Real Time Novel View Scene Rendering From Multiple Sparse Videos

Jeferson Rodrigues da Silva, Thiago Teixeira Santos, Carlos Hitoshi Morimoto
Computer Science Department
Institute of Mathematics and Statistics - University of Sao Paulo
Sao Paulo, Brazil

***Abstract*—Free Viewpoint Video (FVV) might become one of the most interesting features in future digital television, enabling viewers to move around freely and observe the scene in three dimensions (3D) from any desired viewpoint, similar to many 3D interactive computer graphics applications today. This paper describes a FVV appropriate for virtual reality systems that allows the user to explore a 3D scene in real time using a small set of sparse cameras. Our method combines a simple geometric model of the scene composed of planes with tracking information of moving targets to generate novel perspective corrected 3D views of a virtual camera. To achieve real-time rendering frame rates, view dependent textured mapped billboards are used to render the moving objects at their correct locations and foreground masks are used to remove the moving objects from the projected video streams. Our current real-time prototype is able to generate an interactive 2D virtual camera view from 3 synchronized video streams of resolution 768×576 with several moving objects at about 12 fps.**

***Keywords*-Free viewpoint video; real time image and video synthesis using sparse cameras.**

I. Introduction

Future digital television (DTV) features might include 3D-DTV (3 dimensional DTV) and FVV (Free Viewpoint Video), which enables viewers to move around the scene, similar to many 3D interactive computer graphics applications today. This feature allows the user to visualize events in 3D with a lot more flexibility and control. FVV is a particular interesting problem because its solution must combine elements from computer graphics, computer vision, image processing, and human computer interaction. The solution is commonly achieved using Image Based Rendering (IBR) techniques [1]. A recent special issue of the Signal Processing Image Communication Journal [2] focused on these and other issues about the future of 3D video and TV applications such as representation, coding, transmission, and so on.

IBR can be described as a process of sampling and rendering of the plenoptic function [3]. The plenoptic function l is a 7D function that models the appearance of a 3D dynamic environment and requires no geometric model. The function represents the light rays at every space location (x, y, z), towards every direction (α, β), over all wavelenghts (λ) and at any time (t), so that it can be written as:

$$l(x, y, z, \alpha, \beta, \lambda, t). \quad (1)$$

Rendering is possible after appropriate samples are taken from l. Samples can also be used for representation and storage. For practical reasons, the 7D plenoptic function l must be simplified. For example, wavelength may be constrained to visible and quantized to RGB values, the motion of the viewer may be restricted to a surface, or the position of the viewer may be fixed, what reduces the dimension of the plenoptic function. Time is also commonly dropped for rendering static scenes.

Very realistic rendering is achieved by using a large amount of samples or images. For example, 120 computer controlled still cameras were used to capture the scene in the movie *Matrix*, where Keanu Reeves dodges bullets, a technology named "Flo-Mo" by the movie producers. In order to reduce the number of cameras needed, it is possible to increase the amount of knowledge about the scene, such as scene geometry, as suggested by Debevec *et al.* [4].

The focus of this paper is the efficient rendering of novel views from multiple sparse video streams to allow the use of previously recorded real scenes in virtual reality systems. We are particularly interested in generating novel views of scenes recorded with multiple sparse cameras with significant overlap, but far less than the minimum required to apply "Flo-Mo".

The two main contributions of this paper are the development of a real time system for free viewpoint video rendering from multiple sparse video streams, and the implementation of an efficient algorithm to compensate camera illumination and color differences.

A. Related Work

A survey of IBR techniques is presented in [1] by Zhang and Chen. For purely IBR techniques, data acquisition (sampling) is generally non-trivial due to calibration and synchronization of a large number of cameras required (for example, for techniques such as Light Field and Lumigraph). Real-time performance for VBR remains a challenge for reasons such as video capture and synchronization, data storage, transmission(for live applications), and rendering.

For the purpose of this work, we select the rendering of large scenes and people activity as the most important features for FVV. Kang *et al.* [5] introduced plenoptic primitives to enable virtual walkthroughs in large environments. A plenoptic primitive can be any type of local visual

experience, such as 360^o static panoramas, panoramic video, or concentric mosaics. By combining these primitives, the user visual experience is enhanced.

Real time image-based rendering systems are reported in [6]–[8]. They are all similar in the way that they all compute depth or structure as a pre-processing step. Once the structure information is available, novel views are generated in real time. Such techniques do not deal with video streams and their associated problems, such as moving objects.

Starck and Hilton [9] and Carranza *et al.* [10] address the synthesis of novel views of people from multiple video streams. Starck and Hilton [9] consider the target area of a multiple camera capturing system (a 3D virtual studio) with the requirement for free-viewpoint video synthesis for a virtual camera with the same quality as the captured video. View-dependent shape optimization combines multiple view stereo and silhouette constraints to estimate correspondences between images in the presence of visual ambiguities such as uniform surface regions, self-occlusion, and camera calibration error. Carranza *et al.* [10] extract silhouettes from the synchronized video frames using background subtraction that are used to compute the pose of a 3D human body model.

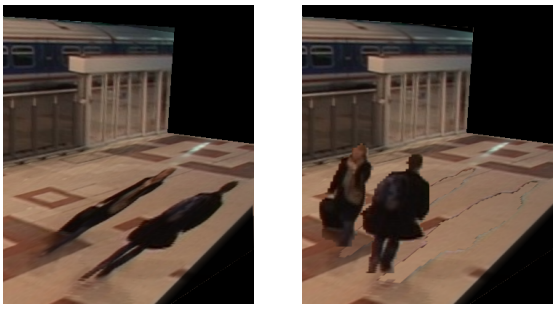
Min *et al.* [11] presents a method to synthesize novel virtual views in multi camera configurations for 3D TV systems. Similar to the IBR technique described by Willems *et al.* [6], they first compute disparity maps of neighboring views. Occlusion is handled by using cost function to appropriately weight the views.

For our purposes the methods described so far would not produce good results due to the lack of camera control, low image quality, low number of cameras and large distance between cameras. To deal with sparse uncalibrated cameras, we build a simplified scene model, that includes scene geometry and texture mapping, similar to Debevec *et al.* [4] who introduced an algorithm that combines IBR with an approximate 3D scene model to efficiently render a realistic fly-by video of the scene. Our algorithm, described in Section II, extends this technique to handle video streams and moving objects. The simple model creation is also described and allows the system to handle occlusion artifacts. Another large source of artifacts is due to the color differences between images of different non color calibrated cameras. We present a color calibration method that compensates for color differences between cameras and between each camera and the scene model. Experimental results are presented in Section III, and Section IV concludes the paper.

II. Novel view scene rendering

Our method for novel view scene rendering is composed of a pre-processing stage and an interactive real-time rendering stage as seen in Figure 1.

The pre-processing stage receives as input a set of synchronized videos of the scene, the calibration data for each camera, and a textured 3D scene model. At first, all videos and the calibration data are used as input for a segmentation and tracking module. This module computes foreground masks and tracking information for all frames of every video. Then the videos and the scene model are used as input for a color calibration module that calculates and stores a different look-up table for each video. Such tables are applied to each video to minimize the color differences between each video and the scene model.



(a) Objects as projections (b) Objects as billboards

Figure 2. Perspective error when the viewpoint of the virtual camera differs from the viewpoint of the projected video

The rendering stage uses the same input as the pre-processing step in addition to the outputs from the segmentation and tracking module and from the color calibration module to generate novel views of the scene in real time. Basically the texture mapped 3D scene model is first rendered from the viewpoint of the virtual camera controlled by the user. Then the video with viewpoint closest to the virtual camera viewpoint is projected onto the scene.

By projecting the video onto the scene model we are able to show the moving objects over the static scene model although these moving objects appear as if they were painted over the scene model textures. This happens when the chosen viewpoint differs from the viewpoints of the set of videos as in Figure 2a. In order to render these moving objects correctly for novel views, we use the data obtained from the segmentation and tracking module to render view-dependent texture mapped billboards representing each object. Using the foreground mask information, the pixels corresponding to the moving objects are marked as transparent, therefore hiding these objects from the video projection. The holes produced by this approach are automatically filled by the scene model that is rendered before the video projection. Texture swapping on virtual camera movement for video projection and billboards is done smoothly because of view-dependent texture mapping. Figure 2b shows an example of the final rendering result.

In the following subsections we detail the pre-processing and rendering stages.

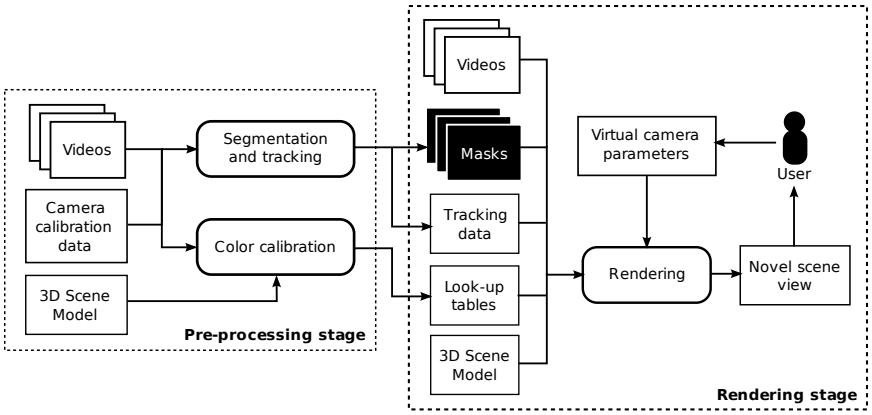


Figure 1. Method overview

A. Pre-processing stage

1) Model construction: A manually built 3D scene model is used as input for the color calibration module and is also used during the rendering stage. The simplified scene model is composed of a set of planes with textures extracted from the real scene. Those textures can be obtained from photos of the scene by correcting radial and perspective distortions. Figure 3 shows examples of scene models used in our experiments, where the planar decomposition is noticeable.

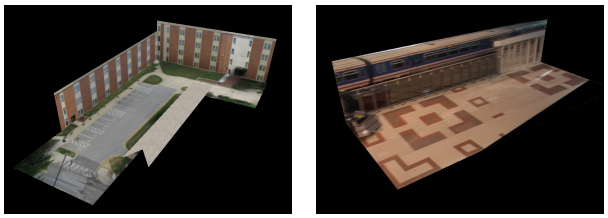


Figure 3. Renderings of the parking lot and PETS'06 scene models.

2) Camera calibration: Assuming that the cameras in all videos are static, they can be calibrated by the correspondence of a set of points whose world and image coordinates are known. In our system, the Tsai's camera calibration method [12] was used to estimate the internal and external camera parameters from a set of manually extracted 3D scene points and its corresponding 2D image points in each camera view. Since this calibration method depends on the accuracy of the coordinates of the given points, the position and viewpoint of each camera was further refined manually to provide a better alignment of the video projections to the 3D model.

3) Segmentation and Tracking: Moving objects are segmented from the static scene background using a mixture of Gaussians background subtraction algorithm [13]. Assuming that the objects move along a flat surface, the location of each moving object on the ground plane is computed using a homography constraint over multiple cameras similar to [14]–[16]. The major advantage of this method is that it is robust to severe occlusion that occurs in crowded scenes.

After the blobs of the moving objects are segmented in each camera, the location of each object is computed as follows. Let $\mathbf{H}_i$ be the homography that maps the ground plane to the camera view c_i and $\mathbf{v}_i$ be the ideal point in the vertical direction for c_i. Though there are many ways of automatic computing $\mathbf{H}_i$ and $\mathbf{v}_i$ [17], we use manually selected image features to compute them.

Let $\mathbf{x}_i$ be a foreground pixel of c_i. If $\mathbf{x}_i$ belongs to the ground plane, it can be mapped back to the plane using $\mathbf{H}_i^{-1}$. Because $\mathbf{v}$ is known, the number of the foreground pixels above $\mathbf{x}_i$ can also be computed. We call the number of foreground pixels above $\mathbf{x}_i$ the support of $\mathbf{x}_i$, or $s(\mathbf{x}_i)$. The basic idea is that foreground pixels with maximum support correspond to points on the ground plane.

The ground plane Π is represented as a regular grid of finite dimensions. For each camera c_i the support $s(\mathbf{x}_i)$ of all foreground pixels is mapped to its corresponding grid ground point using $\mathbf{H}_i^{-1}$. Figure 4 illustrates this process. Assuming for simplicity that the moving object is a vertical line segment, the line l_i seen from each camera c_i corresponds to a line segment l_i' on the ground plane. Because the line is perpendicular to the plane, each camera c_i and its corresponding homography $\mathbf{H}_i$ projects a different line on the ground plane. From projective geometry it is well known that the set of l_i' will meet at the location of the line on the ground plane.

The homography constraint works well when just a few objects are present but not for crowded scenes due to the large number of line crossings. Including $s(\mathbf{x}_i)$ as weighting coefficients, regions on the ground plane that correspond to actual locations of targets are easily segmented as local maxima, as shown in Figure 5.

The candidate objects segmented using the support information and the homography constraint are tracked using

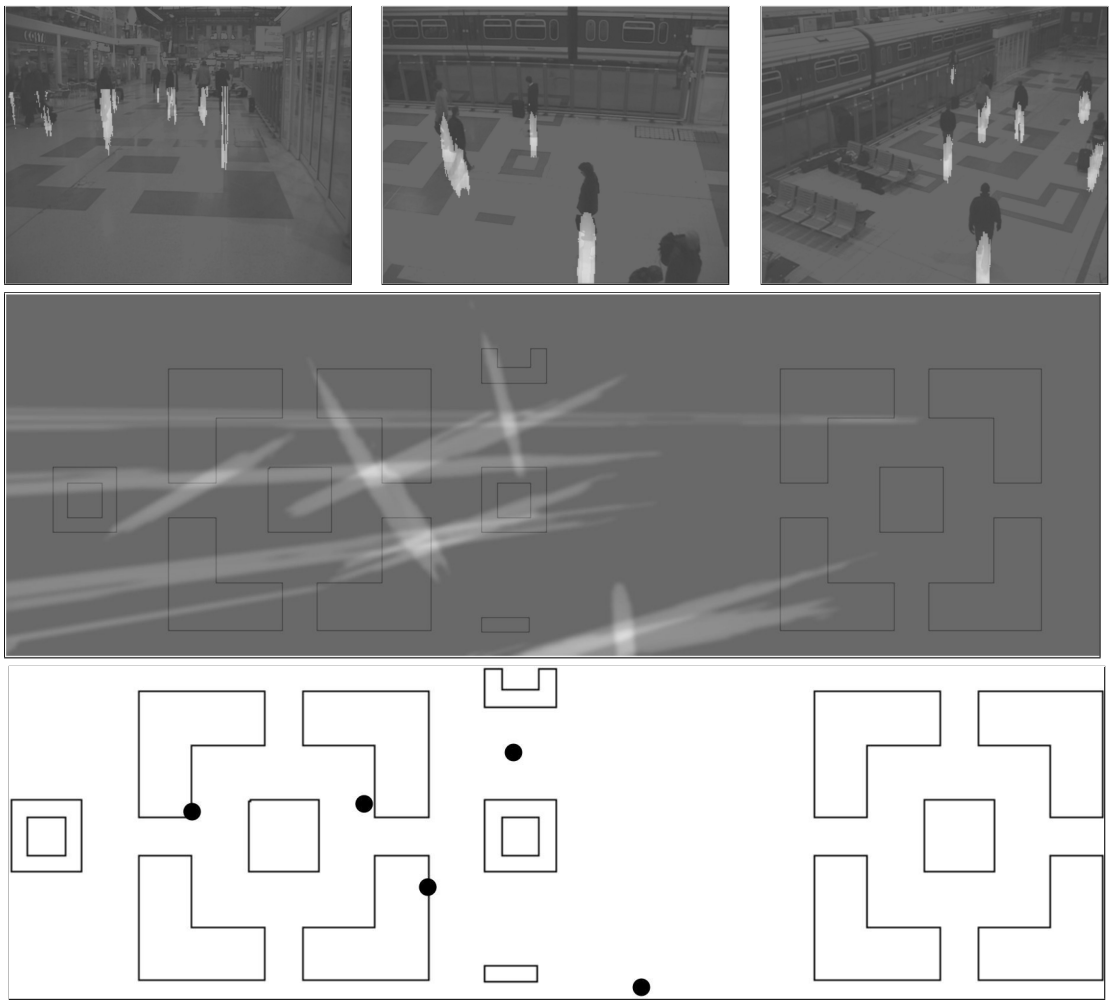


Figure 5. The top row shows 3 synchronized views of a crowded scene. The support of each foreground pixel is overlaid over the 3 views. The middle row shows the projection of the support information on the ground plane. The local maxima near line crossings are used to determine the location of objects. These locations are displayed in the bottom row. The square pattern of the ground plane is reproduced in the middle and bottom rows.

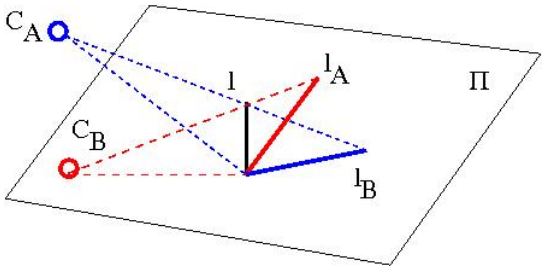


Figure 4. Line l is perpendicular to the ground plane Π. The homographies of Π to the image plane of the cameras C_A and C_B project l onto the lines l_A and l_B respectively.

Kalman filters. A simple constant velocity motion model is assumed for each object.

4) Color calibration: Due to differences in camera properties and settings such as sensors, lenses, exposure, white balance etc., images from the same scene taken by different cameras may present different colors and illumination. The function of the color calibration module is to calculate look-up tables that are to be applied to each video frame so that the difference between the colors of the videos and the scene model is minimized. This is required to reduce color artifacts near transitions between camera views and the model, seamlessly blending the videos with the scene textures.

General histogram-based color calibration methods [18], [19] consist on comparing the color histograms of image regions assumed to have similar color distributions and applying color transformation operations on the images in order to minimize the differences in color distribution. Our color calibration module uses as input the set of videos of the scene in addition to a set of images of the rendered 3D scene model matching the viewpoint of each video. We select the camera with more natural color and brightness to use as reference. Then, the RGB color histograms of corresponding

pairs of video and reference images are compared and look-up tables that minimize the difference between their color histograms are generated. A different look-up table is calculated for each color channel mapping all 8-bit color values in a channel to their corresponding calibrated 8-bit color values.

We have developed a simulated annealing color calibration algorithm that finds monotonically increasing polynomials used to generate the desired look-up tables. Such polynomials are desirable to keep the basic shape of the histograms by maintaining the relative order between color intensities, therefore not changing the overall image color. For cameras that present large color difference, one possible solution is to minimize the difference between the color histograms using only overlapping regions of the cameras.

Let $I = (S, C)$ be an 8-bit grayscale image, where S is a set of points representing the shape of the image I and C a function that assigns a color intensity for each point in S. Formally, S and C are defined by:

$$S = \{(x, y) \in \mathbb{N}^2 : 0 \leq x < width, 0 \leq y < height\}$$

$$C : \mathbb{N} \times \mathbb{N} \rightarrow [0, 255]$$

The color v of an arbitrary $p = (x, y) \in S$ is given by $C(p) = v$.

A grayscale image histogram is a discrete function that represents the distribution of all possible color intensities on the image. Formally, the histogram h of image $I = (S, C)$ is given by:

$$h_I(k) = |\{p \in S : C(p) = k\}|, \text{ for } k \in \mathbb{N} \text{ and } 0 \leq k \leq 255$$

where $|A|$ indicates the cardinality of A.

We define the normalized histogram of an image by

$$H_I(k) = \frac{h_I(k)}{|S|}$$

We define the difference between two histograms h_1 and h_2 by:

$$D(h_1, h_2) = \sum_{0 \leq k \leq 255} (h_1(k) - h_2(k))^2$$

Let $I_a = (S_a, C_a)$ and $I_b = (S_b, C_b)$ be two distinct images from a same scene that share an overlapping region. Let R_a and R_b be the overlapping regions that represent a common plane between images I_a e I_b and let G be the homography that takes a point $p_a \in R_a$ to the corresponding point on $p_b \in R_b$. We define the overlapping regions of I_a and I_b by the sub-images $I_{ab} = (S_{ab}, C_a)$ and $I_{ba} = (S_{ba}, C_b)$ where

$$S_{ab} = \{p_a \in R_a : Gp_a \in S_b\}$$

$$S_{ba} = \{p_b \in R_b : G^{-1}p_b \in S_a\}$$

A look-up table can be defined as a function that assigns a new intensity value for each gray level of a grayscale image. Formally, a look-up table is defined by $L : [0, 255] \rightarrow [0, 255]$. Let $l(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$ be a polynomial of degree n then we can obtain a look-up table from l by calculating $L(x) = l(x)$ with $x \in \mathbb{N}$ and $0 \leq x \leq 255$.

By applying a look-up table L on an image $I = (S, C)$ we obtain a new image $L(I) = L(S, C) = (S, L(C))$.

Now we can define the color calibration problem as an optimization problem where we need to find the polynomial l that minimizes $D(H_{L(I_{ab})}, H_{I_{ba}})$ or $D(H_{L(I_{ba})}, H_{I_{ab}})$, where L is the look-up table obtained from l.

To solve this problem, we used an algorithm based on the probabilistic method of simulated annealing [20]. To apply the simulated annealing method, we need to define the problem in terms of states, energy and temperature.

We start by defining a state s by a set of $n + 1$ points on a $g \times g$ grid defined over the $[0, 255] \times [0, 255]$ space that will represent a polynomial of degree n that contains all the $n + 1$ points in s. Formally, we define a state s by $s = (p_0, p_1, p_2, \ldots, p_n)$ with $p_k = (x_k, y_k)$. The x_k and y_k values must be restricted to the $g \times g$ grid and for that reason $x_k = gu$, $y_k = gv$ with $g, u, v \in \mathbb{N}$, $0 \leq x_k \leq 255$ and $0 \leq y_k \leq 255$.

We also define some additional positioning restrictions for the state points so as to avoid the generation of states representing non monotonically increasing polynomials. Given two points $p_i = (x_i, y_i)$ and $p_{i+1} = (x_{i+1}, y_{i+1})$ with $0 \leq i < n$ and $i \in \mathbb{N}$, they must satisfy the restrictions: $x_{i+1} > x_i$ and $y_{i+1} > y_i$.

Let $s = (p_0, p_1, p_2, \ldots, p_n)$ and $t = (q_0, q_1, q_2, \ldots, q_n)$ be two arbitrary states. We say s and t are neighbours if $\exists i \,|\, p_i \neq q_i, p_k = q_k \,\forall k \neq i$ and $|x_{p_i} - x_{q_i}| + |y_{p_i} - y_{q_i}| = g$. Figure 6 illustrates an arbitrary state and all of its neighbours.

The energy E_s of the state $s = (p_0, p_1, \ldots, p_n)$ is obtained by calculating $D(H_{L_s(I_{ab})}, H_{I_{ba}})$, where L_s is the look-up table obtained from the state s. To build the look-up table L_s, we need to obtain the polynomial $l(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0$ represented by s. This polynomial can be obtained by solving the following system:

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Finally, the temperature T is defined by an initial temperature T_0, chosen accordingly to the problem, and it is updated each K iterations of the algorithm by calculating $T_{i+1} = \alpha T_i$ with $0 \leq \alpha < 1$.

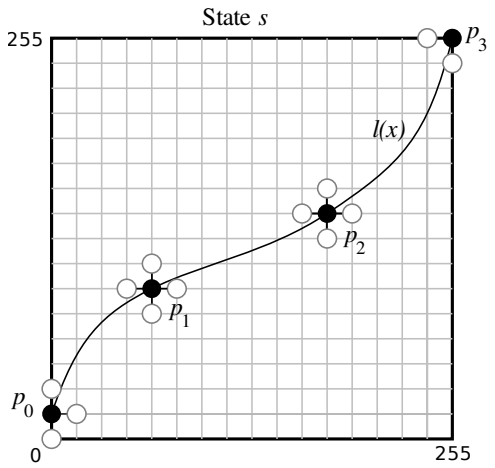


Figure 6. State and neighbours for the simulated annealing algorithm. The state $s = (p_0, p_1, p_2, p_3)$ represents a polynomial of degree 3 indicated on the figure by $l(x)$. Each neighbouring state of s is obtained by moving one point p_k to any of its neighbours indicated by the white filled circles on the grid.

On Algorithm 1 we show the algorithm to calculate the color calibration look-up table from image I_a to image I_b.

B. Rendering stage

The rendering step is responsible for the generation of an interactive real-time visualization of novel views of the scene.

The rendering process uses all previously calculated data from the pre-processing step and some additional parameters that can be controlled by the user: the camera position, the viewing direction, and the video playing mode (such as play, pause, rewind, and so on). These parameters are controlled by the user in real-time via the user interface.

The process starts by using the stored look-up tables to adjust the colors of each video. This operation is very efficient and is done as the current frame of each video is loaded by changing each pixel RGB value for its corresponding look-up table value.

The next 2 textures for each video frame are generated: a foreground and a background texture. The foreground texture is built using the calibrated RGB color channels of the video frame and the foreground mask computed from the Segmentation and Tracking Module is used as its alpha channel. The background texture is also built using the calibrated RGB color channels but uses the inverted foreground mask as its alpha channel. The invertion is required to remove the incorrect projection of the moving objects from the background textures, i.e., making them transparent in the background texture. Let L_i be the look-up table, V_i the RGBA image and α_i the foreground mask associated with camera i with $1 \leq i \leq n$ where n is the total number of cameras. Then the foreground texture F_i and the background texture B_i associated with camera i are defined as

$$F_i = \alpha_i L_i(V_i) \quad (2)$$

$$B_i = (1 - \alpha_i) L_i(V_i) \quad (3)$$

After producing all textures for the current frame, the 3D scene model is rendered from the viewpoint chosen by the user. We then determine which video has the viewing direction closest to the chosen virtual camera by choosing the camera whose angle formed by its viewing direction and the virtual viewing direction is the smallest. Then the video corresponding to the chosen camera is projected over the scene model using the associated background texture. Note that since the scene model is rendered prior to the video projection and since the color difference between the video and the model is minimized, the holes present on the background texture due to the moving objects removal are minimized.

In the final step, the moving foreground objects are rendered as view-dependent texture mapped billboards over the model. For each object present on the tracking data, a

Algorithm 1 Algorithm to calculate color calibration look-up table from image I_a to image I_b

Require: $s_0, T_0, \epsilon, K, H_{I_{ab}}, H_{I_{ba}}$
Ensure: L

```
1.  i ← 0
2.  s ← s_0
3.  T ← T_0
4.  while T > ε do
5.      if i = K then
6.          T ← UPDATETEMPERATURE(T)
7.          i ← 0
8.      end if
9.      s_n ← RANDOMNEIGHBOUR(s)
10.     L_s ← CALCLOOKUPTABLE(s)
11.     L_{s_n} ← CALCLOOKUPTABLE(s_n)
12.     E_s ← D(H_{L_s(I_ab)}, H_{I_ba})
13.     E_{s_n} ← D(H_{L_{s_n}(I_ab)}, H_{I_ba})
14.     δE ← E_{s_n} − E_s
15.     if δE < 0 then
16.         p ← 1.0
17.     else
18.         p ← e^{−δE/T}
19.     end if
20.     q ← GETPROBABILITY()
21.     if q < p then
22.         s ← s_n
23.     end if
24.     i ← i + 1
25. end while
26. L ← CALCLOOKUPTABLE(s)
27. return L
```

billboard is rendered over the scene using the previously obtained position information.

In order to generate the correct texture for each billboard it is necessary to determine which cameras have the best viewing angles relative to the billboard position. Viewing directions are represented by a position on a sphere centered at the billboard position. The sphere is decomposed in triangles as seen in Figure 7. The set of vectors from the sphere center to each triangle vertex represents the discrete set of viewing directions that are considered for the billboard. The number of relevant viewing directions can be reduced by considering just the vertices of the triangle intersected by the line segment defined by the virtual camera position and the billboard position. Each vertex v_i is associated to camera c_j as defined by

$$c_{v_i} = \max_j < c_j, v_i >, \forall j \in \{1, \ldots, n\}, i \in \{1, 2, 3\} \quad (4)$$

where c_{v_i} is the camera associated with v_i, that corresponds to the camera c_j with viewing angle that maximizes the dot product with v_i.

Finally, we find the weight of each camera texture calculating the barycentric coordinates of the intersection point of the triangle plane with the line segment from the virtual camera to the billboard position. Let $\mathbf{p}$ be the virtual camera position, $\mathbf{b}$ the billboard position, $\mathbf{v}_i$ the position of the triangle vertex i, and λ_i the weight of c_{v_i}. Then $\lambda = [\lambda_1, \lambda_2, \lambda_3]^t$ can be determined by the line-plane intersection equation

$$\begin{bmatrix} t \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \mathbf{p} - \mathbf{b} & \mathbf{v}_2 - \mathbf{v}_1 & \mathbf{v}_3 - \mathbf{v}_1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p} - \mathbf{v}_1 \end{bmatrix} \quad (5)$$

$$\lambda_1 = 1 - \lambda_2 - \lambda_3 \quad (6)$$

The bounding box of the moving object for each camera is then used to determine which regions of the foreground textures have to be used to compose the billboard texture. Let R_i be the image region extracted from the foreground texture $F_{c_{v_i}}$. The final billboard texture O is defined as

$$O = \lambda_1 R_1 + \lambda_2 R_2 + \lambda_3 R_3 \quad (7)$$

III. Experimental Results

For the purpose of testing the described method, an interactive system was implemented in C++ using the OpenGL library for rendering and the OpenCV library for handling the video sequences and performing image operations. The relevant specifications of the test machine were: Intel Core 2 Duo 2.0GHz processor, 3GB of RAM and a NVidia 8600M GT graphics board.

We performed tests using video sequences from two distinct scenarios. The first video sequence was composed of footage from a parking lot. This sequence was captured using four cameras with distinct viewpoints and the resulting videos had the resolution of 320×240 pixels. Images from the viewpoint of each camera are shown in Figure 9. For the second video sequence, we used the benchmark data-sets available from the PETS 2006 Workshop [21]. This sequence is composed of videos from four cameras with 768×576 pixels of resolution. For our tests, we discarded camera number 2 because the region of interest was too small and wouldn't improve significantly the rendering results. Images from this scene are shown in Figure 11. We also used additional images from the PETS 2009 Workshop [22] to test the color calibration algorithm since some of those images present strong color and brightness differences.

For the parking lot video sequence, the system was capable of rendering novel views at the rate of 32.72 frames per second. For the PETS 2006 video sequence, it was capable of rendering novel views at the rate of 11.54 frames per second. Both results were produced by caching all video frames in memory before rendering. The difference in the frame rates of the two sequences is mainly due to the difference in video resolution. Since it is necessary to upload two new 32-bit textures per video every frame to the graphics board memory, videos with higher resolutions require more texture upload bandwidth in order to maintain high frame rates. By uploading all needed video frame textures before rendering, the system is capable of rendering both scenes at more than 50 frames per second.

Results for the color calibration algorithm can be seen on Figure 8. We can observe that even in cases of severe color and brightness differences, the algorithm is capable of producing calibrated images with color and brightness similar to the reference images.

A step-by-step image description of the rendering process can be seen on Figure 10. Figure 10a shows the start of the rendering process where the 3D scene model of the parking lot is rendered from the viewpoint of the virtual camera. Figure 10b shows the second rendering step where the video with best matching viewpoint is projected over the model. We can observe the sharp edges of the video projection and the color differences between the projection and the model. In Figure 10c, the edges of the projection borders were smoothed and the moving objects were removed. The holes left by the moving objects are clearly visible in the image. In the next step, the color calibration transformation is applied over the video frame and the result is seen on Figure 10d. Then, the video projection and the holes are better blended with the 3D model. In the final step, the moving objects are rendered as billboards and the results can be seen on Figure 10e and Figure 10f.

Rendering results for the PETS 2006 sequence can be seen on Figure 12. In Figure 12a and 12b, we can observe rendering artifacts due to segmentation errors. Also, in Figure 12c, there is another type of artifact that can be seen on the group of people to the left. This artifact appears during texture generation for the billboards when the bounding box of an

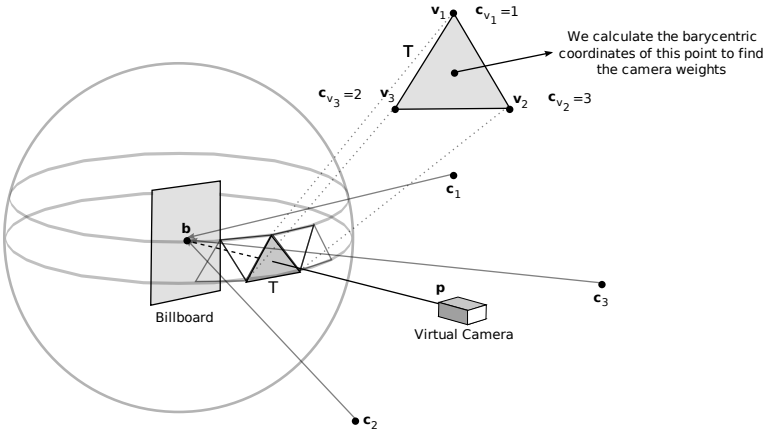


Figure 7. View-dependent Texture Mapped Billboard

(a) (b) (c)

(d) (e) (f)

Figure 8. Color calibration algorithm results using $T_0 = 256$, $K = 100$, $\epsilon = 10^{-5}$. The left column shows the original images that need to be calibrated, the right column shows the target reference image that contains the expected color and brightness for the correction and the middle column shows the calibrated images.

object is occluded by the bounding box of another object. The resulting billboard texture for those objects will share the overlapping region of their bounding boxes resulting in parts of those objects being replicated on the texture of both billboards.

IV. Conclusion

We introduced a real-time free viewpoint video (FVV) method that allows for the rendering of novel scene views at interactive frame rates using sparse cameras. Scene modelling and moving object segmentation and tracking is performed offline. We use a multi camera homography constrained object segmentation algorithm that is robust to severe occlusion between objects. The 3D position of the tracked objects are used to render them in 3D at their correct location using view dependent texture mapped billboards. Foreground masks from object segmentation are used to remove artifacts caused by the moving objects from the virtual video stream background.

To minimize color artifacts when combining multiple sparse camera video streams, we have also developed a

Figure 9. Images from the parking lot scene

(a) 3D scene model (b) Video projection (c) Moving objects removal

(d) Color calibration (e) VDTM billboards (f) Novel view rendering

Figure 10. Steps of the parking lot scene rendering

simulated annealing color correction algorithm that finds monotonicallly increasing polynomials that optimizes the histogram transformation between cameras and the scene model.

The final quality of the rendered views is similar to that of the source videos and the system allows a user to navigate the rendered 3D scene in time and space.

Future work includes the detection and handling of the occlusions between billboards to remove phantom artifacts. It is also possible to improve the rendering quality by using a per pixel view-dependent texture mapping method instead of using projective texture mapping to blend video and model.

REFERENCES

[1] C. Zhang and T. Chen, "A survey on image-based rendering – representation, sampling, and compression," *Signal Processing: Image Communication*, vol. 19, no. 1, pp. 1–28, Jan 2004.

[2] M. R. Civanlar, J. Ostermann, H. M. Ozaktas, A. Smolic, and J. Watson, "Special issue on three-dimensional video and television," *Signal Processing: Image Communication*, vol. 22, no. 2, pp. 103–107, Feb 2007.

[3] E. Adelson and J. Bergen, "The plenoptic function and the elements of early vision," in *Computational models of visual processing*, M. Landy and J. Movshon, Eds., The MIT Press, 1991, p. Chapter 1.

[4] P. E. Debevec, Y. Yu, and G. D. Borshukov., "Efficient view-dependent image-based rendering with projective texture-mapping," in *Proceedings of the Eurographics Rendering Workshop*, G. Drettakis and N. Max, Eds., Vienna, Austria, June 1998, pp. 105–116.

[5] S. Kang, M. Wu, Y. Li, and H. Shum, "Large environment rendering using plenoptic primitives," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 11, pp. 1064–1073, Nov 2003.

[6] G. Willems, F. Verbiest, M. Vergauwen, and L. Gool, "Real time image based rendering from uncalibrated images," in *Proc. of the Fifth International Conference on 3D Digital Imaging and Modeling - 3DIM'05*, Ottawa, Canada, June 2005, pp. 221–228.

[7] K. Mueller, A. Smolic, P. Merkle, B. Kaspar, P. Eisert, and T. Wiegand, "3d reconstruction of natural scenes with view-adaptive multi-texture," in *In Proc. of 3DPVT*, 2004, pp. 116–123.

Figure 11. Images from the PETS 2006 scene

(a) Rendering 1 (b) Rendering 2 (c) Rendering 3

Figure 12. Novel view renderings from the PETS 2006 scene

[8] M. Sainz, R. Pjarola, and A. Susin, "Photorealistic image based objects from uncalibrated images," in *Posters of the IEEE Visualization Conference - VIS'03*, 2003.

[9] J. Starck and A. Hilton, "Virtual view synthesis of people from multiple view video sequences," *Graphical Models*, vol. 67, pp. 600–620, 5 2005.

[10] J. Carranza, C. Theobalt, M. A. Magnor, and H. P. Seidel, "Free-viewpoint video of human actors," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 569–577, 2003.

[11] D. Min, D. Kim, and K. Sohn, "Virtual view rendering system for 3DTV," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, 2008, pp. 249–252.

[12] R. Y. Tsai, "A efficient and accurate camera calibration technique for 3d machine vision," *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 86)*, pp. 364–374, 1986. [Online]. Available: http://ci.nii.ac.jp/naid/10012692972/en/

[13] H. Wang and D. Suter, "A re-evaluation of mixture of gaussian background modeling," in *Proceedings of 30th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, vol. 2, 2005, pp. 1017–1020.

[14] W. Hu, M. Hu, X. Zhou, T. Tan, J. Lou, and S. Maybank, "Principal axis-based correspondence between multiple cameras for people tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 4, pp. 663–671, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1597123

[15] K. Kim and L. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering," in *Proceedings of 9th European Conference on Computer Vision (ECCV'06)*, vol. 3953, Graz, Austria, 2006, pp. 98–109.

[16] T. T. Santos and C. H. Morimoto, "People detection under occlusion in multiple camera views," in *Proceedings of the XXI Brazilian Symposium on Computer Graphics and Image Processing - SIBGRAPI '08*, Campo Grande, MS, 2008, pp. 53–60. [Online]. Available: http://dx.doi.org/10.1109/SIBGRAPI.2008.25

[17] A. Criminisi, I. D. Reid, and A. Zisserman, "Single view metrology," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 123–148, 2000.

[18] F. M. Porikli, "Inter-camera color calibration by correlation model function," in *ICIP (2)*, 2003, pp. 133–136.

[19] M. Zhang, J. Xie, Y. Li, and D. Wu, "Color histogram correction for panoramic images," *Virtual Systems and MultiMedia, International Conference on*, vol. 0, p. 328, 2001.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[21] P. 2006, "Pets 2006 benchmark data," http://www.cvg.rdg.ac.uk/PETS2006/data.html.

[22] P. 2009, "Pets 2009 benchmark data," http://www.cvg.rdg.ac.uk/PETS2009/a.html.