Real-Time Relighting of Video Streams for Augmented Virtuality Scenes

Till Davin and Jens Herder

Hochschule Düsseldorf - University of Applied Sciences
Münsterstraße 156, 40476 Düsseldorf
Tel.: +49 211 4351-3268, E-Mail: jens.herder@hs-duesseldorf.de
Video: vsvr.medien.hs-duesseldorf.de/publications/gi-vrar2021-relighting

Abstract: The combination of real and virtual image elements is inevitable in virtual productions and mixed reality applications. One major problem in combining real and virtual elements lies in the visual discrepancy between virtual and real environment. We present an implementation of real-time relighting of a video stream with the light setting of a virtual scene. A system is realized that utilizes image processing and modern graphics calculations to relight the input video stream with additional depth information. A two-dimensional contour is extracted from the video image through image processing. The contour is then used to generate a displacement texture, which gives the two-dimensional video plane three-dimensional depth. The video stream then can be relit with the virtual light in the scene. An integrated markerless motion-tracking system creates the possibility to correctly position the video with geometry in the scene. Image results, advantages, and disadvantages of the proposed system are discussed and evaluated.

Keywords: real-time rendering, markerless motion tracking, virtual studio, augmented virtuality, relighting, light estimation

1 Introduction

Composition of virtual and real image elements has already become a standard technique may it be in professional cinema production with a green screen, an augmented reality app on a smartphone, a virtual presentation of a new car model or software post-processing of a photo. The concept of *Virtual Production* is finding its way into professional film productions and virtual content can be integrated directly into live production. This makes it much easier to control and revise content quickly. Direct feedback for the director, camera operators and actors allows for more effective iterations and adjustments. Incorporating more complicated techniques into production, such as motion capturing or elaborate 3D image calculations with direct feedback, is now possible with simpler and more cost-effective means.

In the rapidly growing field of augmented reality for industrial and entertainment applications, the integration of real video recordings into a virtual environment is also central. These applications are fundamentally designed as real-time applications and are intended to allow the user to interact with the environment and provide direct feedback. Thus, the input video signal is also integrated into a virtual scene in real-time. A seamless and high-quality embedding of the video image increases immersion and usability. For example, correctly calculated shadows and occlusion also give the viewer additional information to spatially orient himself in the application.

A major problem in the composition of virtuality and reality is visual discrepancies between virtual and real content, whether in the form of different color and lighting moods, different light scenes and shadows, or incorrect object occlusion. Despite generally excellent image quality, combined shots can quickly look artificial and unrealistic as a result. This is where challenges in virtual production and augmented reality still lie. A believable blending of the different contents requires not only a good quality of the elements themselves, but also a process to realistically align the individual elements with each other to create a coherent and believable overall image.

The developed system implements a real-time lighting adjustment of a video texture with the lighting mood of a virtual scene. This adjustment is done in a real-time environment and thus is capable of being integrated in different mixed reality applications and productions. The main research questions revolve around the feasibility and quality of a real-time illumination calculation.

2 Related Research

[GDT+19] presents a light stage that enables photorealistic relighting of a motion sequence of a person. The complex recording system consists of a sphere of 58 RGB and 32 depth cameras, as well as 331 LED rings that spherically surround the actor. Complex processing of the camera images using neural networks enables a volumetric three-dimensional reconstruction of the actor. High-resolution textures and an illumination model of the surface are also captured. The recorded data allows for realistic re-lighting of the 3D model of the actor in an arbitrary virtual environment. LED rings can be used to illuminate the subject with two different color gradients. The color gradients are used to generate a reflection model. In contrast to a uniform, diffuse illumination of the person with white light, the recorded textures of the color gradients contain additional the reflection properties like albedo, normals, and gloss of the surface. The system requires memory of 650 GB and eight hours of computing time in a highly optimized computer network for a 10-second or 600-frame recording. With the Free-form Light Stage, [MDA02] present a simpler technical setup of such a light stage. A portable light source is used to illuminate an object from different directions while shooting from a static camera perspective. From small diffuse spheres next to the object, the incident direction of the light can be estimated afterwards. The captured perspective can be re-lit using these images with an arbitrary environment map.

Relighting of a video sequence in a virtual studio was implemented by [Gra06]. A multi-camera system capture person geometry while assuming a diffuse illumination model by segmenting the foreground by chroma keying and then computing the visual envelope. An environment map is set up as a diffuse illumination model. A simplified form of a reflection model is described to calculate the light intensity of a pixel from a diffuse and a specular part. The specular part is neglected, because accurate surface normals would be necessary for this. The diffuse component is the integral of all incoming light rays, taking into account the direction of incidence and the surface normals. The intensity of the incident light rays is obtained from an environment map. This environment map is a spherical high contrast image (HDR).

A multi-camera system with video streams from different perspectives can be used to generate the 3D geometry and surface features of a person, as presented by [TAL$^+$07] and [CVH07]. Multiple synchronized and calibrated video cameras are used as input to the system. The acquisition of the geometry and skeleton of the person takes place through markerless person tracking.

Similar [SWHG10], first compute a reflectance model from static reference footage. The model can then be used to re-light dynamic video footage and also allows for changes in material properties, such as adjusting the color of a person's clothing. It deals with transferring both reflectance properties and fine shading details of loose clothing from the static reference image to the dynamic video image. A local nearest-neighbor search is used to establish a connection between the reference shot and the target video. This search is spatially consistent only in textured image regions, but still allows for correct transfer of color values because multiple representations of the same color value exist in single-color image regions. [ISS14] use the depth information from a *Microsoft Kinect* to estimate the normals of an actor and separate it from the background. Using the obtained normals in combination with an environment map, the video texture is relit.

[HNK07] also use an environment map to transfer the real occurrences of light to virtual objects. Using two fisheye cameras, the spatial positions of the studio lights were captured, as was done by [FKGK05]. The estimated light sources were used for generating shadows using shadow maps.

Most of the presented work shows a high computational and hardware effort for illumination adjustments. The large number of unknown variables implies a high resource cost to achieve good image quality. Systems that process input images in real time therefore mostly re-light virtual objects based on the previously acquired real lighting environment, since the necessary geometry information is already available for these virtual objects. Re-illumination of real objects or persons requires additional acquisition of geometry and surface information. The quality of the new illumination is therefore strongly dependent on the quality of the captured geometry.

3 Relighting System

In the following, the proposed relighting system will be described and explained in its functionality. The structure of the system as a whole is shown in figure 1.

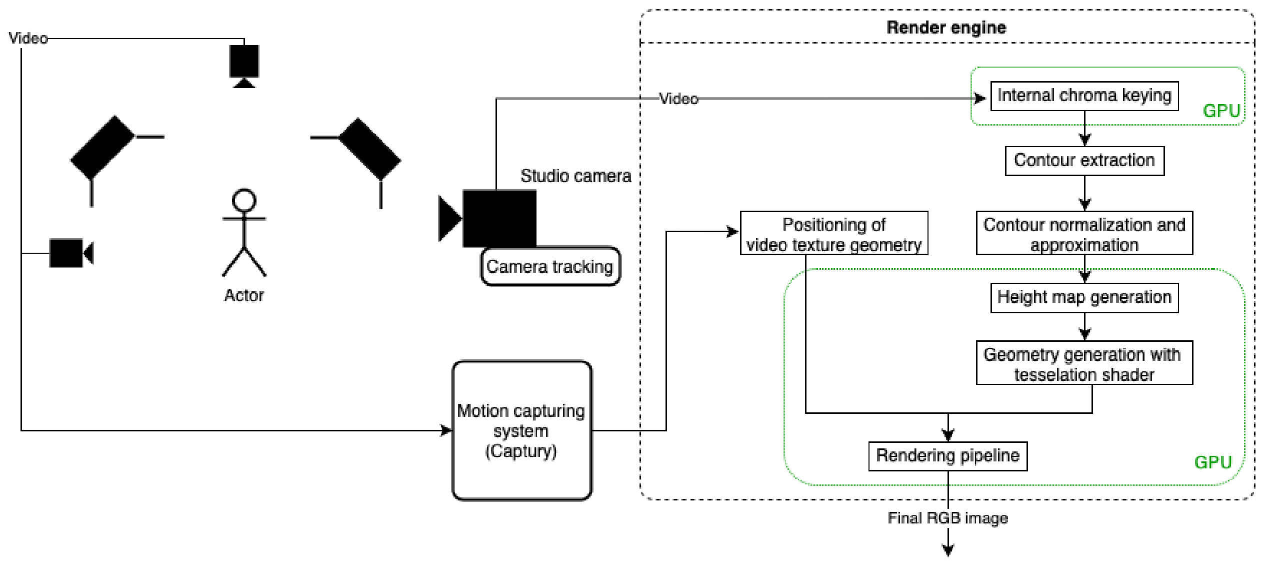


Figure 1: Structure of the implemented system

Geometry reconstruction from visual hull Human geometry requires complex adjustments to match a generic geometry with acceptable results. Elaborate algorithms would be necessary, but again difficult to reconcile with the real-time demands of the system. Therefore, we resort to the alternative of generating the geometry solely from the two-dimensional video texture. This greatly simplifies geometry creation, but makes it difficult to accurately describe the geometric surface. Thus, to describe the depth of the geometry, a simple dependency of the depth z of the vertices to the distance to the contour d was defined according to equation $z = 1 - (1/(1 + d * b))$. The parameter b adjusts the strength of the effect.

Keying To integrate the video texture into the virtual scene, it is necessary to punch out the relevant parts of the image from the video texture. For this purpose, a chroma keying method in form of a shader program is used for recordings made in front of a green screen. To compensate for the sometimes strong spill of green light onto objects in a greenbox, the shader was extended with a spill suppression. This spill suppression limits the green value of a pixel to the maximum value of the red and blue channels. This prevents a strong green tint in the final video image. By limiting the green value, the affected pixel has a lower brightness. This lost brightness is added back as a luminance value to modify only hue and not brightness of the pixels.

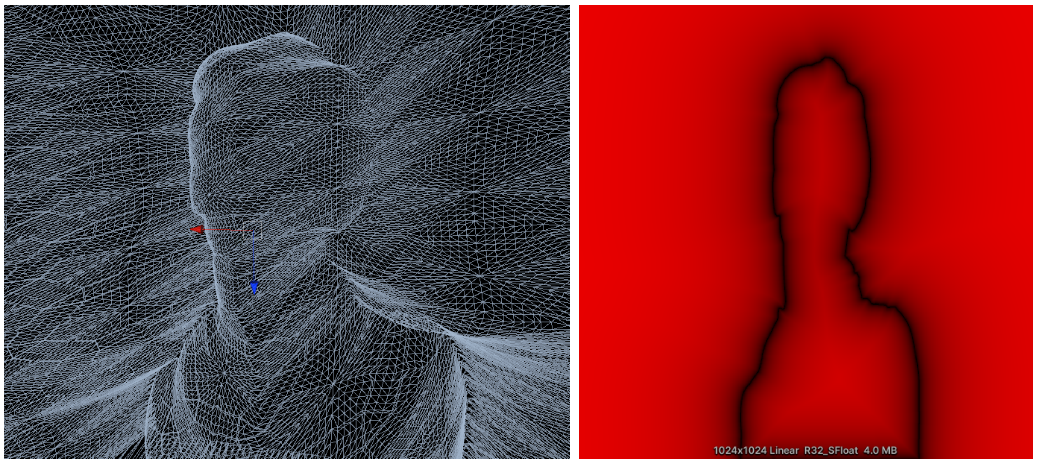
Video texture transport from GPU to CPU for contour extraction The texture keyed by a shader program is only available on the memory of the graphics unit. For contour extraction it must be available to the CPU, so it is necessary to move the texture data to other memory areas. This data shifting is comparatively costly. Furthermore, a synchronous access of the CPU to GPU memory requires a stop of CPU and GPU processing and is therefore not very suitable for real-time applications, especially if the data shift has to be performed in every render cycle. To implement the presented system, an asynchronous read-back of the texture data from the GPU to the CPU is used. This asynchronous read-back has the advantage of not requiring other computations to be paused for long periods of time, but it introduces more delay. As a result, a captured video texture is only integrated into the graphics pipeline and ultimately displayed with an additional delay. Once the video texture is asynchronously downloaded from the GPU, however, the further listed processing and the re-upload of the processed texture to the graphics unit can take place within a single render cycle.

Contour extraction All relevant contours can now be obtained from the video texture available for the CPU. For this purpose, an integration of the open source image processing library *OpenCV* was included in the *Unity* engine. *OpenCV* contains an algorithm with the function *findContours()* which captures all contours in a texture as polygons or arrays of two-dimensional points [Ope20]. The function expects a binary image as input, so the alpha channel is extracted from the keyed texture. Thus, the contours present in the texture have been extracted in form of two-dimensional vectors. The calculation of the contours requires little computing power even with many existing contours and hardly degrades the performance. However, the performance of later calculations strongly depends on the number of contour points found. Hence it can be useful to approximate the found contours.

Tesselation shader In the following step, a geometry is created from the extracted contour. To create the geometry as efficiently as possible, a tessellation shader is used. This part of the graphics pipeline creates additional geometry on the GPU by breaking down existing surfaces into smaller parts. Thus a simple planar surface with 10 x 20 triangles and 121 vertices is sufficient as geometry, more precise details are generated at run-time on the graphics unit (see figure 2). The vertices generated by the graphics unit can also be shifted in depth on the GPU. This step enables the generation of geometry that can be integrated into all lighting calculations of the graphics pipeline without compromises.

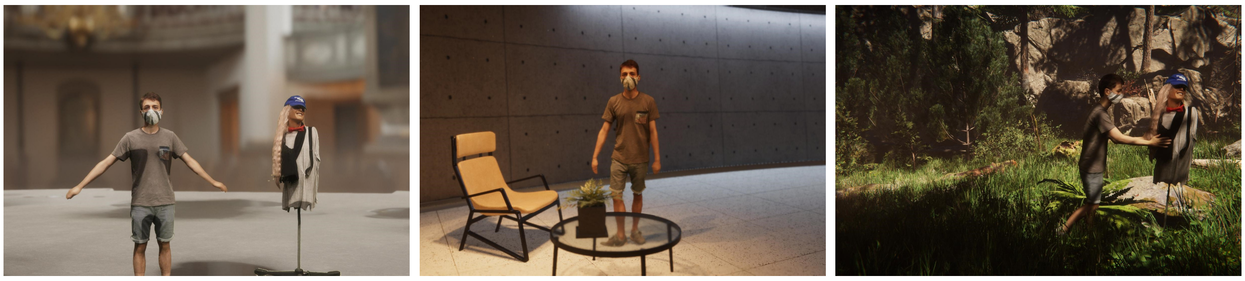
To provide information how the vertices should be shifted, a height map is necessary. Each pixel of this height map has to be assigned a depth value depending on the extracted contour. This filling can be optimally parallelized, which is why the use of a compute shader is recommended here. The implemented compute shader calculates the smallest distance of each pixel in the height map to the polygon contour. This distance is mapped to a final color value in the target texture.

Tracking and geometry placement The generation of geometry is only done in the local object space. Information about the global positioning of the object is missing. For this the markerless motion tracking system is used. For test purposes or objects that are not covered by motion tracking, a constant distance can also be specified. With markerless motion tracking, the distance of the actor to the camera is calculated. With this calculated distance, the planar surface is placed as a full screen plane in front of the virtual camera. The calculated distance, the frustum height of the camera and the image format are thus



(a) Wireframe (b) Height map 2048x2048 without uv-mapping

Figure 2: Generation of geometry from an associated height map of the mannequin in Table 2



(a) selfshadowing (b) occlusion (c) receiving shadow

Figure 3: Image results of the proposed system with a person and mannequin

used to determine the scaling and position of the plane. The correct rotation of the plane can be solved by simply appending the transformation node of the plane under the camera, the plane inherits the rotation of the camera.

Image rendering No additional operations are necessary to output the final RGB image, due to the full integration of the video texture into the graphics pipeline. The created geometry, textured with the video input signal, is lit by the virtual scene lighting just like any other existing virtual geometry. This allows for automatic shadow calculation as well as reflections or the application of post-processing effects. Some image results of the system in different virtual scenes are shown in figure 3.

4 Evaluation

To evaluate the implemented relighting system, the visual quality and the real-time suitability are assessed. To evaluate the real-time capability of the system, the additional image delay caused by the system and the general computational load, measured by the achievable frame rate, have to be recorded. In addition to the subjective comparison of images, various objective metrics exist for evaluating the visual quality of images. For the objective evaluation of the generated images of the system, the metrics SSIM (Structural similarity index measure) [WBSS04, HZ10], PSNR (Peak signal-to-noise ratio) [HZ10], and LPIPS (Learned perceptual image patch similarity) [ZIE+18] were used.

4.1 Performance measurements

Evaluating the performance of the system, a computer system with the following technical specifications was used: Unity 2019.4 with the High Definition Render Pipeline 7.5, AMD Ryzen 5 2600 CPU, six-core processor 3.40 GHz, 16 GB RAM, and NVIDIA GeForce GTX 1660 Super with 6 GB video memory. Static video recordings with a resolution of 1920x1080 pixels were used as input images for measuring the frame rate. The software tools *Profiler* and *Profile Analyzer* integrated in the *Unity* engine were used to record and analyze the measurements. The results of the measurements can be seen in figure 4.

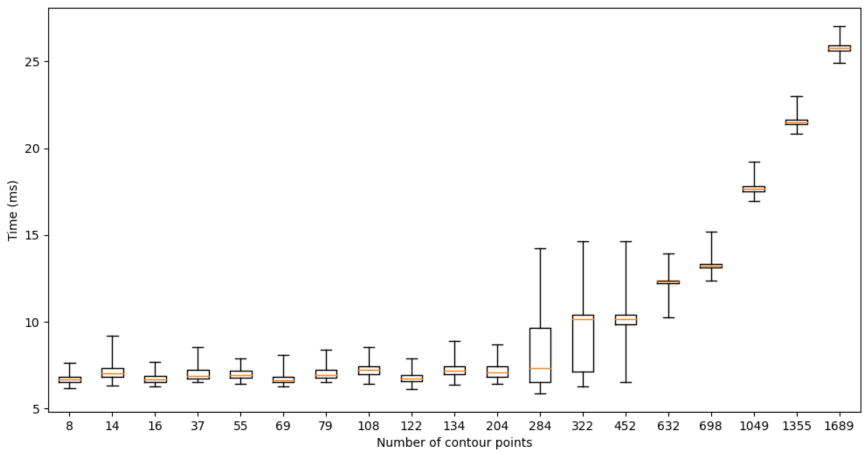


Figure 4: Box graphs of the results of the performance measurement. The time axis indicates the duration in milliseconds to calculate one render cycle.

The measurements show that the computational effort of the system depends on the number of contour points found in the image. With less than 1000 contour points, the above hardware is consistently able to achieve frame rates of over 50 frames per second. With more than 1000 contour points, the system's computation time increases significantly and fluctuations also occur more strongly. This is because the application is dependent on the graphics unit (GPU-bound) for its calculation with this amount of contour points. This means that the CPU has to wait for the graphics card to start another calculation cycle of an image. This heavier load on the graphics card is rooted in the computation of the compute shader program, which iterates over each contour point to calculate the smallest distance. The computation time of the compute shader is proportional to the number of contour points.

The relighting system introduces an additional image delay due to asynchronous access to video memory by the CPU. The delay caused by this memory access was measured separately, and the measurement results are listed in table 1. The measured values indicate the time in milliseconds that elapsed from the CPU requesting the texture until the texture could actually be accessed in memory. This delay was measured using texture objects with a resolution of 1920x1080 pixels, a color depth of 8 bit, and a memory size of 8 MB.

The measurement of the duration of the operation without calculating the relighting system (labeled *Isolated* in table 1) is 10.69 ms, well below the maximum calculation time span of 20 ms at 50 frames per second. Even in combination with the relighting system and a moderate number of contour points, the texture is still available in the same or next calculation cycle and thus produces a maximum delay of one frame. A high number of contour points also causes a greater delay, the reason for this is the high load of the graphics unit by the compute shader.

The performance measurements show that the system is suitable for real-time applications. However, the performance of the system is strongly dependent on the level of detail of the captured contours. The larger the contour data, the longer it takes to compute the compute shader to create the height map. To be able to guarantee a fast computation even in cases of large contour data, an adaptive approximation of the contour data could be implemented.

By requiring the CPU to access the video texture, the system introduces an additional delay of approximately 20 ms. For large contour data delays of up to 60 ms were measured. To this end, it should be noted that the asynchronous download of the video texture from the GPU can be circumvented by of accessing the video texture closer to hardware level. The implementation of this work uses middleware that makes the input signal from the video card only available to the graphics unit and therefore demands the asynchronous access. A direct access of the CPU to the video card is more complex to implement, but would make this process unnecessary. Thus, the system would not cause any additional delays.

Number of contour points	Isolated	16	55	122	284	632	1355
Mean delay in ms	10,69	15,60	15,98	15,71	19,38	32,30	60,25

Table 1: Results of measurements on the delay caused by asynchronous texture access.

4.2 Qualitative evaluation

For the best possible comparability of the images, a static setup with fixed camera, object and light positions was chosen as seen in figure 5. The object under consideration is vertically above the point of origin of the real studio. The real camera's position, rotation and field of view are tracked by the camera tracking and the parameters are applied to the virtual camera. The position of the virtual video texture was set by measuring the distance from the camera to the real object. The position and orientation of the virtual light was adjusted to the measured values of the real spotlight. Light measurement with a luxmeter from the perspective of the test object allows a replication of the light intensity in the graphics engine. The illuminance in lux at a certain distance, here 494 lx, can be specified in the virtual scene. To determine the strength of the ambient light component, the illuminance on the test object at the camera position was measured with the headlight off. The measured value of 180 lx was set as the ambient illuminance of the skybox in the graphics engine.

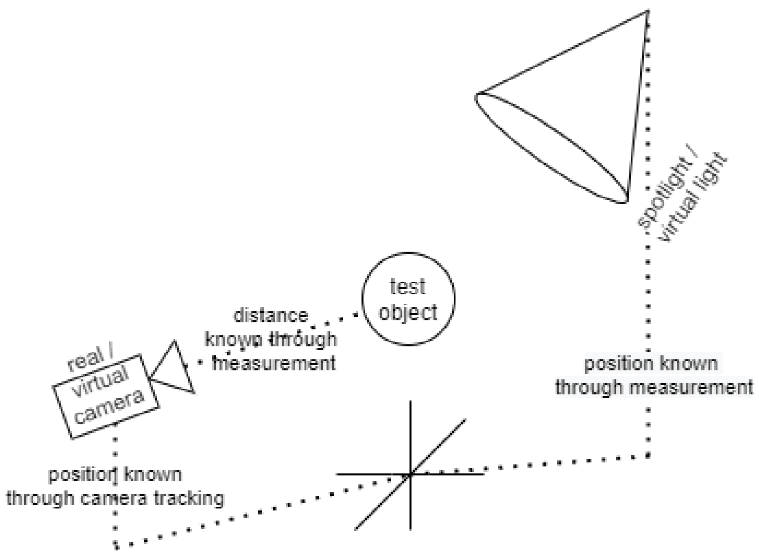

spotlight / virtual light
test object
distance known through measurement
real / virtual camera
position known through measurement
position known through camera tracking

Figure 5: Schematic diagram showing the setup of the evaluation recordings.

The test objects were then each recorded under four lighting scenarios: with a real spotlight, with the relighting system and a virtual light, with a simple three-dimensional plane and a virtual light, and completely without additional lighting. In each case, the images were taken from the graphics engine to prevent any differences in the image that might occur due to different hardware or software components. The captured images are listed in table 2. The metrics listed above were calculated for the images, using the images illuminated by the real spotlight as reference images.

The computed quality metrics do not show any improvement of the relighting system in comparison to simple virtual geometry. With the limited range of results, neither significantly worse nor significantly better values of the relighting system are apparent. It can be emphasized that the relighting system adds self-shadowing and highlights to the input video image. These lighting effects are matched to the contour of the input image and allow the virtual lighting to be mapped into the image. This makes it easier to see the direction, strength and color of the virtual light in the generated image.

Different parameters in the system are adjustable. The strength of the tessellation and the level of detail of the contour data have an influence on the generated geometry. In summary, the newly lit images of the relighting system do not represent realistic illumination of the real scene with virtual lighting. However, the generated three-dimensional geometry makes directional lighting effects more recognizable on the input image, embedding the video

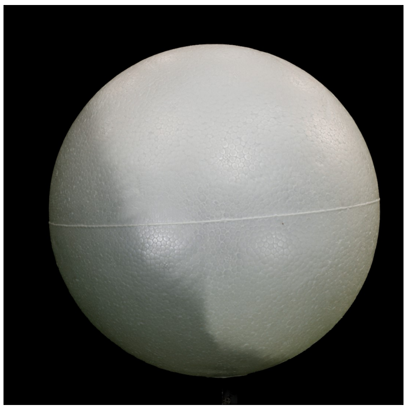
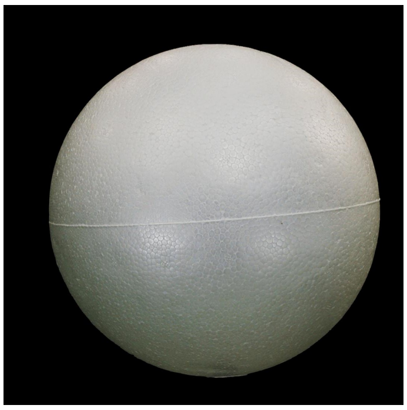
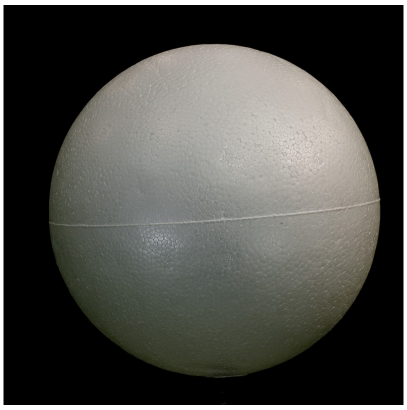
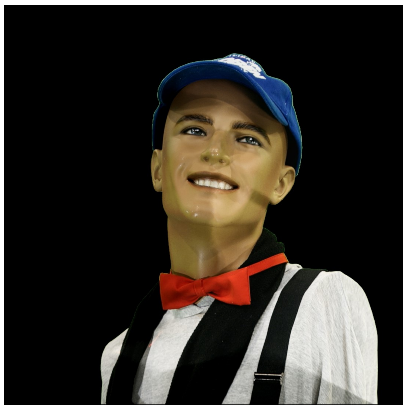
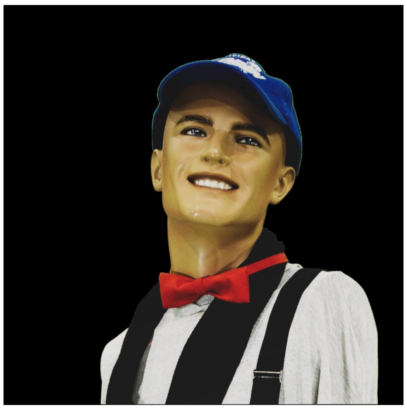
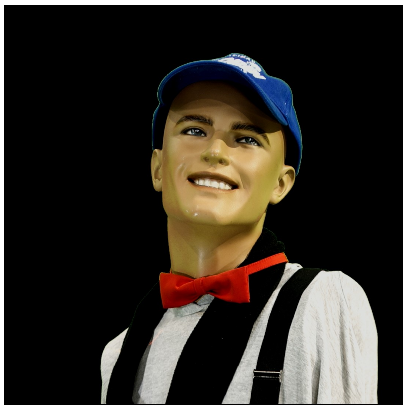
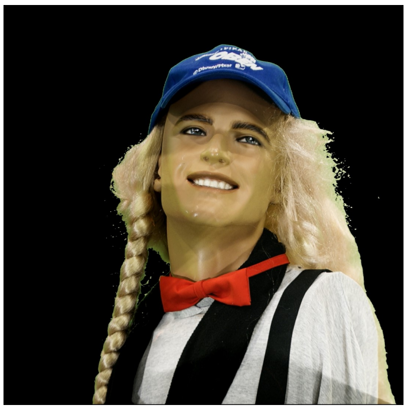
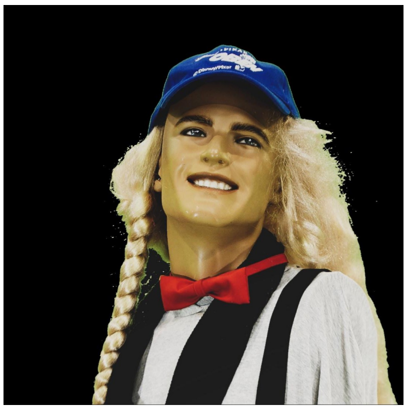
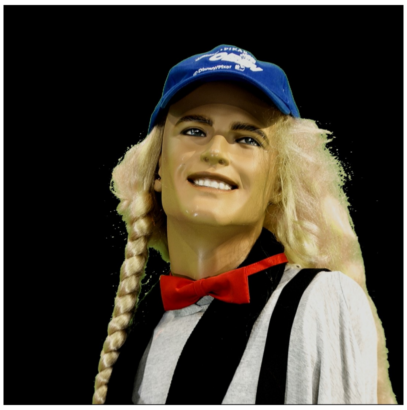
Proposed system	**Simple plane**	**Real lighting (Ref.)**
		
PSNR=20,58 SSIM=0,85 LPIPS=0,13	PSNR=20,00 SSIM=0,90 LPIPS=0,13	
		
PSNR=23,20 SSIM=0,88 LPIPS=0,07	PSNR=24,46 SSIM=14,34 LPIPS=0,09	
		
PSNR=21,12 SSIM=0,86 LPIPS=0,11	PSNR=22,70 SSIM=0,85 LPIPS=0,09	

Table 2: Comparison shots between a simple plane virtually illuminated and the relighting system.

texture in the virtual scene.

5 Conclusion

To map the real geometry, the system could be extended with depth cameras. The image of the depth camera can also be used as a height map, so that the height map would no longer be generated, but would correspond to an actual representation of reality. However, the different projection and position of the depth camera and the video camera must be taken into account.

A congruent matching of an avatar geometry to the person in the video image would allow the use of more detailed geometry and significantly improve the quality of the image results. Using a multi-camera setup to generate the geometry allows for the generation of geometry specific to the person. Continuous matching of the geometry volume to the person's volume could also be used to generate congruent geometry. To solve adaptations of this more complex nature, an implementation using artificial neural networks is likely to be helpful. The system does not consider surface and material properties for illumination; an arbitrary given illumination model is assumed. Estimation of surface properties would be useful to correctly calculate specular lighting properties or to consider volume scattering of the human skin.

A relighting system has been developed that augments an incoming video stream in real time with geometry containing depth information. Correct positioning is given by integrating a markerless motion tracking system. The relighting system allows for a better representation of virtual lighting in the video image, enables surface effects such as self-shadowing and specular highlights, and offers a full integration of the textured geometry into the lighting calculation of a virtual scene. The depth information is mapped by a generic function and has no relation to the actual surface geometry of the real scene.

References

[CVH07] Péter Csákány, Ferenc Vajda, and Adrian Hilton. Recovering refined surface normals for relighting clothing in dynamic scenes. In *4th European Conference on Visual Media Production*, page 21. European Conference on Visual Media Production and Institution of Engineering and Technology, IEEE, 2007.

[FKGK05] Jan-Michael Frahm, Kevin Koeser, Daniel Grest, and Reinhard Koch. Markerless augmented reality with light source estimation for direct illumination. In *CVMP 2005*, pages 211–220. Institution of Electrical Engineers, 2005.

[GDT+19] Kaiwen Guo, Jason Dourgarian, Danhang Tang, Anastasia tkach, Adarsh Kowdle, Emily Cooper, Mingsong Dou, Sean Fanello, Graham Fyffe, Christoph Rhemann, Jonathan Taylor, Peter Lincoln, Paul Debevec, Shahram Izad, Philip

Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, and Rohit Pandey. The relightables: volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics*, 38(6):1–19, 2019.

[Gra06] Oliver Grau. Multi-camera radiometric surface modelling for image-based relighting. In Katrin Franke, Klaus-Robert Müller, Bertram Nickolay, and Ralf Schäfer, editors, *Pattern recognition*, volume 4174 of *Lecture Notes in Computer Science*, pages 667–676. Springer, 2006.

[HNK07] Jens Herder, Christian Neider, and Shinichi Kinuwaki. Hdr-based lighting estimation for virtual studio (tv) environments. In *10th International Conference on Human and Computer*, pages 111–117. 2007.

[HZ10] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *20th International Conference on Pattern Recognition (ICPR)*, pages 2366–2369. Institute of Electrical and Electronics Engineers (IEEE), 2010.

[ISS14] Takuya Ikeda, François de Sorbier, and Hideo Saito. Real time relighting with dynamic light environment using an rgb-d camera. In *International Workshop on Advanced Image Technology*, 2014.

[MDA02] Vincent Masselus, Philip Dutré, and Frederik Anrys. The free-form light stage. In Tom Appolloni, editor, *ACM SIGGRAPH 2002 conference abstracts and applications on - SIGGRAPH '02*, page 262. ACM Press, 2002.

[Ope20] OpenCV team. Opencv 4.5.0 documentation, 2020.

[SWHG10] Peter Stroia-Williams, Adrian Hilton, and Oliver Grau. Reflectance transfer for material editing and relighting. *JVRB - Journal of Virtual Reality and Broadcasting*, (7(2010)), 2010.

[TAL+07] Christian Theobalt, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel. Seeing people in different light - joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):663–674, 2007.

[WBSS04] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 13(4):600–612, 2004.

[ZIE+18] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595. IEEE, 2018.