MVA2005 IAPR Conference on Machine VIsion Applications, May 16-18, 2005 Tsukuba Science City, Japan
13-8

Image-Base Rendering using Plane-Sweeping Modelisation

Vincent Nozick Sylvain Michelin Didier Arquès

Marne la Vallée University, Gaspard Monge Institut, SISAR team
6 cours du Danube, 77700 Serris, France
{vnozick,michelin,arques}@univ-mlv.fr

Abstract

This paper presents a method to create new images of a scene from a small set of calibrated images using color matching. While most of the existing techniques attempt to build a 3d reconstruction of the scene, the only geometric constraint our color matching method requires is camera calibration. This paper describes a new adaptation of the plane-sweeping algorithm using a local dynamic scoring method that handle occlusions. We also propose a real-time implementation of our method using fragment programs. These modifications are well suited for on-line real-time rendering of dynamic scenes.

1 Introduction

Realistic rendering can be achieved using two approaches : computer graphics where an image is generated from a scene description and computer vision where the image is generated from pictures of real objects. The second approach often uses geometric reconstruction like texture map rendering [9,7] or volumetric techniques [8], which implies geometric interpretation of the scene. These techniques become more and more accurate but they require a large set of input images and a lengthy computation time before visualisation. They also need complex points and surfaces reconstruction algorithms, accurate feature extraction and a robust calibration method. However these techniques allow a fluid navigation in the reconstructed scene. The main drawback of these methods is the cost of the computation which rules out their use for animation in the case of a dynamic scene.

Implicit geometry methods like [5,2,4] generate new views without any geometric interpretation of the scene and achieve real-time rendering using interpolation between views but they need a large set of input images. [6] propose a real-time reconstruction that shades visual hulls from silouhette image data but therefore can not handle concave objects.

Finally, view-dependent geometry methods provide realistic rendering without geometric reconstruction nor correspondence estimation between different images. Some methods such as [3,1,10] generate new views without any other geometric information than the calibration matrices of the input images. These methods are based on color matching betwen differents views. They do not explicitly compute 3d geometry but this implicitly appears in the resulting images. Our algorithm belongs to this latter familly and follows a plane-sweeping algorithm. We will first present the basic plane-sweeping algorithm and some related work. Then, we will introduce our new scoring method and explain our implementation.

2 Basic plane-Sweeping algorithm and related work

Given a small set of calibrated images from video cameras, we wish to generate a new view of the scene from a new viewpoint. Considering a scene where objects are exclusively diffuse, we first place the virtual camera and divide space in parallel planes D_i in front of the camera as shown in Figure 1. We project the input images onto each plane D_i in a back to front order. Let's consider a visible object of the scene lying on one of these planes at a point M. The input cameras will project on M the same color (i.e. the object color). Therefore, points on the planes D_i where projected colors match together potentially correspond to an object of the scene.

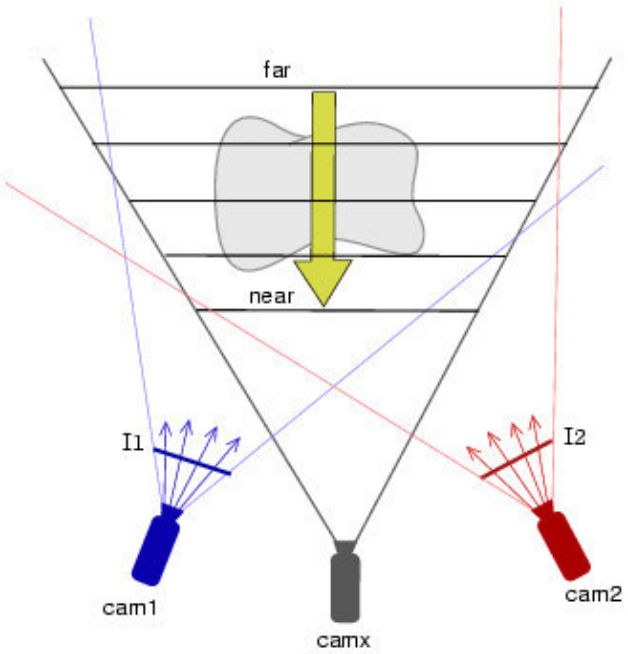

Figure 1. Plane-sweeping algorithm : geometric configuration.

Let $I_1...I_n$ denote a set of n images and $P_1...P_n$ their respective 3×4 projection matrices. I_x is the new image to

be computed, cam_x is its virtual pinhole projective camera and P_x its associated projection matrix. We define a *near* plane and a *far* plane parallel to cam_x image plane such that all the objects of the scenes lie between *near* and *far*. For each pixel of each plane D_i, a score and a color are computed according to the matching of the colors. The plane-sweeping algorithm can be explained as follows :

- initialize I_x's score
- **for** each plane D_i from *far* to *near*
  - → project all the input images $I_1...I_n$ on D_i as textures
  - → project D_i multi-textured on I_x
  - → **for** each pixel p of I_x
    - compute a score and a color according to the coherence of the colors from each camera's contribution
    - **if** the score is better than the previous ones **then** update the score and the color of p
- draw I_x

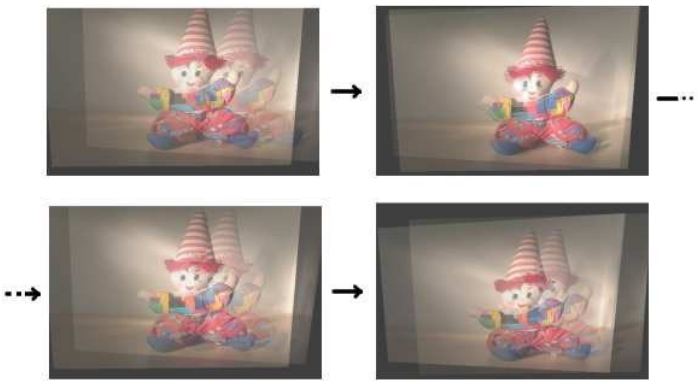

Figure 2. Four pictures associated to four steps of the algorithm using two input cameras.

Yang et al. [10] propose an implementation of the plane-sweeping algorithm using pixel shaders and reach real-time rendering using five input cameras. For the scoring stage, they choose a reference camera that is closest to cam_x and pairwise compare the contribution of each input image with the reference image. According to the small number of instructions, this method provides good speed results, however the occlusions are not handled. This method also implies geometric constraints on the position of the cameras : the input cameras have to be close to each other and the navigation of the virtual camera should lie between the viewpoints of the input cameras, otherwise the reference camera may not be representative of cam_x. Lastly, there may appear discontinuities in the computed images when the virtual camera moves and changes its reference camera.

3 New color scores

The score computation is a crucial step in the plane-sweeping algorithm. Both visual results and speedy computation depend on it. We propose a new method to compute a score according to all the input image colors instead of computing by pairs. For this purpose, we use multi-texturing fonctions to access to each input camera color contribution.

For each pixel of I_x, we propose a finite number of positions X in the scene (one per plane D). A score is computed for each position and this score depends on the color C_i of the projections of X in each input image. We propose an iterative algorithm to reject outliers colors and compute an optimal score.

This method first computes the variance v of the color set $S=\{C_i\}_{i=1...n}$ and finds the color $C_f \in S$ the furthest from the average. If this color is further than a defined distance d, then it is removed from S. This step is repeated until stability or until S contains only 2 elements. The returned score is the variance found in the last step. This algorithm can be summarized as follows :

- **bool** *stable* = **false**
- $S = \{C_i\}_{i=1...n}$
- a = average(S)
- v = variance(S,a)
- **do**
  - → find the farest color $C_f \in S$ from a
  - → **if** $|\text{dist}(C_f,a)| > d$ **then**
    - $S = S - \{C_f\}$
    - a = average(S)
    - v = variance(S,a)

    else *stable* = **true**

  while Card(S) ≥ 2 **and** *stable* = **false**
- *score* = v
- *color* = a

If an object is viewed by most of the input cameras then it will be correctly displayed since this method rejects outliers colors, hence this method handles occlusions. Again, this method does not need a reference camera i.e. a camera near to the virtual camera and with similar properties. Thus we can place the virtual camera anywhere in front of the scene and it does not need to lie between the input cameras. Finally, our method avoids discontinuities which can happen when the virtual camera is moving and changes its reference camera.

4 Implementation

Our plane-sweeping modelisation method has been implemented using OpenGL 1.5 and OpenGL Shading Language for the scoring stage. The texture coordinates are computed using projected textures from the camera matrices P_i. We use multitexturing in order to get access to each texture during the scoring stage. The scores are stored in the depth buffer and the colors in the color buffer. Hence we let OpenGL select best scores with the z-test and update the color in the frame buffer. Most of the work is done by the graphic card and the CPU is free for others tasks.

Figure 3. Three input calibrated images.

Figure 4. Two real-time images from three input cameras (20 fps with 30 planes).

5 Results

Figure 3 shows three calibrated images taken from three video cameras. Two pictures generated in real-time from this set of input images are shown in Figure 4. Despite the apparition of noise, we can see that the resulting image is close to the real model. The computation of a 325×260 final image using 30 depth samples along cam_x's principal ray provide a 15 fps framerate on an Pentium4 2.8 GHz and an ATI Radeon 9800 Pro. This application is dedicated to virtual reality using stereo display. In that case, the humain brain removes noise errors during the fusion of the two images.

6 Conclusion

We have presented in this article an adaptation of an image-based method using a small set of input images whereas other methods require a large set. The final image is quite realistic compare to the real model. Moreover, since the computation of an image does not depend on the previous generated images, our method allows the rendering of dynamic scenes for a low resolution final image. This method can be implemented for every consumer graphic harware that supports fragment shaders and therefore frees the CPU for other tasks. Since our scoring method takes into account all input images together and reject outliers, our method handle occlusions. Finally, our scoring method enhances robustness and implies fewer constraints on the position of the virtual camera, i.e. it does not need to lie between the input camera's area. We propose to extend our research on a stereoscopic adaptation of our method that compute the second view for low cost.

References

[1] Andrew Fitzigibbon, Yonatan Wexler and Andrew Zisserman. "Image-based rendering using image-based priors", *9th IEEE International Conference on Computer Vision*, pages 1176-1183, 2003.

[2] J. Gortlerand, R. Grzeszczuk, R. Szeliski and M. F. Cohen. "The lumigraph", *SIGGRAPH*, pages 43-54, 1996.

[3] M. Irani, T. Hassner and P. Anandan. "What does the scene look like from a scene point?", *Proc. ECCV*, pages 883-897, 2002.

[4] Marc Levoy and Pat Hanrahan. "Light Field Rendering",

SIGGRAPH, pages 31-42, 1996.

[5] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System , *SIGGRAPH*, pages 39-46, 1995.

[6] Wojciech Matusik, Hanspeter Pfister, Addy Ngan Paul Beardsley, Remo Ziegler and Leonard McMillan. Image-Based Visual Hulls , *Proc ACM SIGGRAPH*, pages 369-374, 2000.

[7] Marc Pollefeys, Maarten Vergauwen and Luc Van Gool. Automatic 3D modeling from image sequences , *International Archive of Photogrammetry and Remote Sensing*, Amsterdam, pages 619-626, 2000.

[8] Steven M. Seitz and Charles R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring , *Proc. CVPR*, pages 1067-1073, 1997.

[9] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms , *IJCV*, 47, pages 7-42, 2002.

[10] Ruigang Yang and Greg Welch and Gary Bishop. Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware , *Proc. of Pacific Graphics*, pages 225-234, 2002.