Interactive Acquisition and Rendering of Humans

Bert De Decker and Philippe Bekaert
University of Hasselt - Expertise Centre Digital Media
Diepenbeek, Wetenschapspark 2, Belgium
Email: {bert.dedecker,philippe.bekaert}@uhasselt.be

***Abstract*— We present a distributed model-based system for interactive acquisition and rendering of free viewpoint video of humans. All current model-based systems are offline. Online systems which can be used for free viewpoint video of humans use few low resolution cameras or use only silhouette information while the method presented here can cope with high resolution images and uses both color and silhouette information. A person is captured with multiple synchronized and calibrated digital video cameras attached to a cluster of pcs. At interactive rates, the shape of the person is estimated and the person can be rendered from novel viewpoints. First the pose of the person is obtained using a real-time motion capturing algorithm. A generic mesh of a human is transformed into this pose to obtain a first approximation of the shape of the recorded person. This mesh is refined to match the input images as closely as possible by translating its vertices along their normals. The refined mesh is rendered using a distributed version of the unstructured lumigraph algorithm.**

I. Introduction

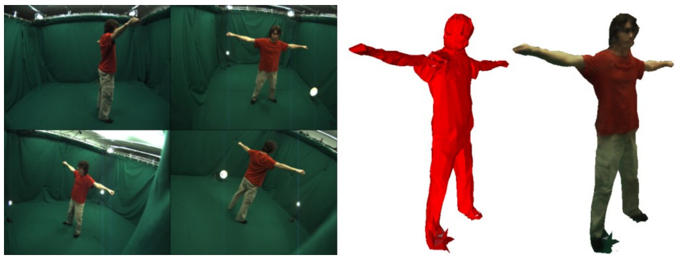
In this paper we present a distributed model based system for interactive acquisition and rendering of free viewpoint video of humans. A person is captured with multiple synchronized and calibrated digital video cameras. From these recorded images a mesh, resembling the shape of the person, is estimated using a model based approach. Both silhouette and color information are used to calculate the mesh. Using this mesh we are able to render the person from an arbitrary viewpoint using unstructured lumigraph. Our system is able to deal with images from 12 cameras at a of resolution 1024x786 at interactive rates because it doesn't need all the pixels. We only sample the images at some interesting positions and know where to sample because a model based approach is used.

In movie production, such a system can be used to compose a person into a hostile environment and preview the results during recording. This way the scene can be altered immediately if necessary. Another application would be to play a computer game featuring ones own body.

To our knowledge, we are the first to present an interactive model-based system for acquisition and rendering of free viewpoint video of humans.

II. Related work

Some model based systems for free viewpoint video of humans have been proposed. In the work by Carranza et al. [1] only motion capturing is done to obtain the shape of the person, and this is not refined later on. An improvement of the work by Carranza et al. that does change the shape after motion capturing to make it better fit the input images is presented by Aguiar et al. [2]. Another improvement to the work by Carranza et al. is presented where it becomes possible to relight the person [3]. The previous three mentioned methods don't use skinning, every limb is presented as a rigid body. The work presented in this paper is most similar to the one presented by Starck et al. [4]. In that work one also tries to fit a mesh to input videos by using motion capturing to obtain a first approximation of the shape of the person and refine that shape later on by using silhouette and color information. They also use skinning to represent their model, but their system is offline. Chueng et al. [5] describe an offline system to scan humans. A system that estimates animatable human models from range scan data is presented by Allen et al. [6].



The main difference between the method presented here and other model based methods is that they are all offline while our method is an interactive online system.

Some online systems have been presented, but they only use few low resolution cameras or they only take silhouette information into account. An online system that calculates visual hulls is presented by Matusik et al. [7]. This system can be used for free viewpoint video of humans, but it works with low resolution images and uses only silhouette information. Another online system based on point clouds has been developed by Würmlin et al. [8]. They use a distributed system to calculate a point cloud representation of the visual hull. Li et al. [9] describe an interactive system that calculates the visual hull and refines it later on using color information.

Numerous methods have been proposed for online rendering in a small baseline setup [10]. But these systems only allow a small change of camera position while our system is able to do true free viewpoint video.

III. Overview

In this section the system will be discussed in general, more details of every part will be given in later sections.

We use one single mesh to represent a person. The mesh can be transformed into every possible pose by using skinning [11]. This is a technique that assigns each vertex of the mesh to one or more limbs of the skeleton. When a limb is moved, the vertices assigned to it move with it. Weights are used to determine how much influence a certain limb has on a vertex.

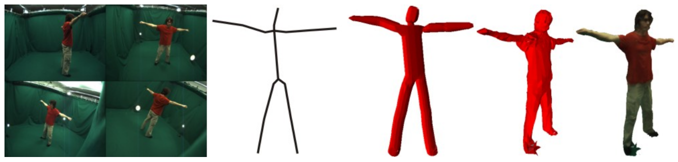


Fig. 1. Different steps in the algorithm. From left to right: input images, pose estimation, transformation of generic human mesh, mesh refinement, texturing.

Our system can be considered a pipeline consisting of a number of steps. In Figure 1, the result of each step is shown. The steps are as follows.

- Capturing Images
- Motion Capturing: First the pose of the person in the input images is estimated using the realtime motion capturing algorithm presented by Caillette et al. [12]. It is fast enough for our purposes, but this speed comes at a cost. The pose isn't very precise and shivers even if the person that is being motion captured isn't moving at all. Most of these imperfections are eliminated during the mesh refinement phase.
  Using skinning, a mesh of a generic human is transformed into the pose estimated by the motion capture algorithm. This transformed mesh is a first crude approximation of the shape of the person in the input images.
- Mesh Refinement: The transformed generic human mesh from the motion capturing step is refined to better approximate the shape of the person in the input images. This is done by translating its vertices along their normals. A more detailed discussion of this process can be found in section IV.
- Rendering: Unstructured lumigraph [13] is used to render and texture the refined mesh calculated during the mesh refinement step. Section V elaborates on this.

The motion capturing step and the mesh refinement step need silhouette information, but they both only need it at some pixels. They don't need silhouette information for all the pixels of the images. So when we need to know whether a pixel is foreground or background, the segmentation algorithm is executed locally for this pixel only. We never calculate the segmentation for the entire image. A simple background subtraction algorithm is used. To facilitate the segmentation process, we worked in a green screen environment.

In our implementation we have, As is shown in Figure 2, four pcs which each have three cameras attached to them, henceforth we refer to these pcs as "camera nodes". A distributed approach is used where the camera nodes do all the computations concerning the images of the cameras attached to them, and send partial solutions to a central module that calculates the final solution from these partial solutions. For every step of our algorithm, we have one dedicated module to combine all the partial solutions. As is visible in Figure 2 the following modules are present in our system: a motion capture module, a mesh refinement module and a render module. These modules only have to gather and combine the partial solutions, so they don't need that much resources and can all run on one pc without a problem. The cameras are calibrated using a matlab toolbox by Svoboda et al. [14].

Not all the images of all the cameras are needed for doing the motion capturing and full resolution images aren't needed either. So, every camera node calculates a low resolution image for two of its cameras and sends these images to the motion capture module. Using these images the motion capture module estimates the pose of the person.

All the camera nodes work in parallel with each other and all the steps of the pipeline are executed simultaneously in parallel. There are always 4 frames present in the pipeline: one is being captured by the cameras, one is being motion captured, yet another one is in the mesh refinement phase and finally the fourth one is being rendered. By processing this many frames in parallel we are able to process more frames per second, at the cost of an increased lag in our system.

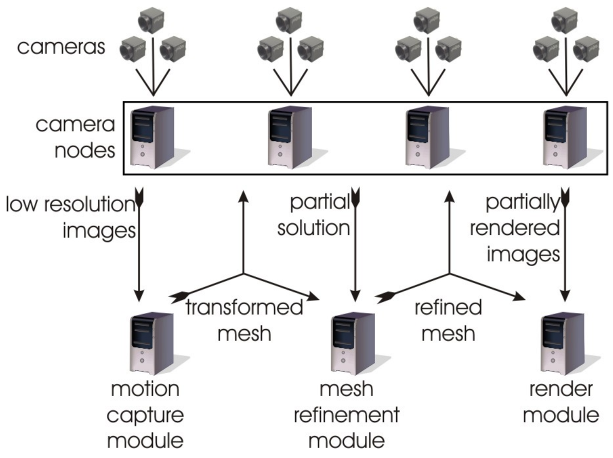


Fig. 2. Overview of the network setup and the network traffic between the components of the system.

IV. Mesh Refinement

In this section is explained how the mesh, obtained from the motion capture phase, is adjusted to match the input images as closely as possible.

The mesh is improved by translating its vertices along a line defined by the normal of the vertex. Improving the mesh now boils down to calculating the translation distances. For every vertex a fixed set of translation distances is considered. The translation distances are uniformly distributed in the interval $[-t_{max}, t_{max}]$ with t_{max} a user defined parameter. For every translation distance an error is calculated. The smaller the error, the higher the probability of the translation distance being the correct one.

The error function is a sum of four terms. One term makes sure all the vertices lie inside or on the visual hull. Another term favors large translation distances, it is a cheap heuristic which prefers translated vertices near the visual hull. If a translated vertex is projected to all cameras for which it is visible, the colors of all these projected points should be as similar as possible. The third term takes care of this. To decide whether a vertex is visible from a camera, the mesh is rendered from this camera and a low resolution depthmap is extracted. The vertex is visible if its depth matches the depthmap. When using only the previous three terms, sometimes multiple local minima occur, as can be seen in Figure 3. Each minimum corresponds to a 3d point on the surface of the real person. When this occurs, the last term chooses the minimum closest to zero or, put differently, the translated vertex closest to the mesh obtained by the motion capturing step.

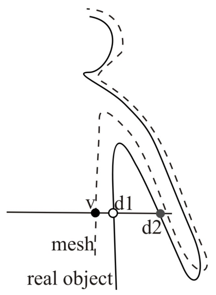


Fig. 3. Vertex v that belongs to the torso has two valid translation distances, d1 on the torso and d2 on the arm. The last term of the error function favors d1, the one on the torso.

The camera nodes extract all the needed information from the input images and send only this information to the mesh refinement module. The mesh refinement module uses this information to calculate the error function and select the best translation distance for each vertex.

The camera nodes need to calculate and send the color for each translated vertex for each camera, the average color of each translated vertex, the number of cameras for which each translated vertex is projected inside the silhouette, and visibility information for each vertex.

All this data fits well on a gigabit ethernet network and it needs less than one tenth of the bandwidth needed to send the raw images.

than sending the images, even if we are able to compress them ten times.

V. Distributed Rendering

Unstructured lumigraph [13] is used to render the mesh with texture information from the input images. Every vertex of the mesh is given a texture coordinate and a blending weight for each camera. The texture coordinates are calculated simply by projecting and distorting the vertex for every camera. The blending weight for vertex v, input camera c_i and desired camera c_d is a function of 2 terms. On one hand the angle between the surface normal at v and the vector connecting v and c_i. This make sure that the input cameras that see the vertex head on provide much texture information, and input cameras that see the vertex not at all or at a small angle are ignored. On the other hand, the angle between the vector connecting v and c_d and the vector connecting v and c_i. This makes sure that when de desired camera is very close or on an input camera, almost all texture information is gathered from this input camera. The final image is obtained by rendering the mesh from every input camera and blending all these rendered images together using the blending weights.

Since it is not feasible to send all the images to one pc for processing, a distributed unstructured lumigraph algorithm is required. The refined mesh is distributed to all the camera nodes and every camera node calculates the unstructured lumigraph using only the images of the cameras attached to it. This is done by means of a pixel shader. To render one pixel, the pixel shader takes as input the texture coordinates of all cameras and a blending weight for each camera. The alpha value of the pixel is the sum of all the blending weights. The color of the pixel is calculated by taking the weighted average of the colors of the pixels from the input cameras using these blending weights. The colors along with the alpha values of this rendered image are compressed using run length encoding and are sent to the render module.

Finally, the images received from the camera nodes are blended using a pixel shader. The color of a pixel is calculated by normalizing all the alpha values of the received images and calculating the weighted average of the colors using these weights.

VI. Results

All results are generated with four camera nodes which each have three cameras attached. The motion capture module, the mesh refinement module and the render module ran on one pc, so in total five pcs were used. The generic human body mesh we used, consists of 2000 vertices and 21 translation distances are considered. The input images are 1024x768 pixels and the system runs at 3 to 4 frames per second when rendering images of 640x480 pixels. The camera nodes are the slowest part of our system. No real bottleneck is present in the system, all components of the pipeline are approximately equally fast.

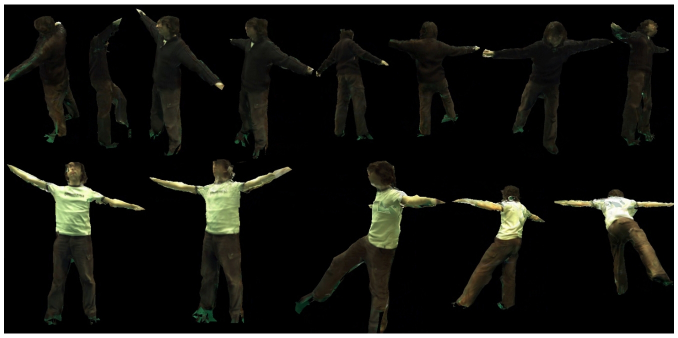


Fig. 4. Some frames of two movies where the camera rotates around the person being captured.

The algorithm was tested on real world data. Some frames of a movie generated by our system are shown in Figure 4.

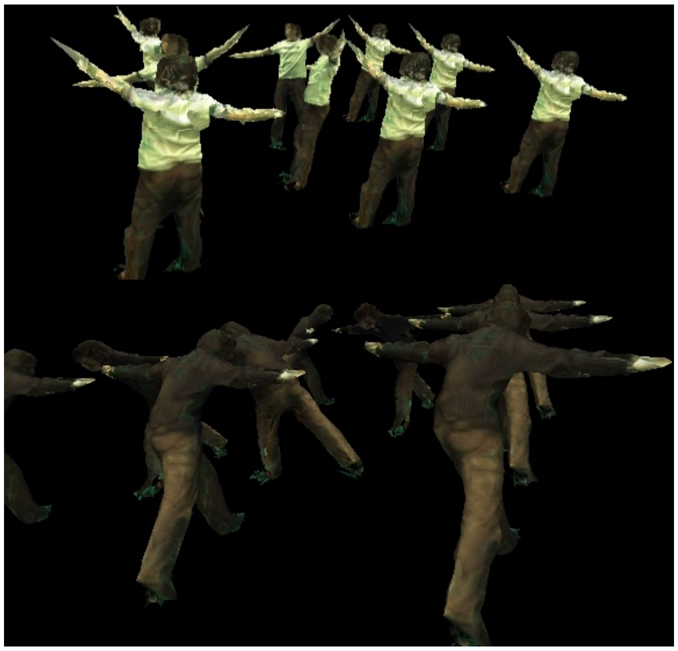


Fig. 5. Actor cloning is possible with negligible loss of speed.

Our scene representation permits 3D video editing. An example of actor cloning where the person is copied to other positions in the scene is shown in Figure 5. By construction, the algorithm is able to do this with negligible loss of speed.

VII. Conclusion and Future Work

We presented a distributed approach for interactive acquisition and rendering of 3D video of humans. Such a system could be used for generating content for a live 3D TV application or it could be used in a computer game.

A first approximation of the shape of the person was obtained by transforming a mesh of a generic human into a pose estimated by a real-time motion capturing algorithm. A distributed method to refine and render this mesh was presented. The vertices of the mesh were translated along their normals to approximate the shape of the recorded person as closely as possible. Finally this refined mesh was rendered using a distributed version of the unstructured lumigraph algorithm.

Currently no threading is used in our implementation, so our system spends some time waiting for the network, the cameras and the graphics hardware. When threading is implemented, this time could be used for useful things and a speedup is expected. Most of the artifacts visible in our results are because the motion capturing isn't very precise. More research could be done to improve this. After the mesh refinement step, a reasonable shape of the person is obtained and this information could be used for better realtime motion capturing.

VIII. Acknowledgements

The authors acknowledge financial support on a structural basis from the ERDF (European Regional Development Fund), the Flemish Gouvernment and the Flemish Interdisciplinary institute for BroadBand Technology (IBBT), and from a research grant by ”tUL impulsfinanciering”.

References

[1] 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.

[2] E. de Aguiar, C. Theobalt, M. Magnor, and H.-P. Seidel, “Reconstructing human shape and motion from multi-view video,” in *To appear in: Proc. of IEE CVMP*, (London, UK), 2005.

[3] C. Theobalt, N. Ahmed, E. de Aguiar, G. Ziegler, H. Lensch, M. Magnor, and H.-P. Seidel, “Joint motion and reflectance capture for creating relightable 3d videos,” *Technical Report MPI-I-2005-4-004, Max-Planck-Institut fuer Informatik, 2005*.

[4] J. Starck and A. Hilton, “Model-based multiple view reconstruction of people.,” in *ICCV*, pp. 915–922, 2003.

[5] K. M. Cheung, S. Baker, and T. Kanade, “Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 77–84, June 2003.

[6] B. Allen, B. Curless, and Z. Popović, “Articulated body deformation from range scan data,” vol. 21, no. 3, pp. 612–619, 2002.

[7] W. Matusik, C. Buehler, and L. McMillan, “Polyhedral visual hulls for real-time rendering,” in *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, (London, UK), pp. 115–126, Springer-Verlag, 2001.

[8] S. Würmlin, E. Lamboray, and M. Gross, “3d video fragments: Dynamic point samples for real-time free-viewpoint video,” in *Computers and Graphics 28*, vol. 1, pp. 3–14, 2004.

[9] M. Li, H. Schirmacher, M. A. Magnor, and H.-P. Seidel, “Combining stereo and visual hull information for on-line reconstruction and rendering of dynamic scenes,” in *Proceedings of the 5th Conference on Multimedia Signal Processing*, (St. Thomas, US Virgin Islands), pp. 9–12, IEEE, IEEE, 2002.

[10] R. Yang, G. Welch, and G. Bishop, “Real-time consensus-based scene reconstruction using commodity graphics hardware,” in *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, p. 225, October 2002.

[11] M. Deloura, *Game Programming Gems*. Rockland, MA, USA: Charles River Media, Inc., 2000.

[12] F. Caillette and T. Howard, “Real-Time Markerless Human Body Tracking with Multi-View 3-D Voxel Reconstruction,” in *Proceedings of British Machine Vision Conference (BMVC)*, vol. 2, pp. 597–606, September 2004.

[13] C. Buehler, M. Bosse, S. Gortler, M. Cohen, and L. McMillan, “Unstructured lumigraph rendering,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 425 – 432, 2001.

[14] T. Svoboda, D. Martinec, and T. Pajdla, “A convenient multi-camera self-calibration for virtual environments,” *PRESENCE: Teleoperators and Virtual Environments*, vol. 14, pp. 407–422, August 2005.