Free-Viewpoint Video Rendering for Mobile Devices

James Imber[*] [†]
james.imber@imgtec.com

Marco Volino[*]
m.volino@surrey.ac.uk

Jean-Yves Guillemaut[*]
j.guillemaut@surrey.ac.uk

Simon Fenney[†]
simon.fenney@imgtec.com

Adrian Hilton[*]
a.hilton@surrey.ac.uk

ABSTRACT

Free-viewpoint video renderers (FVVR) allow a user to view captured video footage from any position and direction. Despite the obvious appeal of such systems, they have yet to make a major impact on digital entertainment. Current FVVR implementations have been on desktop computers. Media consumption is increasingly through mobile devices, such as smart phones and tablets; adapting FVVR to mobile platforms will open this new form of media up to a wider audience. An efficient, high-quality FVVR, which runs in real time with user interaction on a mobile device, is presented. Performance is comparable to recent desktop implementations. The FVVR supports relighting and integration of relightable free-viewpoint video (FVV) content into computer-generated scenes. A novel approach to relighting FVVR content is presented which does not require prior knowledge of the scene illumination or accurate surface geometry. Surface appearance is separated into a detail component, and a set of materials with properties determining surface colour and specular behaviour. This allows plausible relighting of the dynamic FVV for rendering on mobile devices.

Categories and Subject Descriptors

I.3.1 [**Computer Graphics**]: Hardware Architecture—*graphics processors*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*virtual reality*; I.4.3 [**Image Processing and Computer Vision**]: Enhancement—*filtering*; J.7 [**Computers in Other Systems**]: Consumer Products

General Terms

Algorithms

Keywords

Free-Viewpoint Video Rendering, Image-Based Rendering, Relighting, Mobile Devices

[*]University of Surrey
[†]Imagination Technologies Ltd

1. INTRODUCTION

In recent years, handheld devices such as smartphones and tablets have changed the way in which people use computers. Such devices are becoming increasingly powerful, and are now a major and convenient means of entertainment consumption, with applications ranging from games and video through to augmented reality. Increased media consumption on mobile platforms is driving demand for new forms of video-quality interactive media.

Free viewpoint video rendering (FVVR) allows interactive visualisation of dynamic scenes from multiple view video capture with a quality similar to captured video [27]. Current uses of FVVR and FVVR-like systems tend to be in niche applications, such as in sports analysis. Delivery on mobile platforms will introduce FVV to a wider audience increasing demand for new content and video-based interactive applications. This paper demonstrates that the current generation of SoC (System on a Chip) in smartphones and tablet computers has computational resources that are capable of supporting applications such as FVVR.

The contribution of this paper is two free-viewpoint renderers which run in real time with user interaction on a mobile device. The first adapts the layered representation of FVVR content (LFVVR) introduced by Volino et al. [23] to mobile device rendering (Sections 3.2 and 3.3). The other, which is an extension of the first, takes a novel approach to supporting arbitrary relighting of scenes (RFVVR), requiring no prior knowledge of the lighting conditions during capture (Section 4). FVVRs presented to-date have been implemented on, and intended for, high-end desktop computers [20, 21, 27]. FVVR on mobile devices presents a number of challenges: streaming of multi-view video and mesh data; view-dependent rendering; processing of multi-view video. Producing a high-quality FVVR on a mobile device requires performance enhancements to be made, as well as an adapted rendering pipeline to shift as much of the processing as possible to the offline pre-processing stage. At runtime, the capabilities of the mobile GPU are utilised to allow for high-quality rendering of real-world content from a user-defined viewpoint.

2. RELATED WORK

2.1 Free-Viewpoint Video Rendering

FVVR grew from the field of image-based rendering (IBR), which aims to solve the problem of generating novel viewpoints of a scene from a limited set of images taken from different positions. Video-based rendering (VBR) is an extension of IBR to a sequence of frames. In such schemes, the cameras must be both calibrated and synchronised - in other words, the positions of the cameras in the scene and their direction and focal length are known, and each frame is captured at the same instant by each camera [20, 13].

Early attempts tend to use large camera arrays and narrow baselines, and use interpolation or warping to obtain novel viewpoints. Chen and Williams [6] pioneered one of the first image-based rendering systems, interpolating novel views between closely spaced cameras. To achieve video-based rendering of dynamic scenes, Kanade et al. [14] introduced the Virtualized Reality$^{\mathrm{TM}}$ system using a dome of 51 cameras. Buehler et al. [3] introduced the unstructured lumigraph, which allows view reconstruction between arbitrary arrangements of cameras. Recent image interpolation methods have achieved photorealistic results [27, 16], although the best results often take into account geometry to model occlusions. The method of Lipski et al. [16] allows free user placement of the camera in time and space, and gives highly realistic results for kinds of content that are hard to model using a geometric representation, such as flames.

VBR algorithms have increasingly taken advantage of the computational power and flexibility offered by programmable GPU pipelines. Image-based reconstruction [12, 13] of scene geometry from sparse (calibrated) camera sets enables an alternative approach than that of image interpolation, in which the scene geometry can be reconstructed and projectively textured from the captured images [9, 4, 21, 10, 20]. This produces a representation of the scene which can be viewed from any angle using a standard rasterisation pipeline. Making use of mature graphics rasterisation technology allows real-time interactive performance to be achieved. Such FVVR techniques typically reconstruct a rough 3D mesh of the scene with a low polygon count (refered to hereafter as the *geometric proxy*) and projectively texture it from the original images.

View-dependent projective texture mapping and blending was introduced by Debevec et al. [9]. Projective texture mappling is used to texture a 3D proxy from 2D images to achieve a high degree of realism. Projective texture mapping was first applied in the context of FVVR by Carranza et al. [4] using a visibility test to handle occlusions. More recently, optical flow guided image warping has been applied to deal with projective misalignments between images [10]. Volino et al. [23] introduces a Layered FVVR representation which greatly improves the efficiency of real-time projective texturing by shifting the texture projection itself to an offline preprocessing stage. A minimum of blending between textures is then performed during real-time rendering.

2.2 Relighting

Textures extracted from captured video already have implicit lighting information from the capture setup 'baked-in'. Seamless combination of real-world capture with computer generated content requires relighting to match the illumination of FVVR content with the scene. Many existing techniques that allow accurate relighting of scenes require an active lighting arrangement [8, 17, 5] to allow extraction of the material properties of the object to be relit. Relighting scenes with a single initial uncontrolled lighting arrangement is considerably more challenging.

Material properties must also be taken into account. For example, a polished metal surface will have strong specular highlights, the apparent positions of which will change with changing camera position. This poses problems in any reconstruction techniques using feature extraction as a cue; but even assuming successful geometric reconstruction of the surface, the material properties still remain to be estimated [17]. Surface specularity is a primary contributor to change in appearance with viewpoint. Other properties that cause change in appearance include anisotropic lighting properties, such as with hair and brushed metal, and irridescence.

Many schemes rely on prior knowledge of scene geometry and lighting to deduce material properties to allow accurate scene relighting. Theobalt et al. [21] suggest using a high-resolution set of extracted surface normals to correctly reproduce the scene under arbitrary lighting conditions. They use calibrated lighting to make the problem of normal map extraction more tractable, and require a coarse geometric proxy to initialise the model-fitting step and act as a reference during normal refinement. Normals are fitted, and BRDF properties estimated, in an iterative joint optimisation method. Similarly, Csakany and Hilton [7] use an environmental light map and require accurate scene geometry in their single-viewpoint facial relighting scheme, in order to achieve their results.

A related problem in image-based reconstruction is the estimation of shape given scene lighting [26, 1], which can be applied to refining geometric proxies [25]. One way to ensure correct relighting is to faithfully reconstruct surface geometry to a high level of detail. A robust method that works under general lighting conditions is that of Wu et al. [25]. Estimating a low-frequency representation of the light map, and assuming Lambertian surface reflectance allows accurate reconstruction of high-frequency details.

3. FVVR FOR MOBILE DEVICES

3.1 FVVR Pipeline

Starck et al. [20] developed an open-source FVVR, on which the contributions of this paper are based. Figure 1a presents the rendering pipeline of Starck et al., which is representative of many PC-based implementations. This section gives a brief overview of their FVVR.

An initial coarse approximation of the scene is reconstructed as the visual-hull [15] given by the intersection of the multi-view image silhouettes. This is then refined by stereo matching using a volumetric graph-cut with silhouette rim constraints [19]. Reconstruction is performed independently at each video time frame resulting in an unstructured mesh sequence with a different number of vertices and mesh connectivity at each frame. The unstructured mesh sequences allows free-viewpoint video rendering, but is inefficient for storage and rendering. Enforcing temporal coherence between frames ensures that only vertex positions change with time [2]. The temporally aligned mesh sequence provides the basis for efficient representation and UV texture mapping of the captured perfomance. Each frame is then textured depending on the corresponding video frames, so that changes in appearance in the captured video are preserved in the free-viewpoint renderer.

Each render cycle in the online rendering stage starts by

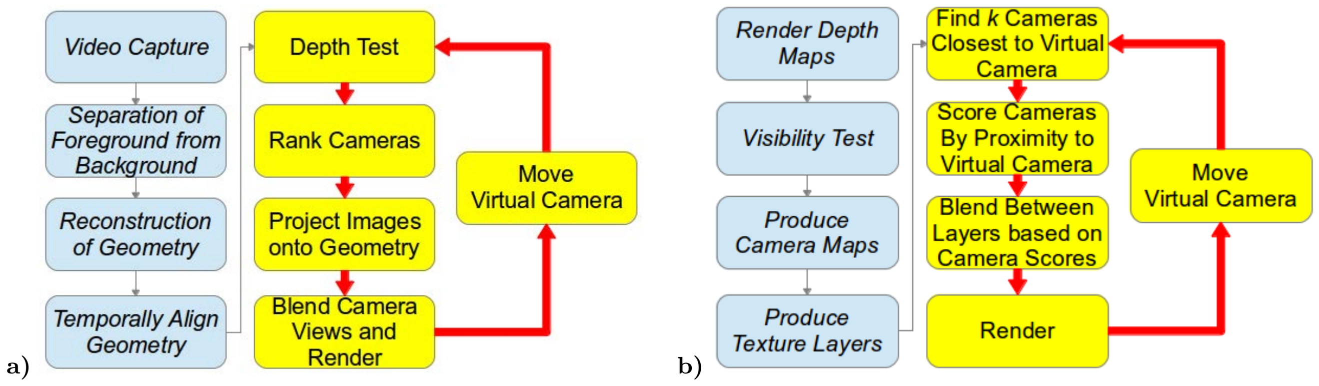
a)
Video Capture
Separation of Foreground from Background
Reconstruction of Geometry
Temporally Align Geometry
Depth Test
Rank Cameras
Project Images onto Geometry
Blend Camera Views and Render
Move Virtual Camera
b)
Render Depth Maps
Visibility Test
Produce Camera Maps
Produce Texture Layers
Find *k* Cameras Closest to Virtual Camera
Score Cameras By Proximity to Virtual Camera
Blend Between Layers based on Camera Scores
Render
Move Virtual Camera

Figure 1: (a) The original open-source FVVR pipeline of Starck et al. [20] (b) The pipeline used for rendering on mobile devices, assuming availability of geometry. The stages in italics are done offline, and the others are done at runtime. The thick arrows indicate the render loop.

performing a depth render from the point of view of each camera. This allows the visibility of a given point on the surface of the proxy to be found by a depth test. Next, for each point on the surface of the proxy, the cameras are scored according to their suitability for use for projective texturing according to three measures: the visibility of the point (a binary mask); the difference in angle between the novel and original viewpoint; and the directness of view of the surface (the closer to the face normal, the better). Finally, the projected textures are blended together, with camera weighting derived from visibility, directness of view and proximity to the novel view. For reasons of performance, blending is carried out between the k cameras closest to the novel viewpoint. In the capture system used by Starck et al., eight evenly spaced cameras were used at intervals of 45 degrees for full angular coverage of the scene. In this case, $k = 3$ is a good choice.

Textures projected from adjacent camera views will not match perfectly. Colour balance often differs slightly between cameras, and there will be inaccuracies in projection resulting in camera boundaries not matching up correctly. View-dependent texture mapping has been introduced to overcome this problem by combining overlapping projected camera views according to the surface orientation:

$$\boldsymbol{R}(\underline{x}) = \frac{1}{\alpha}\sum_{c\in C}\boldsymbol{I}_c(\underline{x})v_c\,(\hat{\underline{N}}\cdot\hat{\underline{D}}_c)\exp\left(\frac{\text{angle}(c,n)^2}{\gamma^2}\right) \qquad (1)$$

Equation 1 shows how the blending between views in a free-viewpoint renderer may be achieved. $\boldsymbol{R}(\underline{x})$ is the rendered output colour of the pixel at screen co-ordinates $\underline{x}$. $\boldsymbol{I}_c$ is the reference image for camera c projected onto the proxy and viewed from the novel camera. C is the set of k physical cameras closest to the novel viewpoint. v_c is the binary visibility mask for camera c, which can take values of either 0 or 1. $\hat{\underline{N}}$ is the surface normal and $\hat{\underline{D}}_c$ is the direction from the rendered point to the camera, so that $\hat{\underline{N}}\cdot\hat{\underline{D}}_c$ is a measure of the directness of view of the surface the camera has. angle(c, n) gives the angle between camera c and the novel viewpoint n. The closer to the physical camera is to n, the greater is its contribution to $\boldsymbol{R}$. γ is chosen depending on the angles between the physical cameras. α is a normalisation factor, so that the weights applied to $\boldsymbol{I}_c$ add up to one.

3.2 Layered FVVR

In this work we build on the layered FVVR implementation recently introduced by Volino et al. [23]. Figure 2 gives an overview of layered FVVR (LFVVR), and the rendering pipeline is shown in Figure 1b. Instead of direct projective texturing of the proxy from the original images, the projective texturing is done in a preprocessing stage, to produce a stack of texture layers that can be used at runtime to texture the proxy. This significantly reduces the amount of processing done when rendering. The top layer in the stack is a composite of the best camera views; the second is a composite of the second best, and so on to the bottom layer, which is a composite of the worst camera views. Beyond the first λ layers ($\lambda = 4$ in the 8-camera setup used here), the information content is very low.

The preprocessing begins by producing λ camera maps. These are images with $C + 1$ greyscale levels (where C is the number of cameras used in capture), which record which camera should be used to obtain each part of the corresponding texture layer. Black represents points for which no data is available. Cameras are scored for each point in the camera map by $v_c\left(\hat{\underline{N}}(\underline{x})\cdot\hat{\underline{D}}_c(\underline{x})\right)$, where the terms have the same meaning as in Equation 1. The highest scoring camera for each point is stored in the top camera map, through to the lowest scoring camera in the bottom camera map. The texture layers are found once the camera maps are available; for each point on each camera map, the image from that camera is projected, by knowledge of scene geometry, onto the UV plane to form the texture layers.

When rendering, as in conventional FVVR, views from the k closest cameras are blended together. A camera view is reconstructed by searching the camera maps for parts of the corresponding texture layer sampled from that view. The blending is done as in Equation 1, with the exception of the visibility test, which is done in the preprocessing stage. For a circle of eight cameras with 45 degree separations, with angle(c, n) in radians, a value $\gamma = 0.7$ in Equation 1 was found to work well. Table 1 shows how rendering quality improves as λ is increased.

Figure 3 shows how blending between three adjacent camera views (k) with four layers (λ) improves the quality of the render. In the image difference, blue is no difference and red is an L1 distance of about 0.3 or more in normalised RGB

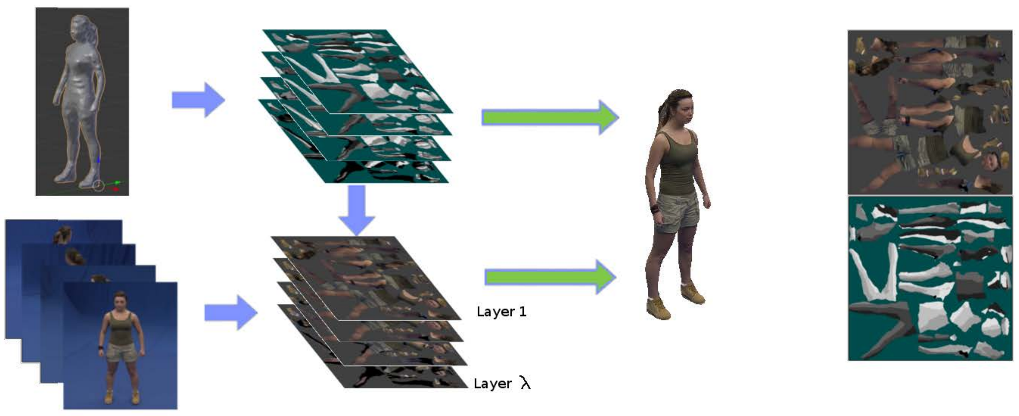

Figure 2: A stack of camera maps is produced for each frame from the geometric proxy. The camera maps are used as a reference when resampling original images to produce the stack of texture layers. Camera maps and texture maps are both used when rendering (green arrows). On the right, a camera map and corresponding texture layer. In the camera map, there is a greyscale level for each of the C cameras.

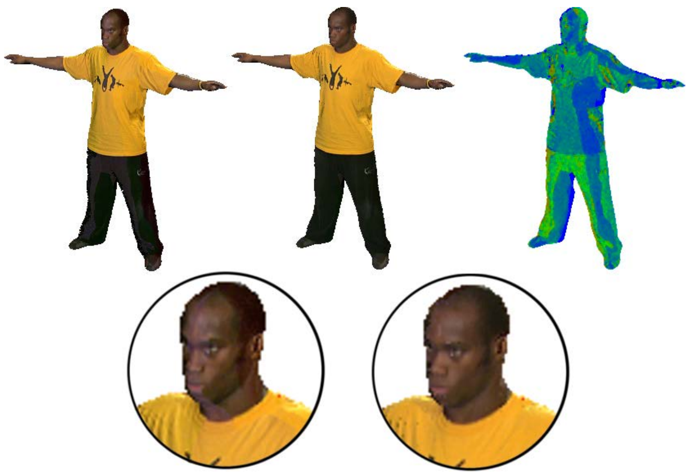

Figure 3: Free-viewpoint rendering using only the best camera for each pixel (top-left) compared to blending between camera views in layered FVVR (top-centre). On the right, the difference between the first two images.

space (R, G and B co-ordinates in range 0 to 1). Close-ups of the two renders show how blending between camera views eliminates issues with colour mismatches. However, blending views from too many physical cameras has the potential to blur the render by introducing ghosting artefacts when the camera calibration or geometry is imperfect.

3.3 Adaptation to Mobile Devices

A mobile platform has considerably reduced capabilites compared to a desktop computer, for which FVVRs to-date have been developed. This leads to some particular challenges in producing a viable renderer.

One major difficulty relates to the amount of data required to correctly render a time-changing FVV sequence. This is an issue even on desktop computers, where bandwidth between secondary storage and main memory can become a performance bottleneck [27]. Table 2 shows the storage requirements of different representations of texture data for FVVR. Every second of uncompressed HD footage (1920 x 1080 px resolution) shot simultaneously with eight cameras at 30 fps requires 1.39 GB of space. Considering that many mobile devices have less than 64 GB of storage, this is an unacceptably high storage requirement. There is also an additional problem of streaming data into main memory, which is generally around 1 GB in size. There is therefore a strong motivation to reduce the storage requirements of the data for FVVR, both by compression and by judicious discarding of redundant data.

A major advantage of LFVVR, and the reason it is suited to application on mobile devices, is that it discards redundant data when compiling the layer stack. However, in LFVVR as in conventional FVVR, each frame requires a separate set of textures to replicate changes in the appearance of the model with time. These include, but are not limited to, surface lighting, folds in clothing and actor facial expression. Therefore, high-ratio compression and subsequent high-speed decompression is required. This can be achieved by working within the strengths of the platform.

Table 1: PSNR (dB) of uncompressed Layered FVVR with different numbers of layers in Roxanne sequence

Number of Layers	Avg	Max	Min
1 layer	20.25	21.40	18.36
2 layers	20.77	22.03	18.78
3 layers	21.26	22.71	19.05
4 layers	21.32	22.77	19.09

Table 2: Storage Requirements for 10 seconds of FVVR Playback

Scheme	Uncompressed	PVRTCI (2bpp)
Initial HD Video	13.9 GB	1186.5 MB
Layered FVVR	4800.1 MB	300.1 MB
Relightable FVVR	2304.4 MB	192.4 MB

Many graphics units have the ability to directly decompress textures in hardware. For example, the PowerVR graphics unit used to produce the results in this paper is able to decompress PVRTC compressed textures [11] in hardware at negligable cost to performance. The PVRTCI 2bpp compression standard has a compression ratio of 16:1, and the quality is sufficiently good for high-quality mobile FVVR.

The alpha channel of the layers is used in each frame to hold the corresponding camera map. This is an optimisation which greatly improves performance. It reduces traffic between GPU and main memory, requires only a single texture lookup in the fragment shader, and halves the storage space that would otherwise be required. Many texture compression standards, like PVRTC, allocate fewer bits to coding the alpha channel than the R, G and B channels, but this is not a concern with the camera map, where there is a very limited number of greyscale levels, equal to the number of cameras used in capture. Real time performance was achieved by optimising the shaders used in the FVVR to keep costly operations such as texture lookups to a minimum.

4. RELIGHTABLE MOBILE FVVR

4.1 FVVR Detail Mapping

We propose a novel scheme for relightable FVVR (RFVVR), which does not require any knowledge of scene lighting. This is in contrast to methods which fit a physical lighting model given calibrated light sources. This scheme is based on the assumption that the human eye is insensitive to detail when estimating scene lighting, and a scene can be plausibly relit without correctly relighting the finer detail. It is also relatively fast in preprocessing, requiring only a segmentation and high-pass filtering stage. Figure 4 shows a FVV frame relit from various directions using this *detail mapping* technique.

The texture map comprising the top layer from LFVVR (that with the most direct camera views of each face) is segmented by material in each frame. The texure map of the first frame of the sequence is hand-segmented into user-specified constituent materials of consistent material properties. This segmentation is automatically propagated to subsequent frames; the initial segmentation cannot be used for all frames, since the material boundaries drift over the surface of the mesh with time. These segmented versions of the top layer are termed *material maps.* The segment propagation is achieved using our own tool, which is an adaptation of the method of Wang and Collomosse [24].

Whilst directly rendering the material map at this stage supports relighting, no detail within the materials is retained. For each region the surface albedo and specular reflectance properties are estimated. The detail is extracted from the top LFVVR layer for that frame using a high-pass image filter: each channel of the RGB image is filtered to remove the low frequency components, which include large regions of shadow. The filtered channels are then recombined to form a full RGB detail map. This is then directly added to the relit materials, as shown in Figure 5.

The results in this paper were rendered using the Phong lighting model [22]. Only a single light source was used for testing, although it can easily be extended to handle multiple light sources.

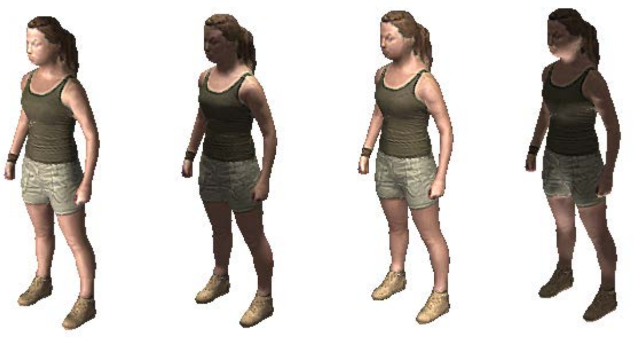
Figure 4: Relighting from various directions by material-based rendering with FVVR detail mapping.

$$\underline{I}_p = (\underline{k}_d + \rho.\underline{k}_D)(i_a + i_d(\hat{\underline{L}} \cdot \hat{\underline{N}})) + i_s(\hat{\underline{R}} \cdot \hat{\underline{V}})^\gamma \qquad (2)$$

The parameter ρ governs the contribution of the detail map $\underline{k}_D$. Both this and the material-specific colour albedo $\underline{k}_d$ are derived from the initial texture map. The coarse geometric proxy gives the surface normals $\hat{\underline{N}}$. i_a and i_d are, respectively, the ambient and diffuse lighting strengths of the scene in which the character is rendered, whereas i_s is determined from the texture map to match the specular properties of the original materials. A value of 2 is used for the specular exponent γ thorughout this work, since it is a reasonable estimate for materials such as hair and skin. Fixing γ to a constant value for all surfaces simplifies the estimation of the specular reflectance component i_s, giving a tractable solution. In practice the fixed exponent has not been found to be a significant limitation in the resulting rendering quality.

4.2 Frequency Domain Effects

A filter suitable for this task must completely reject large regions of shadow in the image, whilst retaining the smaller shadows and lit regions that comprise the surface detail. The cut-off point should be high enough that the material base colours are rejected, and low enough that relevant detail is not lost. A highly selective filter was required to achieve this. However, it also introduced spurious ringing artefacts into the result. Artefacts at seams in the UV map were eliminated by expanding the boundaries of the textured regions of the top layer by a few pixels before applying the high-pass filter. This prevents ringing artefacts due to filtering from appearing on seams in the render. The layer background colour was chosen to minimise discontinuities at the boundaries with the textured regions, so that discontinuities between foreground and background were minimised. This helps reduce the appearance of ringing artefacts.

5. RESULTS

5.1 Mobile Platform used for Testing

The platform used to obtain the results presented is a Pandaboard running Ubuntu 12.04. The Pandaboard has a Texas Instruments OMAP4 SoC, which is broadly representative of the capabilities of smartphones and tablets currently on the market. The OMAP4 contains an Imagination Technologies PowerVR SGX540 graphics core.

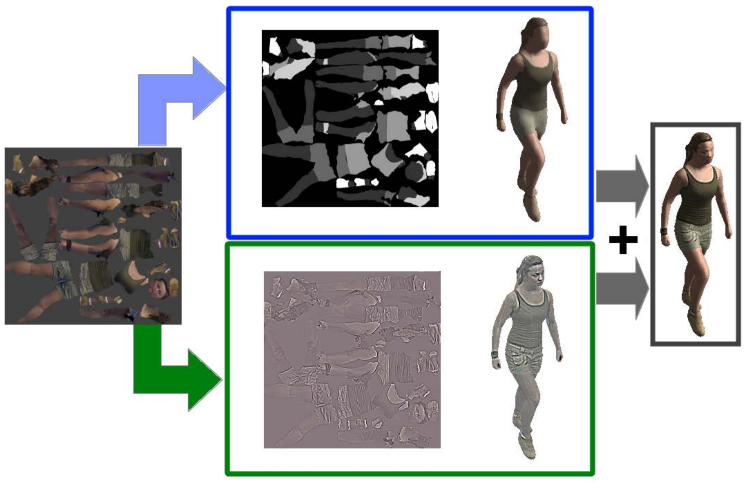

Figure 5: Processing the top UV layer (left) to obtain a version segmented by materials (top) and a detail map (bottom). These are combined to give the final render (right).

5.2 Evaluation of FVVR for Mobile Devices

In a conventional FVVR, the original images can be directly blended to texture the proxy. LFVVR uses an intermediate representation of the original images, in the form of a layer stack, to reduce the computational load at runtime. When rendering from one of the original cameras, a conventional FVVR uses only the image from that camera, since this will give the best results. In an LFVVR scheme, such a direct projection is possible, although aliasing of faces that are viewed from a glancing angle can become an issue, since there will not be a sufficiently good representation of these faces stored in the texture. Such faces appear smeared in the render and suffer from aliasing in the texture layers. For this reason, the best results are achieved by incorporating a small contribution from adjacent cameras, as in Equation 1. In addition, there may be some general loss of quality over direct use of the original images due to resampling to an intermediate texture map representation. The additional resampling stage, and the small contribution from neighbouring cameras, are the reasons for the differences in Figure 6.

Peak Signal to Noise Ratio (PSNR) was used as a concrete measure of FVVR quality. Here, P is the set of image pixels, $\boldsymbol{I}$ is the original image, and $\boldsymbol{F}$ is the FVVR render. $\max(\boldsymbol{I})$ is the highest value that a pixel can take.

$$\text{MSE} = \frac{1}{|P|} \sum_{\underline{x} \in P} \left(||\boldsymbol{I}(\underline{x}) - \boldsymbol{F}(\underline{x})||_{L_2} \right)^2 \tag{3}$$

$$\text{PSNR} = 10 \log_{10} \left(\frac{\max(\boldsymbol{I})^2}{\text{MSE}} \right) \tag{4}$$

The entire sequence is rendered from the point of view of one of the original cameras. The PSNR for each individual frame is found by comparison with the original footage. Table 3 shows the PSNR for each of the sequences tested.

The use of main memory should be maximised to avoid thrashing secondary storage. 1GB of memory is available on the OMAP4, although some of this is taken up by the operating system and utility programs. The storage requirements of a sequence ten seconds long, played back at 30 fps, is given in Table 2. When running on the OMAP4-based test board at a resolution of 800×600 pixels, a refresh rate averaging 33.5 fps was achieved. This is a reasonable framerate, which even compares with desktop renderers [21, 20].

Table 3: PSNR (dB) of Layered FVVR compared to Original Images. The first 50 frames of the 'freestyle' JP sequence from the SurfCap dataset were used for evaluation.

Sequence	Avg	Max	Min	Length
Roxanne	21.32	22.77	19.09	30 frames
Roxanne Compressed	20.61	21.83	18.75	30 frames
JP	15.34	17.07	13.78	50 frames
JP Compressed	15.26	16.94	13.69	50 frames

5.3 Evaluation of Relightable FVVR

Since the RFVVR scheme developed here only uses two textures per frame, it achieves a better performance than LFVVR, both in terms of frames per second and in terms of memory usage. The detail map must be mipmapped if reasonable renders are to be achieved, and this increases the memory overhead for those textures by 56%.

Illumination of FVV with scene lighting allows seamless integration into computer-generated scenes, as in Figure 7. In this scene, which includes skybox and shadow mapping, the refresh rate averages 36 fps at 800×600 pixels. This remains consistent, even when the user changes the viewpoint by large amounts between frames. The lack of on-the-fly camera scoring reduces the per-frame computation time.

Results from detail maps are not physically accurate in the same way as results using a normal map derived from model fitting, since the details will retain the original scene lighting. Detail mapping essentially uses the fact that the human vision system is insensitive to the lighting of surface details - the correct lighting of the low frequency components of the model is a far stronger visual cue than the overlaid high-frequency components [18]. One region where this does not hold is the face: human vision is much more sensitive to facial details than details in other parts of the model. This is particularly noticeable under lighting conditions signifi-

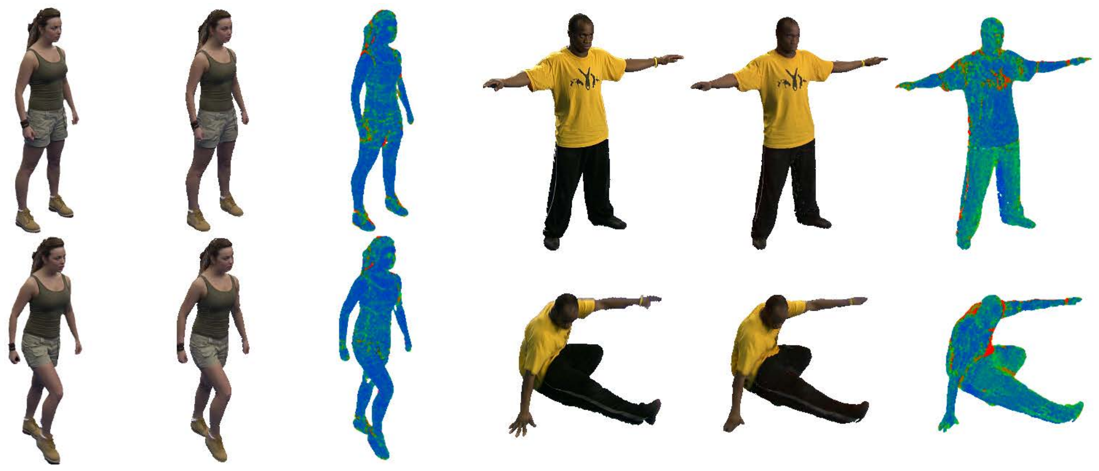
Figure 6: Original images (left) compared to the output of the layered FVVR from the same viewpoint (centre). On the right, the difference between the two images.

cantly different to the lighting in the original images.

The materials do not need to have constant colour albedo, as long as any regions of 'atypical' colour detail are relatively small. For example, the shoes in the sequence used for testing in this paper have small white regions, which are reproduced faithfully in the detail-mapped renders. The cut-off of the high-pass filter is a good indicator of the size of regions that can be preserved in this way. If the size of the region is lower than the cut-off, then it is best to model it with a material of its own.

The proposed split into a material map and detail map is an efficient solution to relightable FVVR, which moreover is convenient for implementation in existing applications. In addition to enabling relightable FVVR on mobile devices, it offers significant performance benefits to desktop renderers.

6. CONCLUSION

An implementation of FVVR for mobile devices is presented based on representing the captured image appearance as layered texture maps (LFVVR). This allows view-dependent rendering at interactive rates on a mobile device. Hardware-supported texture compression and use of the texture alpha channel to store the camera maps made this possible. This approach is extended to enable relighting (RFVVR) through estimation of material properties and shading. The resulting relit character model gives a plausible rendering of the character body and clothing. Visible artefacts occur on the face for extreme light positions due to the reconstructed geometry having insufficient detail.

The relighting scheme has been demonstrated to reproduce the appearance of studio-captured scenes as they would appear under arbitrary lighting conditions without the need for calibrated light sources. In addition, the preprocessing step is efficient since it is only necessary to segment and high-pass filter the original projected textures, before combining them in an otherwise conventional rasterisation pipeline. Even under novel lighting conditions in computer-generated scenes, the results are plausible.

The proposed free-viewpoint video render for mobile devices allows captured multi-view video content to be integrated with computer generated scenes. Real-time interactive performance is achieved on a standard mobile platform.

7. ACKNOWLEDGMENTS

The authors would like to thank Imagination Technologies for sponsoring this research, providing the test device and for sharing their expertise in mobile device graphics programming.

8. REFERENCES

[1] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric Stereo with General, Unknown Lighting. *International Journal of Computer Vision*, 72(3):239–257, June 2006.

[2] C. Budd, P. Huang, M. Klaudiny, and A. Hilton. Global Non-rigid Alignment of Surface Sequences. *International Journal of Computer Vision*, 102(1-3):256–270, Aug. 2012.

[3] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, pages 425–432, 2001.

[4] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH 2003*, pages 569–577, 2003.

[5] C. Chabert, P. Einarsson, and A. Jones. Relighting human locomotion with flowed reflectance fields. *ACM SIGGRAPH 2006 Sketches*, 2006.

[6] S. E. Chen and L. Williams. View interpolation for image synthesis. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93*, pages 279–288, 1993.

[7] P. Csakany and A. Hilton. Relighting of Facial Images. *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 55–60, 2006.

[8] P. Debevec, T. Hawkins, and C. Tchou. Acquiring the reflectance field of a human face. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 145–156, 2000.

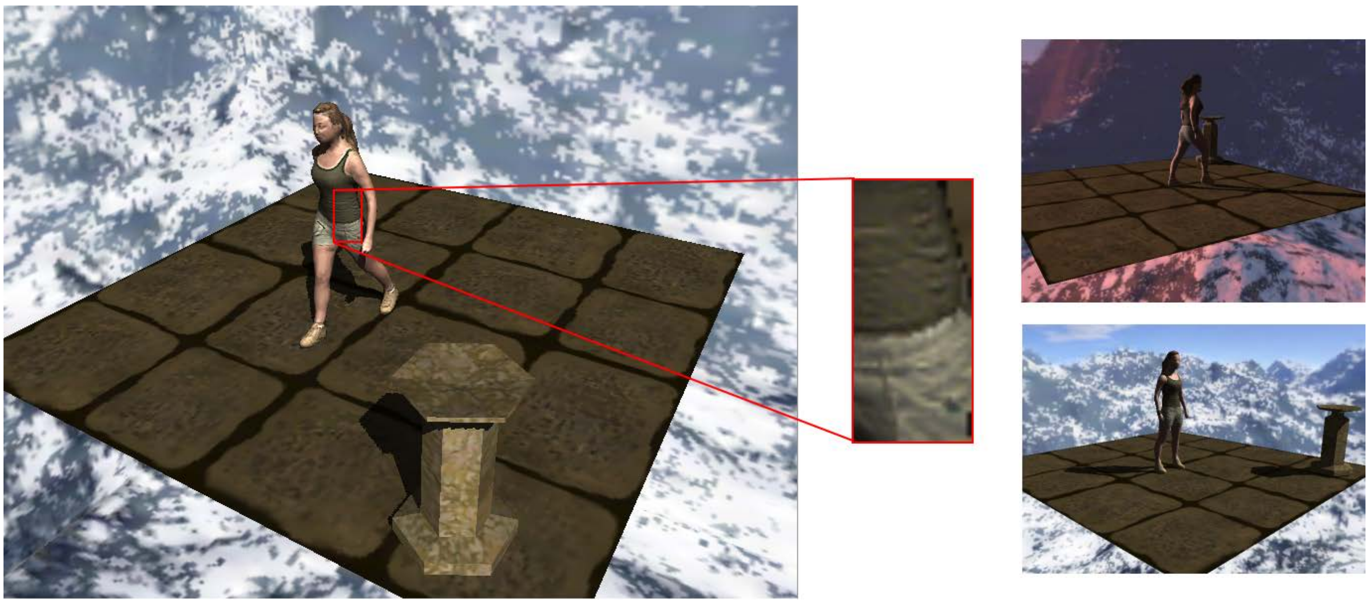

Figure 7: Integration of relightable FVVR into a computer-generated scene, in which the FVVR content is relit to match the scene. Folds in clothing, and details in the face and hair, are preserved.

[9] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 1–10, 1996.

[10] M. Eisemann and B. D. Decker. Floating textures. *Computer Graphics Forum*, 2008.

[11] S. Fenney. Texture compression using low-frequency signal modulation. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 84–91, 2003.

[12] Y. Furukawa and J. Ponce. Carved Visual Hulls for Image-Based Modeling. *International Journal of Computer Vision*, 81(1):53–67, Mar. 2008.

[13] J.-Y. Guillemaut and A. Hilton. Joint Multi-Layer Segmentation and Reconstruction for Free-Viewpoint Video Applications. *International Journal of Computer Vision*, 93(1):73–100, Dec. 2010.

[14] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: constructing virtual worlds from real scenes. *IEEE Multimedia*, 4(1):34–47, 1997.

[15] A. Laurentini. The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1994.

[16] C. Lipski, C. Linz, and K. Berger. Virtual Video Camera: Image-Based Viewpoint Navigation Through Space and Time. *Computer Graphics Forum*, 29(8):1–11, 2010.

[17] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. Image-based 3D photography using opacity hulls. *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH '02*, page 427, 2002.

[18] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. p.489, Prentice-Hall, 1975.

[19] J. Starck and A. Hilton. Model-based human shape reconstruction from multiple views. *Computer Vision and Image Understanding*, (January), 2008.

[20] J. Starck, J. Kilner, and A. Hilton. A Free-Viewpoint Video Renderer. *Journal of Graphics, GPU, and Game Tools*, 14(3):57–72, Jan. 2009.

[21] C. Theobalt, N. Ahmed, G. Ziegler, and H.-P. Seidel. High-quality reconstruction from multiview video streams. *Signal Processing Magazine, IEEE*, 24(November 2007):45–57, 2007.

[22] B. Tuong-Phong. Illumination for computer-generated images. *Communications of the ACM*, 18(6), 1975.

[23] M. Volino, J. Pansiot, O. Grau, and A. Hilton. Layered View-Dependent Texture Maps. *9th European Conference on Visual Media Production (CVMP)*, 2012.

[24] T. Wang and J. Collomosse. Probabilistic Motion Diffusion of Labeling Priors for Coherent Video Segmentation. *Multimedia, IEEE Transactions on*, (November):1–12, 2012.

[25] C. Wu, K. Varanasi, Y. Liu, H.-P. Seidel, and C. Theobalt. Shading-based dynamic shape refinement from multi-view video under general illumination. *2011 International Conference on Computer Vision*, pages 1108–1115, Nov. 2011.

[26] C. Wu, B. Wilburn, Y. Matsushita, and C. Theobalt. High-quality shape from multi-view stereo and shading under general illumination. *Cvpr 2011*, pages 969–976, June 2011.

[27] C. Zitnick, S. Kang, and M. Uyttendaele. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics*, 1(212):600–608, 2004.