point on the ray with respect to the object surface. Since we know the camera motion, we can determine which points on the ray may correspond to the outside and the inside of the object. As we showed in the previous section, the \( xy \) magnitude of the orientation vector corresponds to proximity to the object. In our implementation, however, we use the derivative operator \( D \) to quickly compute the most likely surface location.

5. 3D Scanning System

5.1. Image Acquisition

Our image capture system shown in Figure 3 consists of a Pointgray Dragonfly, which is a color camera with a firewire
interface, attached to the Cyberware laser scanner gantry, and an acquisition PC. The camera was manually calibrated with respect to the axis of rotation of the system. The radius of the circle was about 60 cm and the reconstructed volume was about 30 cm³.

Image sequences were acquired by rotating the scanner around the object at a constant velocity and triggering the camera at equal time intervals. The image acquisition process took 45 seconds per sequence. No restrictions were placed on the background objects or light placements in the room and, consequently, some shadows, highlights and flares are visible in the images.

5.2. Reconstruction Algorithm

Let the voxel volume \( D \), containing the object to be reconstructed. Let \( t_d, d = 1..p \) be the discretized time component of \( \Phi \). This means that we are working with \( p \) evenly spaced cameras.

For our implementation of the method described in previous sections, we make the following simplification which makes the method faster, more parallelizable, and requiring less runtime memory than the full implementation: for a voxel volume \( D' \), we process each \( z = c \) slice separately.

The steps of the algorithm are as follows:

Voxel Space Reconstruction Algorithm:

1. Acquire image sequence \( I \) with \( p \) images.
2. Allocate a \( z \) slice \( S \) of size \( m \times n \times p \).
3. for \( c = 1..r \)
   - Compute each entry in \( S(x, y, t) \) by equation (1), where \( z = c \).
   - Let \( S_x = \text{conv}(S, D_x) \), \( S_y = \text{conv}(S, D_y) \), \( S_t = \text{conv}(S, D_t) \).
   - Compute \( T \) components \( S_2^x, S_x S_y, S_y, S_y^2, S_x^2, S_x S_t, S_y S_t \).
   - Compute neighborhood sums by convolution of \( S \) with \( G \) operator.
   - for \( x = 1, \ldots, m \), \( y = 1, \ldots, n \), \( t = 1, \ldots, p \)
     - Let \( e = [u, v, w]^{\top} \) be the eigenvector corresponding to largest eigenvalue of matrix in (3) at \((x, y, t)\).
     - Let \( \theta \) be the angle between the ray from the camera center to the origin and the \( x \)-axis.
     - Compute \( O(x, y, t) = \sqrt{u^2 + v^2} \). \( \frac{\text{atan2}(u, v) + \theta}{\text{atan2}(u, v) + \theta} \)
   - Output \( R(c) = \sum_k \text{conv}(O, D_0) \), where \( D_0 \) is a derivative operator rotated by \( \theta \).

The volume \( R \) of size \( m \times n \times r \) now contains the reconstructed volume where large values correspond to high confidence in a voxel being on the surface of the object, since the final summation adds the filter responses of all cameras for each voxel.

6. Point Cloud Visualization

Our current implementation does not generate a polygonalized object surface. The algorithm creates a voxel volume where high values of voxels indicate many high filter responses, which corresponds to confidence in the voxel. While we can simply threshold the values to leave only a small subset of volume points, we chose to visualize the uncertainty values as transparency factors. This allows us to apply a lower threshold. In this case, a mass of low probability voxels (such as corresponding to a low texture region) can be highly visible, but a small region of low confidence,
will be barely noticeable. This allows us to quickly generate new views without knowing the complete surface geometry.

7. Results

We present the reconstruction results for three image sequences that we captured with our system. The objects include a teapot, a model tank and a human head. Each image sequence contained ninety images and a $200 \times 200 \times 200$ voxel volume was reconstructed for high resolution models and $100 \times 100 \times 100$ for low resolution. We have experimented with using between thirty and ninety images and provide a variety of reconstruction results in Figures 6, 7, 9, 10 and 11. The following table summarizes the processing times for our compiler optimized C language implementation of the algorithm running on a dual Pentium 4 2.4GHz PC. It shows the approximate time it takes to produce voxel volume $R$ from an image sequence. Additional processing time is required for visualization of the results.

<table>
<thead>
<tr>
<th></th>
<th>$100 \times 100 \times 100$</th>
<th>$200 \times 200 \times 200$</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 images</td>
<td>72 s.</td>
<td>305 s.</td>
</tr>
<tr>
<td>90 images</td>
<td>6 min.</td>
<td>26 min.</td>
</tr>
</tbody>
</table>

Table 1: Running times for different sequence lengths and voxel volumes.

We provide a comparison of our results with the results of space carving [9]. Since we use the same camera to take all of our images, the scene should obey the color consistency constraint up to its non-Lambertian properties. We use standard deviation of intensity values as a color consistency function as suggested in the paper, however, the paper does not suggest a way to quantify the threshold on this constraint for a particular image sequence. Our experiments showed that different sequences require different thresholds to achieve good results, and that relatively small deviations can result in space being either completely carved out or not carved enough. We adjusted the threshold to produce what we thought were the best results for a given data set. In contrast, the results in Figure 4 for the implementation of our method are produced with the same kernel sizes, changing only voxel resolution and number of views. We show the raw output for the $z = 0$ slice through $R$. Higher intensity corresponds to higher confidence in the surface point.

In addition to figures we also include four movies of the reconstructed objects. The following table contains descriptions for each movie in the supplementary file.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head-hr.avi</td>
<td>Head model, 90 images, $200 \times 200 \times 200$</td>
</tr>
<tr>
<td>Pot-hr.avi</td>
<td>Pot model, 90 images, $200 \times 200 \times 200$</td>
</tr>
<tr>
<td>Pot-30.mpg</td>
<td>Pot model, 30 images, $200 \times 200 \times 200$</td>
</tr>
<tr>
<td>Tank-lr.mpg</td>
<td>Tank model, 30 images, $100 \times 100 \times 100$</td>
</tr>
</tbody>
</table>

Figure 4: Results from our method and space carving for the $200 \times 200$ voxel plane at $z = 0$ of the teapot sequence. (a) and (b) show reconstruction from ninety images, (c) and (d) use thirty images and (e) and (f) use thirty images and the calibration matrix of each camera is rotated by $5^\circ$ around its $x$ axis and $5^\circ$ around $z$ axis, introducing a bias.

8. Summary and Conclusions

We introduce a novel scene reconstruction method based on spatiotemporal orientation. We use the implementation of the method to create accurate high resolution models of
Figure 5: Four of ninety “head” sequence input images.

Figure 6: A $200 \times 200 \times 200$ reconstruction from ninety images.

Figure 7: A $100 \times 100 \times 100$ reconstruction from thirty images.

Figure 8: Four of ninety Pot sequence input images.

Figure 9: A $200 \times 200 \times 200$ reconstruction from ninety images.

Figure 10: A $200 \times 200 \times 200$ reconstruction from thirty images.
real world objects. Unlike other approaches, our method does not require background subtraction, does not rely on correlation or optical flow in images and is less sensitive to calibration than Space Carving. However, unlike other methods, it requires continuous camera motion.

There are some implementation related advantages to our method. Since most of the processing is done with convolution and image warping, the time to reconstruct a scene can be reduced with aid of SIMD processor features and hardware assisted texture mapping. Our approach is not iterative in nature and is not scene dependent, so we can guarantee that reconstruction will take a fixed amount of time regardless of scene complexity. The local nature of the algorithm makes it a good candidate for parallelization. Different intervals of cameras and different regions of the voxel space can be reconstructed by different processors with merging at the very end.

In our experiments we reconstruct different scenes with different camera and voxel space resolutions using the same program for all of them. Thus it is the only method to our knowledge where no thresholds have to be set to ensure high reconstruction quality.

References


