IMAGE-BASED RENDERING WITH DEPTH INFORMATION USING THE PROPAGATION ALGORITHM

Ha T. Nguyen and Minh N. Do

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign
{hanguyen, minhdo}@uiuc.edu

ABSTRACT

This paper proposes a new approach for the image-based rendering (IBR) problem. IBR has many potential applications such as remote reality, telepresence in which traditional computer graphic techniques require high computational complexity. Our algorithm proactively propagates all available information from actual cameras to virtual cameras, using a depth availability assumption. This process turns the IBR problem into a nonuniform interpolation problem at the virtual camera image plane, which can be done efficiently at once for all image pixels. Experimental results show the proposed algorithm has low computational complexity and produces accurate rendering, especially around object boundaries-where most of existing methods fail.

1. INTRODUCTION

Image-Based Rendering (IBR) is an emerging technology which enables the synthesis of novel realistic images of a scene from virtual viewpoints, using a collection of available images. The applications of IBR can be found in various situations such as virtual reality, telepresence, thanks to the complexity and performance advantage over *model-based techniques*, which bases on complex 3-D geometric models, material properties and lightening conditions of the scene. A detailed survey of the current IBR techniques can be found in [11, 13].

Although some existing IBR techniques do not use the geometric information of the scene as the input, most of them do try to reconstruct (explicitly or implicitly-e.g. via feature correspondences between images) the 3-D scene as an intermediate step to render virtual images. Light-field [8] and Lumigraph [7] are exceptions, but they require many images to compensate the geometric information. The most commonly used approach, called ray-tracing, is to trace a ray from the camera origin to each pixel in the virtual image plane and try to find the point where this ray hits an object surface in the scene [6, 12, 4]. The intensity of this point is then interpolated using information of the neighboring pixels in the image collection. The ray tracing approach hence is very computationally complex, and has a limited rendering quality around the object boundaries [6, 12].

Our approach, which we call the *Propagation Algorithm*, is different from existing methods in two aspects, namely: 1) we assume that the *depth information is available* for some or all image pixels; and 2) we do not use the ray-tracing approach and intensity interpolation on pixel by pixel basis, but we rather proactively propagate all the available information to the virtual image plane first, then *do the intensity interpolation only once for all the pixels*. For the first aspect, depth information is available in synthesis images, and this assumption is also made for realistic images as well [6, 10, 3]. The assumption is justified by the availability of range camera technology (like 3D range scanner [9]). Another reason is that the existing correspondence techniques [5] can provide the depth of certain image pixels (e.g. image features like corners, edges). Moreover, in some applications we can insert makers in the scene to keep track of the depth of certain points. The second aspect is motivated by a fact usually pointed out in literature that the IBR problem is a nonuniform interpolation problem [13]. However, our algorithm does not locally interpolate pixel intensities (using nearby cameras and/or nearby pixels), but it follows the nonuniform interpolation framework [2], which improves the rendering quality, especially around the object boundaries, while maintaining a low computational complexity.

The remainder of this paper is organized as follows. Section 2 sets up the problem. Section 3 describes our algorithm and experimental results in the case where all the pixels are associated with depth information. Section 4 presents the algorithm when we have the depth information for only a strict subset of pixels. Finally section 5 concludes our paper and discuss future work.

2. PROBLEM STATEMENT

To illustrate the main idea of our propagation algorithm, we consider a simple setting called 2-D light field [8, 7, 6] as described in the Fig. 1. The algorithm can be naturally extended to more general settings of the plenoptic function [1]. All the actual and virtual cameras are pinhole cameras and lie on the camera line $d = 0$ with parameter u; and all the focal lines are on the line $d = f$ with parameter v. Each ray of the parameterization is uniquely determined by a pair (u, v), where u determines the camera position, and v determines pixel's local coordinate on the image plane of camera u. For this simple setting, we will present our algorithm and its result for 1-D signals (e.g. image lines).

We assume that we have as input intensity images $\{I(u, v) : (u, v) \in \mathcal{S}_\mathcal{I}\}$ and depth information $\{D(u, v) : (u, v) \in \mathcal{S}_\mathcal{D}\}$ for some set $\mathcal{S}_\mathcal{I}$ and $\mathcal{S}_\mathcal{D}$. The distribution of intensity cameras and range cameras (depends on $\mathcal{S}_\mathcal{I}$ and $\mathcal{S}_\mathcal{D}$) could be different, and typically $\mathcal{S}_\mathcal{D}$ is a subset of $\mathcal{S}_\mathcal{I}$ ($\mathcal{S}_\mathcal{D} \subset \mathcal{S}_\mathcal{I}$) due to the advantage of the intensity cameras on the availability and the cost over the range cameras. As output we would like to determine the intensity images $\{I(u, v) : (u, v) \in \mathcal{S}_\mathcal{R}\}$, where $\mathcal{S}_\mathcal{R}$ is a set of rays to be rendered, which usually includes $\mathcal{S}_\mathcal{I}$ ($\mathcal{S}_\mathcal{I} \subset \mathcal{S}_\mathcal{R}$). This implies

This work was supported by the National Science Foundation under Grant ITR-0312432.

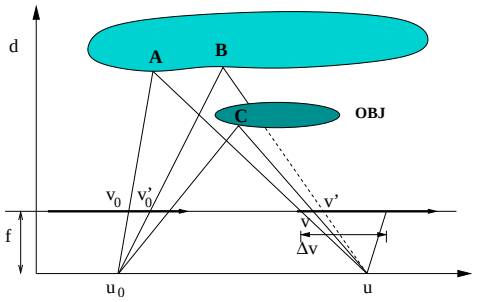

Fig. 1. A simple scene setting. Not all the information of actual camera u_0 is directly used to render the image at virtual camera u. In the figure the intensity at A is useful, but at B it is not as it is occluded by OBJ!

that the position of all the actual and virtual cameras are available (calibrated cameras). We furthermore assume that:

1. **Lambertian surfaces:** the radiance leaving a surface point is independent of the angle. This means that the intensity of a point in the 3D scene is the same seen from cameras at different positions [5]; and
2. **Smooth surfaces and surface intensity functions:** The surfaces and surface intensity functions of the scene are sufficiently smooth (e.g. bandlimited functions). This implies that these functions can be interpolated using the values of a finite number of samples [2].

3. PROPAGATION ALGORITHM WITH FULL DEPTH AVAILABLE

In this section we present our algorithm in the case $\mathcal{S}_{\mathcal{I}} = \mathcal{S}_{\mathcal{D}} = \mathcal{S}$, i.e. for every pixel we know both the intensity and the depth information.

3.1. Overview of the algorithm

Ray tracing techniques in the traditional computer graphics and computer vision communities try to render novel images pixel by pixel. When we render a ray r at a pixel of the virtual camera, we first try to recover the depth associated with this ray, i.e. where this ray hits an object surface in the scene. To avoid the 3-D geometrical reconstruction, techniques like local color consistency (comparing the intensity of corresponding pixels [12]) or depth matching (comparing the depth of corresponding pixels [6]) are proposed to find the depth. Once we get the depth, we can trace back to the actual cameras to get the corresponding pixels. The rendered intensity then is computed by interpolating the corresponding pixels found from actual images. Ray-tracing techniques hence are very computationally complex, and even with the help of depth information, the rendering quality is still limited by depth discontinuities around object boundaries [6].

In our propagation algorithm, we adopt a different approach. For each virtual camera we try to get all the relevant information available and relevant first, then we do the intensity interpolation once for all the pixels. There are several advantages of this approach. First, such proactive use of available information allows us to change the number of cameras during the rendering (e.g. in real time system) without considerably modifying the algorithm. Second, our approach allows occlusions to be detected in a simple way, which reduces the computational complexity. Third, by doing the interpolation at the end, we do not calculate the intensity on the pixel by pixel basis, but we get the whole virtual image at once, which also reduces the computational complexity. Forth, doing the interpolation at the end can also help by taking into account more information than considering only the intensity of neighboring pixels in existing methods (e.g. points in the same surface), and therefore provide more coherent image pixels. Finally, our approach also allows us to obtain the intensity for pixels at the image perimeter, where ray-tracing techniques fail because of the lack of pixel correspondence.

The proposed algorithm includes the following steps, which are illustrated in Fig. 2 and detailed in the following subsections:

1. **Information Propagation:** starting from the actual images, using the depth information we propagate all the intensity information to the virtual cameras. For example, in Fig. 1, the intensity at pixel v_0, v_0' of the actual camera u_0 is propagated to the point v, v' of the virtual camera u.
2. **Occlusion Removal:** remove all the points in whose neighbor there is at least a point with noticeable smaller depth, which are likely occluded at the virtual camera. For example in the Fig. 1 the point v' is propagated from the pixel v_0', but then it should be removed because it is occluded by object OBJ.
3. **Intensity Interpolation:** interpolate the remaining data using an appropriate kernel function depending on the characteristics of the scene.

3.2. Information propagation

In the Fig. 1 the image of the point A in cameras at u and u_0 are v and v_0. The shift of the pixel positions is:

$$\Delta v = -\Delta u f/d, \tag{1}$$

where f is the focal length of the cameras, and d is the depth of A. This equation shows that if we have the depth information at the pixel v_0, we can propagate the intensity at this pixel to the point $v = v_0 + \Delta v$ of the virtual camera. In general configurations, equation (1) can be easily modified, and Δv will depend on the extrinsic and intrinsic parameters of both cameras. The information propagation step is done as following for each virtual camera u:

```
% Propagate information to virtual camera u
For each (u0, v0) ∈ S
  v = Vu(u0, v0) = v0 − (u − u0)f/D(u0, v0);
  I(u, v) = I(u0, v0);
  D(u, v) = D(u0, v0);
Endfor
```
(2)

The result is a set of points:

$$\mathcal{P}_u = \{v = V_u(u_0, v_0) : (u_0, v_0) \in \mathcal{S}\}, \tag{3}$$

which are associated with intensity and depth information $I_u(v) = I(u, v)$ and $D_u(v) = D(u, v)$ propagated as in (2).

This process can be done camera by camera independently, so that our algorithm can work with an arbitrary number of actual cameras, and this number can be modified anytime during the running of the algorithm without the algorithm being considerably modified. Note that:

1. At this stage we still work on continuous signals, and so we do not pixelize the position v.
2. All the information related to v is computed on the local coordination of the virtual camera.

At this stage, all the information are propagated without considering any occlusion. In Fig. 1 the pixel v_0' will be propagated to the pixel v' though it is occluded by the object OBJ. The following subsection will present how we can prevent occluded points like v' from being used in the interpolation step.

3.3. Occlusion removal:

Our occlusion removal step is based on the following observation. Let us consider a point (for example B in Fig. 1) occluded at the virtual camera, if enough information is available then there should be a point (like C) in its ϵ-*neighbor* (in local v axe), with σ-*noticeably smaller* depth. Vice-versa, if a point B has a ϵ-neighboring point with σ-noticeably smaller depth, then *almost surely* this point belongs to a different surface than B, and B will be occluded by this surface.

In the observation, by *B is a ϵ-neighbor of C* we mean the distance between images of B and C at the virtual camera are within a distance ϵ, and by *the depth of C is σ-noticeably smaller than the depth of B* we mean the depth of C exceeds the depth of B by at least σ. In our experiment we used $\epsilon = 0.8$ unit pixel and $\sigma = 0.05 \times D_u(c)$.

Following the observation, we remove all the points in whose ϵ-neighbor there is at least one point with σ-noticeably smaller depth, as following:

```
% Remove occluded points
For each b ∈ P_u
  If ∃c ∈ P_u : |b − c| < ε, |D_u(b) − D_u(c)| > σ
    P_u = P_u − {b};
  Endif
Endfor
```
(4)

3.4. Intensity interpolation

The intensity interpolation step is processed at the end when all the information relevant to the virtual camera is in hand. Let $\mathcal{X}_u$ be the remaining points after the occlusion removal step. As we assume that the intensity function is smooth, with enough remaining points we can reconstruct the intensity function, and so that the image is rendered by resampling the intensity function at the pixel positions (pixelization). In our experiment we use the cubic spline. The interpolation step for each virtual camera u is as following:

```
% Interpolate the intensity function;
I_u = Interpolate(X_u, I_u(v)|_{v∈X_u})

% Pixelize I_u to get the rendered image at u
For each v : (u, v) ∈ S_R
  I(u, v) = I_u(v);
Endfor
```
(5)

One issue of the interpolation step is that usually the intensity function is only piecewise smooth, and so the interpolation step will produce some variation around the discontinuities (i.e. edges). To keep the variation small, in the experiment we double our data before the interpolation. This technique, instead of interpolating the function's value at n points $x_1, x_2, \ldots, x_n$, interpolates the function's value at $2n$ points $x_1, x_1+\mu, x_2, x_2+\mu, \ldots, x_n, x_n+\mu$ for some small value of μ ($\mu = 0.000001$ in the experiment). The function's value at $x_i + \mu$ is given by the function's value at x_i. To overcome this problem at the edges, we can use edge detection techniques [5], and interpolate the intensity function piecewisely.

We observe that our approach in fact turns the IBR problem into a classical nonuniform interpolation problem, which offers a rigorous investigation of the rendering quality and error, as well as opens new room for applying existing mathematical tools [2].

3.5. Experimental results

To illustrate our algorithm step by step, we adopt a synthesis configuration. As the camera line is parallel with the rows of the image planes, we can do our rendering row by row.

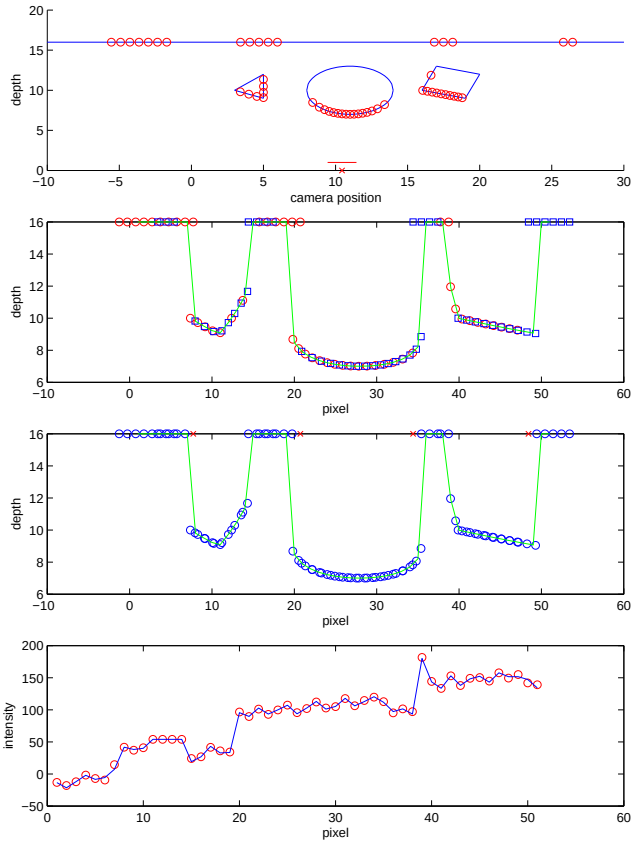

Fig. 2. Propagation Algorithm. From top to bottom: a synthesis scene seen from a camera 10.45; propagated points from cameras 9 (circle) and 12 (square); removed points (x-mark) and points remaining to interpolate (circle); and the rendered image (circle) plotted on the actual one (line).

In the experiment, we choose a scene consisting of a background and objects with various depth. The object surfaces are painted intensity functions of form $\alpha\text{round}(x) + \beta\sin(\gamma x) + \eta$, where $\alpha, \beta, \gamma, \eta$ vary from surface to surface. Fig. 2 shows an illustration of our algorithm and the rendering result. We can see that the algorithm produces a very good result in texture regions, and even at boundaries the algorithm still performs very well.

4. PROPAGATION ALGORITHM WITH LESS DEPTH AVAILABLE

In this section we present the algorithm in the case where $\mathcal{S}_\mathcal{D}$ is strictly a subset of $\mathcal{S}_\mathcal{I}$. The main idea is first to recover the depth information for all the actual image pixels, then to apply the previous algorithm to render the virtual image.

To recover the depth of pixels of an actual camera u, we make use of the correspondence between the intensity of actual images. We first propagate depth-available pixels from other actual cameras to camera u. Then at each propagated point, by calculating the *local intensity consistency* [12] we can detect if there is an occlusion. If there is no occlusion, we obtain the depth of this point. By doing so, we will get the depth at a set of points on the image plane of camera u. We then can do the interpolation to get the depth for other pixels. Fig. 3 shows the recovered depth and rendered image for the same configuration as in Fig. 2. Rigorously, for each actual camera u, the algorithm is as following:

```
% Initialize the set of depth-available pixels
D_u = {v : (u, v) ∈ S_D};

% Propagate the depth
For each (u_0, v_0) ∈ S_D
  v = V_u(u_0, v_0) = v_0 − (u − u_0) f / D(u_0, v_0);

  % Test the local intensity consistency
  If I(u, v) ≈ I(u_0, v_0)
    D(u, v) = D(u_0, v_0);
    D_u = D_u + {v};
  Endif
Endfor

% Interpolate the depth
D_u = Interpolate(D_u, D(u, v)|_{v∈D_u})
For each v : (u, v) ∈ S_I
  D(u, v) = D_u(v);
Endfor
```

(6)

5. CONCLUSION AND FUTURE WORK

In this paper we present a new approach for the IBR problem. By assuming the depth information at some or all image pixels, the algorithm produces an excellent improvement of rendering quality, especially at the object boundaries, while maintaining low computational complexity. Furthermore, the proposed algorithm is flexible with the scene configuration, which enables implementation in real-time systems. While the assumption about the depth availability seems likely in the range camera technology of the future, our algorithm still works with current technologies like feature correspondence or markers. Furthermore, the approach provides a new framework for a rigorous investigation of the rendering quality by turning the IBR problem into a classical 2D nonuniform interpolation problem. The future work will be in this direction.

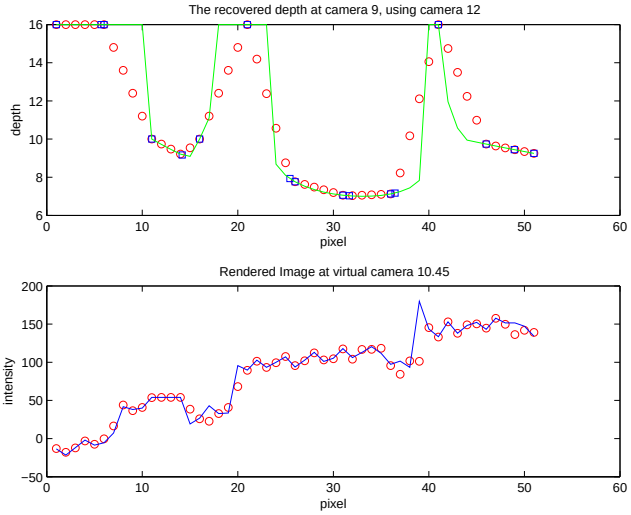

Fig. 3. Depth information available every $k = 5$ intensity pixels and uniformly distributed. The configuration is the same as in figure 2. On top: the considered scene and the recovered depth at camera 9 (circle) using depth depth available pixels of camera 9 and propagated depth available pixels of camera 12 (square); bottom: the rendered image (circle) at 10.45 using two cameras 9 and 12 compared with the actual image.

6. REFERENCES

[1] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, M. Landy and J. A. Movshon, Eds. MIT Press, 1991, pp. 3–20.

[2] A. Aldroubi and K. Gröchenig, "Nonuniform sampling and reconstruction in shift-invariant spaces," *SIAM Review*, vol. 43, no. 4, pp. 585–620, 2001.

[3] C.-F. Chang, G. Bishop, and A. Lastra, "LDI tree: A hierarchical representation for image-based rendering," in *Siggraph 1999, Computer Graphics Proceedings*, Los Angeles, 1999, pp. 291–298.

[4] P. E. Debevec, G. Borshukov, and Y. Yu, "Efficient view-dependent image-based rendering with projective texture-mapping," in *In 9th Eurographics Rendering Workshop, Vienna, Austria*, 1998.

[5] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice-Hall, 2002.

[6] Z.-F. Gan, S.-C. Chan, K.-T. Ng, K.-L. Chan, and H.-Y. Shum, "On the rendering and post-processing of simplified dynamic light fields with depth," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, 2004.

[7] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," in *Proc. SIGGRAPH*, 1996, pp. 43–54.

[8] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. SIGGRAPH*, 1996, pp. 31–40.

[9] M. Rioux, "Digital 3-d imaging: Theory and applications," in *Proc. SPIE Conf. on Vis. Commun. and Image Proc.*, ser. Videometrics III, 1994.

[10] J. Shade, S. Gortler, L. He, and R. Szeliski, "Layered depth images," in *Proc. ACM SIGGRAPH*, Orlando, Florida, July 1998.

[11] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Trans. Circ. and Syst. for Video Tech.*, vol. 13, pp. 1020–1037, Nov. 2003.

[12] C. Zhang and T. Chen, "A self-reconfigurable camera array," in *Eurographics Symposium on Rendering.*, 2004.

[13] ——, "A survey on image-based rendering - representation, sampling and compression," *EURASIP Signal Processing: Image Communication*, pp. 1–28, Jan. 2004.