Journal of Visual Communication and Image Representation 53 (2018) 192–201

Contents lists available at ScienceDirect

Journal of Visual Communication and Image Representation

journal homepage: www.elsevier.com/locate/jvci

Affordable content creation for free-viewpoint video and VR/AR applications

R. Pagés[1], K. Amplianitis*,[1], D. Monaghan[1], J. Ondřej[1], A. Smolić

V-SENSE, School of Computer Science and Statistics, Trinity College Dublin, Ireland

ARTICLE INFO

Keywords:
Free-viewpoint video
3D reconstruction
Texturing
View synthesis
Augmented reality
Virtual reality

ABSTRACT

We present a scalable pipeline for Free-Viewpoint Video (FVV) content creation, considering also visualisation in Augmented Reality (AR) and Virtual Reality (VR). We support a range of scenarios where there may be a limited number of handheld consumer cameras, but also demonstrate how our method can be applied in professional multi-camera setups. Our novel pipeline extends many state-of-the-art techniques (such as structure-from-motion, shape-from-silhouette and multi-view stereo) and incorporates bio-mechanical constraints through 3D skeletal information as well as efficient camera pose estimation algorithms. We introduce multi-source shape-from-silhouette (MS-SfS) combined with fusion of different geometry data as crucial components for accurate reconstruction in sparse camera settings. Our approach is highly flexible and our results indicate suitability either for affordable content creation for VR/AR or for interactive FVV visualisation where a user can choose an arbitrary viewpoint or sweep between known views using view synthesis.

1. Introduction

In the classical experience of movies or TV, a director carefully chooses a particular viewing angle and sequence of scenes in order to convey a story. Emerging types of media support the ability to individually control the viewer's perspective, which can provide a more immersive and more personalised viewing experience. When watching a video of a football game or concert or other events, live or recorded, empowering the viewer to become the director can bring them closer to the action and give them a sense of immersion. To achieve this, 3D geometry information of the scene to be displayed is necessary. In a lab or a professional studio environment, this can be obtained by massively instrumenting a room with large amounts of specialised camera equipment. Even though this can produce high quality 3D reconstruction results, the downsides are that it is difficult or unfeasible to obtain similar quality for larger scale and outdoor scenes and thus it is limited only to professional production setups.

Furthermore, applications on virtual and augmented reality (VR/AR) have received tremendous attention over the last years, while technological advancement is increasing with time and making stationary systems and mobile devices much more powerful for such applications. With prototypes and consumer devices becoming widely available, there is an ever increasing demand for compelling content. Affordable 3D content creation tools, in contrast to high end solutions we mostly see today, will be necessary to satisfy the needs of these emerging consumer mass markets.

Nowadays, most available VR/AR content is synthetic (graphics), created by artists and designers. Available real-world content (live action) is mostly 360-degree video, captured using omnidirectional camera rigs. However, the latter can only provide a 3 degrees of freedom (3DoF) immersive experience, as only rotation is supported for the viewer. The development of new Free-Viewpoint Video (FVV) techniques, such as the one presented in this paper, can overcome these limitations by allowing the user to freely navigate within a recorded scene and select any viewing point he wished to observe at any moment (6 degrees of freedom, 6DoF) in time.

FVV is a well known field and has received much attention commercially but also scientifically. We have seen commercial applications in movie productions to show 'bullet-time' visual effects, or in sports broadcast, where companies such as Vizrt[2] or Intel play with a virtual camera to enhance the viewer experience. Nevertheless, content created with these approaches does not support the interactivity required

This paper has been recommended for acceptance by Olivier Le Meur.

* Corresponding author.

E-mail address: kostampl@gmail.com (K. Amplianitis).

[1] These authors contributed equally to this work.

[2] http://vizrt.com/.

https://doi.org/10.1016/j.jvcir.2018.03.012
Received 30 November 2017; Accepted 12 March 2018
Available online 23 March 2018

in VR/AR. Probably, Microsoft (through the work presented in [1]) and 8i[3] have the best FVV content creation systems that could work for these immersive applications. However, both companies use high-end capturing setups populated with dozens of cameras (both RGB and infrared), green screen studios and high performance computing capacities.

Inspired by the work of Ballan et al. [2], the purpose of our work is to bring the creation of interactive FVV content, which can be experienced in VR/AR, closer to the average user and to make it more affordable for (semi-) professional content creators. To this end, the principal contributions of our work can be summarised as follows:

- An end-to-end system to create and process FVV sequences that can be and visualised either in VR/AR or using a view-synthesis mode.
- A lightweight system that produces high quality content from a limited number of commodity cameras by combining state-of-the-art reconstruction techniques with efficient camera pose estimation, 3D skeletal restraint, colour consistency and multi-view stereo restraint techniques.
- Multi-source shape-from-silhouette (MS-SfS) and efficient fusion of different geometry data.

It should be noted that in this work we do not focus on the areas of efficient data compression for transmission and storage.

2. Related work

Following the structure presented in the work by Smolic [3], previously introduced in the context of graphics by Kang et al. [4], current FVV techniques are normally classified as a continuum in between two groups: *image-based*, where the intermediate views between cameras are generated using interpolation or warping of the available images; and *geometry-based* techniques, where 3D geometry of both the dynamic foreground and static background is acquired, allowing the rendering from any other viewpoint. Although extensive literature exists in the field, in this sections we will focus on the most important contributions related to our work.

A very significant early image-based FVV technique is the one proposed by Zitnick et al. [5]. Their system generates multiple depth maps using a multi-camera setup, allowing the user to experience new rendered views in between cameras and in a limited area around them. Another approach is the one proposed by Lipski et al. [6], where an array of standard consumer cameras is used to generate a FVV scene. The 2D matrix of cameras is extended to a 3D by adding a temporal component, creating a 3D grid that is triangulated using Delaunay triangulation, which defines different paths of virtual camera across the sequence. The rendered view is a weighted warp of all these possibilities. The result presents high visual quality but if the number of cameras is limited, the possible navigation range is also very limited.

One of the most intensively studied application areas of image-based FVV is sports broadcasting. There are many related publications, from properly registering the cameras [7], generation of intermediate viewpoints [8], or the correct representation of the players in the camera transitions [9], normally using simple billboards or a more advanced billboard warping technique [10]. Similarly, the work by Ballan et al. [2], another important example of image-based FVV and the one that inspired our project, uses billboard interpolation. Specifically, a scene is captured using handheld commodity cameras, both foreground masks and camera pose are recovered at every frame, and a simple model of the background is reconstructed to improve the visual experience in camera transitions.

On the other end, geometry-based techniques normally focus on acquiring the 3D geometry of a scene as accurately as possible, which can be seen as a 3D reconstruction problem that is extended to the temporal dimension. Space carving or SfS is a well known technique, widely cited in the literature [11]. Naturally 3D reconstruction of humans has been of most interest. Starck et al. [12] for instance utilise SfS with photo consistency. Multi-view stereo (MVS) techniques have also been widely used for reconstruction of 3D models [13] and especially for dynamic 3D scenes [14]. One of the most interesting works using MVS for FVV is the one by Collet et al. [1]. In a studio setup they use both RGB cameras and infrared structured light to generate dense point clouds using MVS. These point clouds are meshed using a silhouette constrained Poisson surface reconstruction [15]. Their results are impressive, but the approach requires an expensive and sophisticated hardware setup in a studio environment, which is prohibitive for many application scenarios. Recently, Mustafa et al. [16] focus on creating a joint multi-view segmentation and reconstruction system that uses temporal coherence between frames. As it is discussed in Section 5 neither SfS or MVS are perfectly suitable for very sparse camera setups, so we take advantage of the best of each of them in our novel MS-SfS and data fusion approach.

In between these two extremes, we can find very interesting techniques that make use of 3D geometry estimation to improve the synthesis of new views. For instance, the work by Lipski et al. [17] uses MVS to generate a point cloud of the scene, helping their image warping system to perform better. This is also the principle of the immersive instant replay by FreeD, but they use a very dense setup of high quality cameras. Our approach can be included in this last group of techniques, as we use 3D reconstruction of the scene to improve the view synthesis. However, we aim for accurate 3D reconstruction of dynamic scenery, which allows users to fully immerse in related VR/AR/MR visualisations, while supporting affordable capture and processing.

3. System overview

Fig. 1 presents an overview of our system, which is divided into three main blocks: scene acquisition and preprocessing, scene reconstruction, and visualisation. The first stage of our system, described in Section 4, is preparing the data for the later stage of the reconstruction. As we are not restricted to professional setups with controlled background and illumination or still cameras, it is necessary to first correct the colour differences between the images. Then, we apply object segmentation to separate the dynamic foreground from the static background and finally, estimate the camera poses for every frame of each sequence.

The second and core part of the process is the scene reconstruction, described in Section 5. To overcome the complexity of producing dynamic 3D models in very sparse camera setups, we introduce a new method that fuses different sources of data. On the one hand, we estimate a dense point cloud using an enhanced MVS approach. On the other hand, we estimate the volume of the dynamic subject through a novel, multi-source shape-from-silhouette (MS-SfS) approach, which uses silhouettes, colour consistency and the 3D skeleton (pose) information of the subject. The resulting models are fused generating a 3D mesh that contains details of the point cloud and the completeness of the volume.

Finally, Section 6 presents a qualitative evaluation of our approach and different rendering results both in a VR/AR environment but also in FVV view-synthesis applications.

4. Scene acquisition and preprocessing

Our system considers a static 3D background and human performers in the scene, normally captured with a sparse setup of static or moving consumer handheld cameras, each of them with its own clock (timestamp) and frame rate. For preserving consistency between the sequences in the dynamic reconstruction stage, it is essential to convert

[3] http://8i.com/.

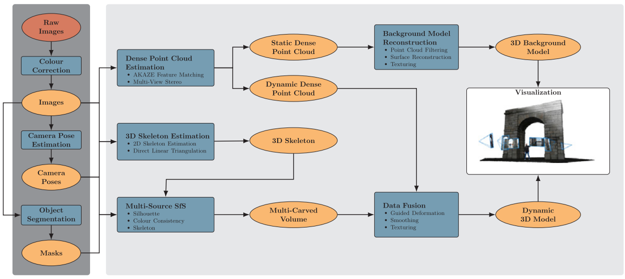


Fig. 1. Proposed system.

such input to the same frame rate and to perform temporal synchronisation. Different temporal synchronisation techniques are widely available and any can be used with our system. For example, in one of our outdoor sequences we used a simple Ultra bright LED flash to synchronise the cameras. Moreover, due to the desired flexibility in our scenario, cameras may have varying resolution and exposure. We therefore include a colour correction stage to minimise such inconsistencies. The cameras' 3D poses are computed relatively to the 3D coordinate system of the background scene. As a final preprocessing step, the object in question to be reconstructed is segmented in all sequences using the latest state-of-the-art algorithms.

4.1. Colour correction

Due to the casual nature of our setup, it is necessary to face the possible use of different camera sensors, with different resolutions and even white balance. This results in images with very different colour tone, that might introduce errors in the colour consistency check or artifacts in the texture mapping stage. To solve this, we apply the colour transfer system by Grogan et al. [18,19], which transforms the colour distribution of a target image to match that of a palette image. This colour transfer method uses Gaussian Mixture Models to represent the colour distribution of the target and palette image, and these are robustly registered to estimate a non-linear parametric colour transfer function. To correct the colour differences between the cameras using this technique, an image frame is selected from both the target and palette image sequences, and a colour transfer function is estimated. As this transfer function is parametric, it is especially good for video sequences: it can then be applied to all frames in the target image sequence without creating any temporal artifacts, and can take advantage of parallel processing architectures to ensure that the recolouring is completed quickly.

4.2. Camera pose estimation

As we consider scenarios with low-cost handheld cameras, it may become necessary to update the camera pose parameters for every frame of a given multiview sequence. However, applying high quality SfM frame-by-frame could be intractable and computationally very expensive. Similarly, monocular SLAM algorithms may fail due to their dependency on good initialisation and their instability during very small motions, which is the typical case with handheld devices. We therefore devise an approach where we estimate accurate calibration using SfM only at certain time intervals within the sequence. In between, we apply a novel algorithm to interpolate calibration parameters for each frame.

More formally, let $\boldsymbol{S} = \{\boldsymbol{S}_1,...,\boldsymbol{S}_n\}$ represent all n handheld video sequences, with $\boldsymbol{s}_i(j), j \in \{1,...,N\}$ denoting the jth frame of video sequence $\boldsymbol{S}_i \in \boldsymbol{S}$. Furthermore, let's define a feature in an image $\boldsymbol{S}_i(j)$ as:

$$\boldsymbol{f}_j(k) = \{\boldsymbol{x}_j(k), \boldsymbol{d}_j(k)\}, \quad \boldsymbol{x}_j(k) \in \mathbb{R}^2, \boldsymbol{d}_j(k) \in \mathbb{R}^d \tag{1}$$

where $\boldsymbol{x}_j(k)$ corresponds to the 2D position of a feature k in frame $\boldsymbol{S}_i(j)$ and $\boldsymbol{d}_j(k)$ represents the descriptor of feature $\boldsymbol{f}_j(k)$ in space d. As we are using SIFT features [20], the space size of the descriptor is always set to $d = 128$.

We estimate the cameras' poses by applying SfM on a small subset of frames $n \cdot m$ (with $m \subset N$) from every data set, denoted as *keyframes*, typically one per second. The keyframe poses are then used as a reference for performing a custom interpolation process for the rest of the intermediate frames. We exploit the fact that every keyframe has very accurate 2D ↔ 3D correspondences, computed during the triangulation and bundle adjustment process of the SfM pipeline. If $\boldsymbol{S}_i'(j)$ and $\boldsymbol{S}_i(j+1)$ represent a keyframe and the following frame in a sequence $\boldsymbol{S}_i \in \boldsymbol{S}$, the first step towards finding the camera pose of $\boldsymbol{s}_i(j+1)$ is to compute successive 2D matches between the two frames. We use the Nearest neighbour search (NNS) matching approach [21] and the filtering methods introduced by Moulon et al. [22].

In the second step, when all successful matches have been found for frame $\boldsymbol{S}_i(j+1)$, every feature $\boldsymbol{f}_{j+1}(k)$ will have a valid match $\boldsymbol{f}'_j(k)$ in frame $\boldsymbol{S}_i'(j)$, which is known to correspond to a 3D point in the reconstruction. The updated 2D ↔ 3D correspondences are then used as an input to a P*n*P algorithm for computing the camera pose for $\boldsymbol{S}_i(j+1)$. Depending on the 3D geometry of the scene, different P*n*P algorithms can be applied. We make use of the Efficient P*n*P algorithm [23], a robust approach for estimating the camera pose from a variety of planar and non-planar 3D surfaces. The camera pose parameters of $\boldsymbol{S}_i'(j)$ are used as an initial estimate for frame $\boldsymbol{S}_i(j)$ and are further optimised.

Furthermore, our experiments showed that the translation parameters of the pose, computed by EP*n*P are highly influenced by the 2D distribution of all successive matches for every frame, resulting to high jumps between subsequent frames. To overcome this issue, we run a two way pass for every frame in the sequence (except the keyframes) and compute the final pose by applying a linear interpolation between the two predictions, relating also the distance of the current frame to it's last keyframe. More formally, if $\boldsymbol{T}_j^{1\to N}$ and $\boldsymbol{T}_j^{N\to 1}$ represent the translation parameters of a camera for frame $\boldsymbol{S}_i(j)$, the following relation should hold:

$$\boldsymbol{T}_j = \alpha \boldsymbol{T}_j^{1\to N} + (1-\alpha)\boldsymbol{T}_j^{N\to 1} \tag{2}$$

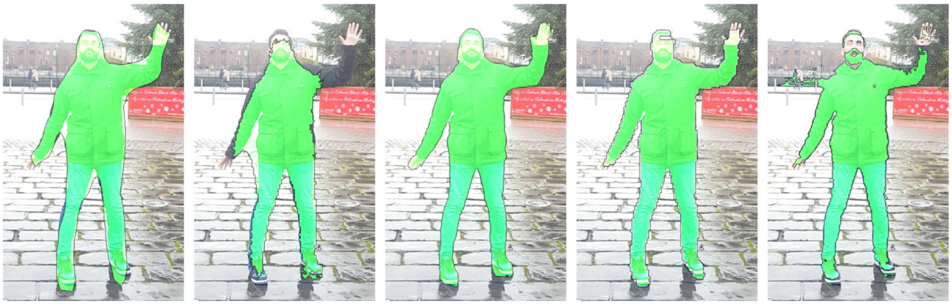
where α is a parameter that increases with the distance of the current frame to the previous last keyframe.

4.3. Foreground segmentation

Segmentation is a crucial component in many FVV systems, which directly affects the quality of succeeding processing steps, such as computing the 3D shape of an object from 2D silhouettes. Obviously, the outline of the 2D projections of objects to be reconstructed should be estimated as accurately as possible. However, this is still a very challenging task, which is why many systems rely on green-screen capture. In order to investigate the suitability of latest state-of-the-art segmentation algorithms for our purposes, we compare several up-to-date approaches, including most recent OSVOS [24] and PSPNet [25] as well as others such as CRF-RNN [26], VOF [27] and BVS [28]. Each approach is assigned to one predefined category, depending on it's architecture and input requirements. Visual results can be seen in Fig. 2.

For numerical comparison we interactively segmented the one of our handheld sequences using appropriate production tools and present these results as ground-truth. With this, we consider a variety of well-known evaluation metrics (Hamming Loss [29], Normalised Hamming Loss [30]), but also recent complementary metrics for video object segmentation presented in [31]. Table 1 shows the overall results of different evaluation metrics.

From the results, it is clear that the OSVOS approach outperforms other methods in this case of human segmentation under very challenging conditions. Even though these results might not be good enough for many reconstruction applications, our system does not rely on very accurate silhouettes and can produce good results even when the



(a) PSPNet (b) CRF-RNN (c) OSVOS (d) VOF (e) BVS

Fig. 2. Qualitative object segmentation results for different state-of-the-art algorithms.

Table 1
Overall segmentation results of different evaluation metrics for each algorithm. Accuracy increases with respect to the arrow direction. Best performances are presented in bold.

			Supervised		Semi-supervised		Other
	Measure		PSPNet	CRF-RNN	OSVOS	VOF	BVS
Hamm. Loss		↓	12909	17155	**8532**	9803	11948
Norm. Hamm. Loss		↑	0.815	0.770	**0.884**	0.858	0.834
$\mathcal{J}$	Mean $\mathcal{M}$	↑	0.846	0.775	**0.889**	0.885	0.835
	Recall $\mathcal{O}$	↑	1.000	1.000	1.000	1.000	1.000
	Decay $\mathcal{D}$	↓	−0.007	−0.013	−0.004	0.0001	**−0.015**
$\mathcal{F}$	Mean $\mathcal{M}$	↑	0.942	0.874	0.939	**0.954**	0.893
	Recall $\mathcal{O}$	↑	1.000	1.000	1.000	1.000	1.000
	Decay $\mathcal{D}$	↓	−0.0007	−0.005	0.020	**−0.009**	0.011
$\mathcal{T}$	Mean $\mathcal{M}$	↓	0.090	0.148	0.110	**0.089**	0.126

foreground masks include part of the background or even when part of the body is wrongly segmented.

5. Scene reconstruction

We consider the case of having very sparse camera setups. On the one hand, this means that MVS reconstruction techniques might fail computing a dense point cloud covering all the subject's surface sufficiently, as the separation between cameras can be large and thus information shared between views limited. On the other hand, silhouette-based reconstruction techniques suffer from significant occlusions in such setups, which may result in significantly inflated volumes. Applying voxel colour consistency in SfS helps detecting concavities and reducing these inflations, but the colour deviation analysis might fail when just two or three images 'see' a specific voxel.

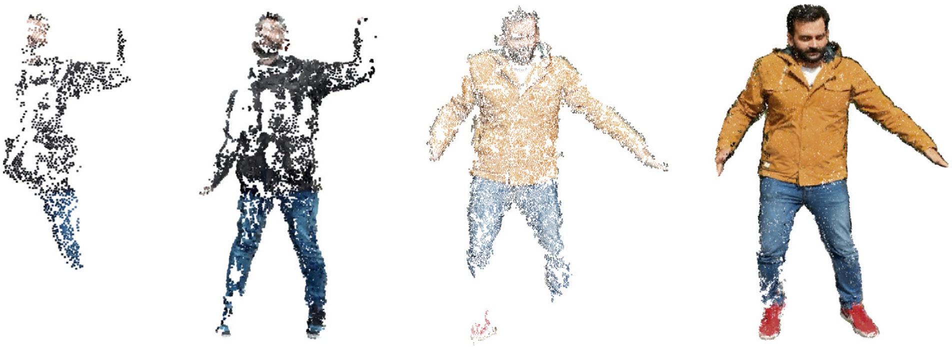
As it is clear that neither MVS nor SfS may provide sufficient accuracy, we propose a combined approach that benefits from the strengths of both. First, we compute a dense point cloud using a custom MVS system. Second we apply SfS with voxel colour consistency check and additional input from an estimated 3D skeleton (MS-SfS). This data is input to a fusion stage, which controls deformation of the estimated volume, using the dense point cloud as reference. This way, our resulting models preserve the detail of the dense point cloud and the completeness of the estimated volume while avoiding inflation. Finally, we texture the resulting mesh using the input images.

5.1. Dense point cloud generation

Dense point cloud estimation techniques normally rely on a two stage process. Initially, a sparse point cloud is calculated through SfM, typically using SIFT features. Secondly, a patch-based point cloud densification algorithm, such as PMVS [32], generates the final dense cloud. The density of the resulting 3D point cloud is directly related to the number of cameras and amount of overlap in the images. For sparse settings, MVS methods are less reliable and might provide inaccurate estimates, so it is crucial to maximise the number of feature-based points in the sparse cloud.

Instead of using SIFT features, our approach is based on the work by Berjón et al. [33], which uses A-KAZE [34,35] in combination with Mult-Scale Retinex image enhancement [36]. This process creates images with high contrast that enhance dark areas, making the feature detection easier and more reliable. The final point cloud is fed to the MVS system proposed by Schoenberger et al. [37]. Fig. 3 compares our result for two sequences with SIFT + PMVS.

Using the foreground masks, we can get an idea of with points belong to the foreground ($\mathcal{P}_f$) and to the background ($\mathcal{P}_b$). However, we



(a) SIFT+PMVS (b) Ours (c) SIFT+PMVS (d) Ours

Fig. 3. Difference between using SIFT + PMVS to get a dense point cloud ((a) and (c)), and using our method ((b) and (d)) on two different sequences.

cannot use the these masks as a binary filter, as any error in the silhouettes would be also introduced in the dense point cloud. Instead, we use them to initialise both groups of points, and the final decision is based on the estimation of Euclidean clusters of points. The foreground point cloud, $\mathscr{P}_f$, will be used later to enhance the estimated volume; however, we can already use the background point cloud, $\mathscr{P}_b$, to generate an accurate background model for our sequence.

5.2. Background model reconstruction

As most of the overlap between images is produced in the background of the scene, the density of the background point cloud $\mathscr{P}_b$ is normally high. This way, we can use it to generate a static 3D model which will enhance the visual experience of our application. Although this point cloud is normally sufficient to generate a photo realistic model of the background (for instance, in the Soccer sequence), it is also possible to register more images to the model and generate an even more complete model.

We remove noisy points using a statistical outlier removal approach based on the assumption that when analysing the neighbourhood of a point $\mathbf{p} \in \mathbb{R}^3$, the distribution of the euclidean distances to the rest of the points in the neighbourhood follows a Gaussian distribution. This way, we can remove points lying outside the standard distance deviation. The final mesh is computed using Poisson surface reconstruction (PSR) [15].

5.3. 3D skeleton estimation

We estimate a 3D skeleton of the subject in every frame by triangulating a set of 2D skeletons detected in the input images (Fig. 4). This is done using Part Affinity Fields (PAFs) [38,39] which uses a convolutional neural network to detect and associate the 2D joints simultaneously in the images. This way, for every image i we have set of m detected skeletons $s^i = \{s^i_1,\ldots,s^i_m\}$, each of them composed of a set of n 2D joints $j^i = \{\mathbf{j}^i_1,\ldots,\mathbf{j}^i_n\}$ and a set of confidence values $c^i = \{c^i_1,\ldots,c^i_n\}$. To filter the unwanted skeletons that PAFs might find people walking by, or in the audience, we use the foreground masks. Moreover, in case there is more than one performer, we apply epipolar constraints to the scene, so we can correctly match the skeletons in different images.

The final 3D joint coordinates of each skeleton are estimated by minimising a set over determined linear triangulation problems [40] in the form of $A\,\mathbf{J}_k = \mathbf{0}$, where $\mathbf{J}_k$ is the 3D position of joint k in the current skeleton. Matrix A is defined using the Direct Linear Transformation [40] method. Thus, for every 2D joint in skeleton $i, \mathbf{j}^i_b = (x^i_b, y^i_b)$, with $b \in \{1,\ldots,n\}$, and a confidence value c^i_b above a certain threshold λ, we add two rows to matrix A:

$$A = \begin{bmatrix} x^i_b \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{3T}} - \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{1T}} \\ y^i_b \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{3T}} - \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{2T}} \\ \vdots \end{bmatrix} \tag{3}$$

where $\boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{1T}}, \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{2T}}, \boldsymbol{p}_{\boldsymbol{i}}^{\boldsymbol{3T}}$ are the rows of projection matrix P_i, corresponding to image i. The threshold λ can vary from sequence to sequence but it must be above 70% to avoid outliers in the minimisation problem.

5.4. Multi-Source Shape-from-Silhouette (MS-SfS)

As traditional SfS techniques cannot handle concavities and suffer from occlusions in very sparse setups, it is necessary to apply additional space carving techniques to get closer to the real volume of the scene. In our case, we define a carving function that is composed of three different terms. Let $\mathscr{V}$ be a shape we want to acquire, our goal is to estimate the volume $\mathscr{S}$, represented by a set of voxels $\mathbf{p} \in \mathbb{R}^3$, such that the difference with respect to $\mathscr{V}$ is minimised. Being $\mathscr{C}$ the set of registered images and $\mathscr{M}$ their corresponding foreground masks, we threshold a carving function composed by three different scores:

$$\mathscr{F}_c(\mathbf{p}) = \phi_{sil}(\mathbf{p},\mathscr{M}) \cdot \phi_{cc}(\mathbf{p},\mathscr{C}) \cdot \phi_{skel}(\mathbf{p},s), \tag{4}$$

where $\phi_{sil}(\mathbf{p},\mathscr{M})$ is the main binary factor defined by the silhouettes.

To estimate the colour consistency score $\phi_{cc}(\mathbf{p},\mathscr{C})$ we analyse the colour variance of the projection of each point $\mathbf{p}$ onto each of the images, assuming that the scene is lambertian. Differently to what it is done in Image-Based Photo Hulls [41], where they use a combination of the RGB colour channels, or in work by Collet et al. [1], where they use the CIELAB colour space, we use the variance measured on the hue channel of the HSV colour space. This helps us accepting less relevant differences in saturation and value that might remain in the images after the colour correction stage, normally due to differences in the camera sensor or white balance, while taking decisions based only on the difference in hue, which is the actually differentiating quantity. Therefore, $\phi_{cc}(\mathbf{p},\mathscr{C})$ will be the standard colour deviation σ of the colour hue value that $\mathbf{p}$ presents when it is projected onto each image. The colour consistency check is specifically suited for smoothing sharp edges that might appear in sparse SfS setups, as it might be in our case. It also makes the system more robust to noisy foreground masks that might include part of the background.

The third score, $\phi_{skel}(\mathbf{p},s)$, increases with the Euclidean distance of each voxel $\mathbf{p}$ to its closest bone in the skeleton. This score is specifically good in the presence of large occlusions, where the colour consistency might fail, as voxels far from the subject's skeleton will be rated with a very low score.

Fig. 5 illustrates the effect of both colour consistency and skeleton scores. It shows an extremely inflated volume due to severe occlusions in a sparse setup. On the left we see the estimated volume just using only $\phi_{sil}(\mathbf{p},\mathscr{M})$, on the right, the result of applying our carving method. Edges have been softened significantly thanks to the colour consistency test. Moreover, a great number of voxels have been removed due to the skeleton score.

5.5. Data fusion

This stage finally combines the independent geometry estimates from our MVS and MS-SfS modules including the advantages of both. Let M_v be the surface of the volume defined by $\mathscr{S}$, estimated through

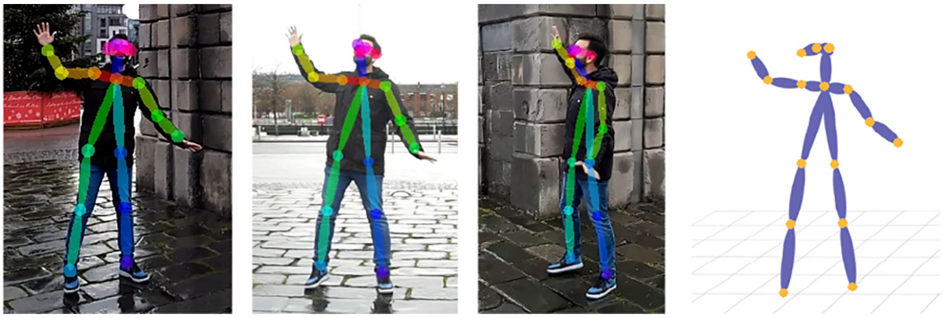


Fig. 4. Skeleton detected in several input images [38], and the final 3D reconstruction.

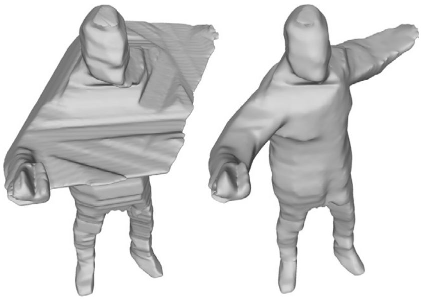


Fig. 5. Result of simple SfS (left) and our MS-SfS (right), in a case with severe occlusions.

the Marching Cubes algorithm [42]. On the other hand, we compute the surface M_f defined by the previously obtained foreground dense point cloud $\mathscr{P}_f$ (see Section 5.1) using PSR [15], and only keeping vertices that are close to the original point cloud, removing interpolated mesh areas. Our approach is using M_f to guide a controlled deformation of M_v, this way, the resulting model M_v' will have both the details captured by the feature points and full volume completeness. This approach also helps reducing the effect of wrongly estimated foreground masks that might produce holes in the MS-SfS volume.

For this, we cast a ray from every vertex $\mathbf{v}_i$ of M_f following its normal, searching for a intersection with M_v. If found, $\mathbf{v}$ will move to the point of intersection. Following the notation in [43], the set of vertices displaced after this stage define the *handle* points $\mathscr{H}$ of our deformation problem. The deformation region $\mathscr{R}$ grows from the handle region by iteratively searching the neighbours of each $\mathbf{v}_i^h \in \mathscr{H}$: for each $\mathbf{v}_j^r \in \mathscr{R}$ we assign a level that increases with number of steps we need to take to approach a vertex of the handle. We also store which is the closest handle vertex and its corresponding displacement vector $\mathbf{d}_i^h$. We can define the displacement function for each $\mathbf{v}_i^r$ as follows:

$$d(\mathbf{v}_j^r) = \mathbf{v}_j^r + \mathbf{n}_j^r \cdot \mathbf{d}_i^h \frac{l - l_j}{l}, \tag{5}$$

where $\mathbf{n}_j^r$ is the normal of vertex $\mathbf{v}_j^r, l$ is the total number of levels in the deformation region, and l_j is the current level. We reduce the possible artifacts by remeshing to our result and by identifying and removing isolated triangle isles and non-manifold edges and vertices. Section 6.1 presents an evaluation of the reconstruction process.

5.6. Multi-view colouring

Once the surface of the subject has been reconstructed, we colour the resulting 3D mesh using the image blending technique proposed by Pagés et al. [44]. This technique is able to combine the colour information provided by the different cameras seamlessly, even though there might be over or underexposed images and differences in colour balance. The blending function is firstly defined in the topology space by backprojecting every facet F_i to each registered image C_j obtaining a per-facet per-image rating R_{ij}. This way, the higher the area of the backprojection of F_i onto C_j, the higher R_{ij}. However, the final rating is smoothed using the angle between the normal of F_i and C_j, penalising wider angles even when the camera is very close to the mesh. Besides, we perform an occlusion test for every facet and image, similar to a z-buffer, which avoids the inclusion of wrongly projected areas on the textured model.

Moreover, to further improve the visual quality of our textured models, particularly of human faces, we search for faces in all images and determine the camera with the largest facial region. The ratings R_{ij}^F of the facets belonging to that detected area are significantly increased, giving a much larger contribution to that particular camera.

To ensure smooth transitions across the mesh, each vertex $\mathbf{v_k}$ also gets a per-image rating r_{kj} by averaging the ratings of all the faces that contain it. The final colour for each point of the mesh is a weighted average of the camera contributions, bilinearly interpolating ratings r_{kj}.

6. Results

6.1. Reconstruction evaluation

The proposed system has been tested in several different datasets with varying conditions: hand-held and fixed cameras, sparse and dense camera setups, indoor and outdoor. Our system is focused on casually captured scenes: similar outdoor sequences that present a sparse setup of a limited number of hand-held cameras (between 6 and 9 cameras). However, we also tested it in other more professional scenarios, such as the Soccer sequence, provided by the Nagoya University Multi-view Sequence database,[4] which contains a denser setup of fixed cameras (up to 20); the DancingDuo [1], which is a dense setup of 106 calibrated and synchronised cameras in a green-screen studio with controlled lighting conditions; and the Swift sequence, similar to the previous one, but with only 12 cameras.

Fig. 6 shows several frames of the same sparse hand-held sequence, in different stages of the algorithm. In the first column, the original SfS model; in the second, our enhanced MS-SfS; in the third, the result of the data fusion; and in the last, the textured final model. As it is possible to see, MS-SfS deals with most the problems that SfS techniques have in sparse camera setups. It is also possible to see how the details of the mode are enhanced after the data fusion stage, which introduces information from the point cloud in the final model. Fig. 7 shows a very interesting case where the hand was lost in the first reconstruction stage, but recovered after the data fusion.

Fig. 8 shows the result of using our system in more professional scenarios. On the left, a model of the Jonathan Swift sequence. As it is possible to see, the data fusion is able to provide details in the face, occluded by the wig in the original SfS. On the right, a model of the DancingDuo [1] reconstructed only using the 53 RGB cameras provided. This dataset also contains 26 IR stereo pairs images with a structured light pattern projected onto them. However, as our purpose is keeping as close as possible to the average user, we do not use them for our reconstruction.

In the absence of a ground truth, which is difficult to obtain for a real-world dynamic scene, we attempt to measure the objective quality of our reconstruction technique by using the Microsoft DancingDuo sequence. This sequence is composed of a set of 53 RGB images, most of them configured in stereo pairs. Using a sample frame mesh generated by Collet et al. [1] as ground truth, we perform a sequence of reconstructions reducing the number of cameras by one each iteration. This way, we can evaluate how accurate a shape we can get when the number of input cameras is considerably reduced. We have divided this experiments into two: for the first one, we iteratively remove cameras trying to preserve as much scene coverage as possible, which means that the reconstruction benefits volume estimation over dense point cloud estimation. For the second, we remove them prioritising camera overlap, which in this the opposite to the previous configuration. Fig. 9 shows the result of computing the Hausdorff distance [45] between the ground truth mesh and three different models in each case: the result of a simple SfS process (red line) and the result of our reconstruction system (blue line). The y-axis expresses the Hausdorff distance as a percentage: RMS distance with respect to the global volume.

As shown in Fig. 9, our method performs best for every camera configuration. In the first experiment we can see that the error is considerably lower due to the effect of the details from the dense point cloud, included in the fusion stage. However, as it is not possible to estimate a point cloud when the number of cameras is below 10 (the overlap is minimum for this camera configuration at that stage), the

[4] http://fujii.nuee.nagoya-u.ac.jp/multiview-data/.

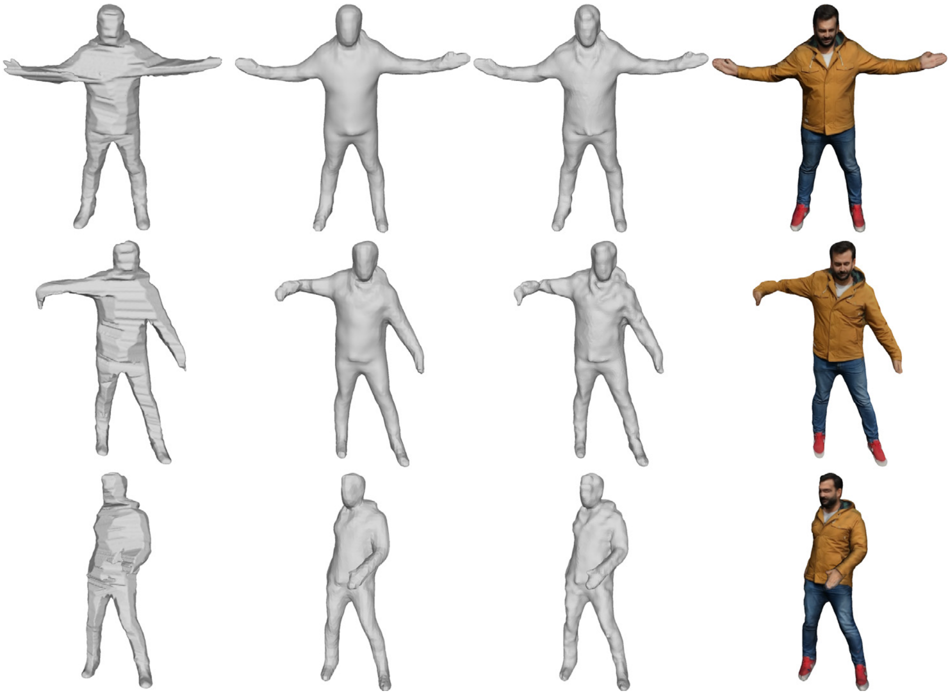


Fig. 6. Three different frames of the same sequence, in four different stages of the system. From left to right: original SfS model, result of the MS-SfS, result after fusion and final textured model.

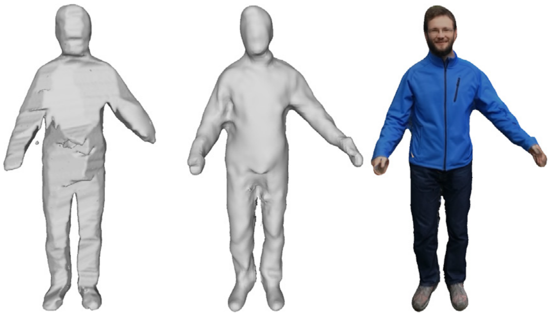
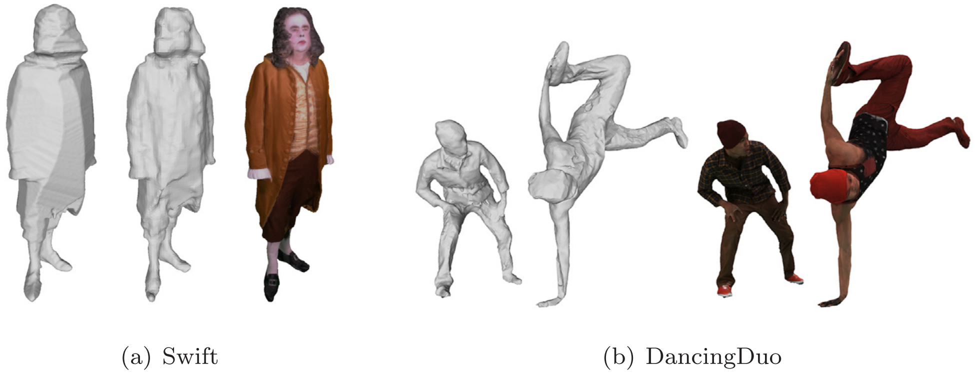


Fig. 7. Example where one hand of the model is lost in the original SfS model (left column), and recovered after the data fusion stage (centre column). On the right, the final textured model.

Fig. 10 helps illustrating the differences between both approaches. The result of three reconstructions is shown: using all the 53 images, using 18 with experiment 1, and using 18 cameras with experiment 2. In each case you can see the result of applying traditional SfS on the left, the result of meshing the dense point cloud obtained through our enhanced MVS on the centre, and the result of our data fusion technique on the right. As it is possible to see, when we have all 53 images, the result of meshing the dense point cloud is very accurate and the carving very restrictive, minimising the maximum Hausdorff distance to 3–4 cm (see Fig. 8 for a textured result). When we perform the reconstruction using only 18 cameras with the setting of the first experiment, the maximum Hausdorff distance of our method increases to around 9 cm (red colour, e.g. neck) in concave areas, which are neither well covered by the dense point cloud. This means that both inputs fail to resolve such parts sufficiently. Observing Fig. 10 (c) one could guess where the cameras are located in experiment 2, as the MVS model presents very fine detail on the right side. However, as the carving is much worse in this case, the final model of our method is also visually worse than in experiment 1 (with a maximum distance of around 11 cm).

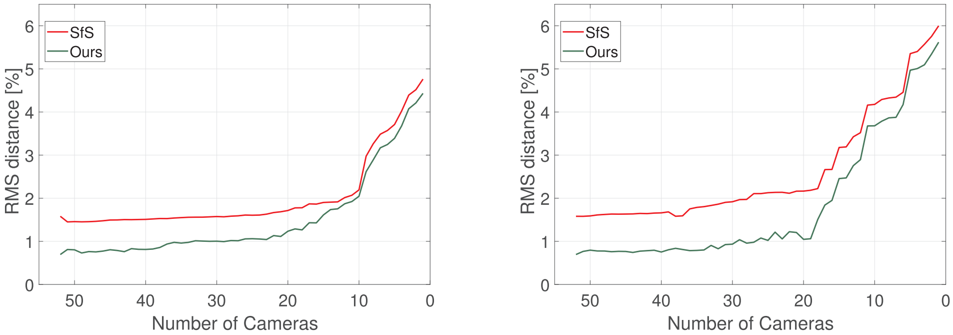


(a) Swift

(b) DancingDuo

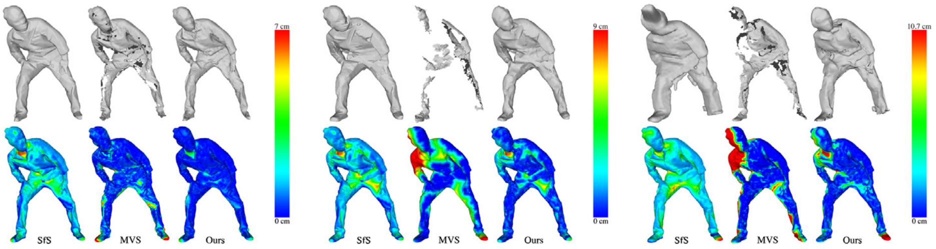
Fig. 8. Result of using our system in two controlled scenes. The Swift sequence uses 12 cameras and the DancingDuo 53.

error grows closer to the original SfS. Further, in the second experiment, we see that with decreasing number of cameras, as SfS is less and less able to carve a proper volume, its Hausdorff distance grows rapidly. With our method, using the skeleton to discard superfluous voxels, helps to reduce the error considerably.



(a) Experiment 1 (b) Experiment 2

Fig. 9. Hausdorff distance between ground truth and: basic SfS (red), and our method (green). Distance is measured with respect to the global volume. Experiment 1 reduces the number of cameras prioritising coverage, while experiment 2 prioritises camera overlap. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) 53 RGB cameras (b) Experiment 1: 18 cameras (c) Experiment 2: 18 cameras

Fig. 10. Hausdorff distance measured in cm with respect to the ground truth. On the left (a), reconstruction using all 53 cameras; on the centre (b), only 18 using experiment 1; on the right (c), 18 cameras using experiment 2.

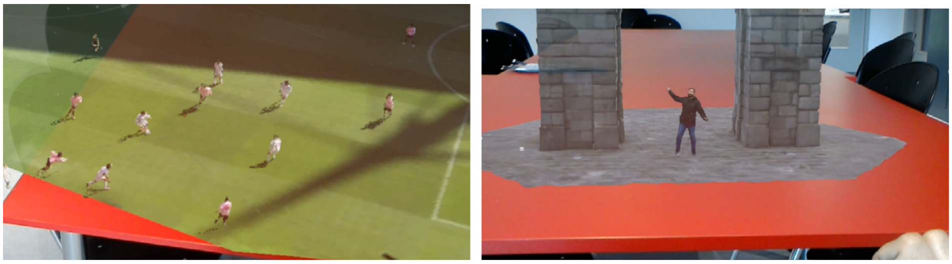


Fig. 11. Soccer and one of our hand-held sequences seen through Microsoft HoloLens.

6.2. *Scene visualisation and examples*

One of the traditional way of displaying FVV content is using view synthesis, where in-between camera views are generated from the nearest two (or more) camera images. With the advancement of VR/AR technology, especially on mobile devices, new application areas are emerging.

To display the reconstructed scenes we use several representations depending on the targeted use. The static objects in the scene are always rendered as textured 3D mesh. Dynamic part of the FVV scene (i.e. people) are then represented either as point cloud or textured mesh. This results from the need of different display methods and performance requirements and limitations of different platforms.

6.2.1. *VR/AR rendering*

For real-time rendering, especially in mobile VR/AR, there are several restrictions. First is the amount of vertices various devices can process. Here we use triangulated mesh with a texture as the data format. Another restriction is the size of RAM in mobile devices. FVV sequences can be very large (over 1.5 GB/s, depending on the quality and frame-rate) and need to be loaded into memory at run-time. Due to these restrictions we had to reduce the quality of the original meshes and keep the number of vertices bellow 10k and texture resolution at 1K per frame. In our system, for each frame we store *vertices*, *normals*, *texture coordinates* and a *texture* compressed for targeted platform (DXT1, ETC1). This allows us to display our sequences on various mobile device in real-time without length restrictions. Fig. 11 shows some AR examples.

6.2.2. *View synthesis*

The basic view synthesis mode we implemented is similar to [2]. The virtual camera path is restricted to a trajectory between real cameras. When positioned at real camera, the virtual camera V is using extrinsic and intrinsic parameters from the selected real camera A and the camera's image is projected onto the dynamic geometry and static background mesh. During the transition phase, when the user changes the viewpoint, new parameters of the virtual camera are computed as an interpolation between the current camera A and the destination camera B. In the case of different image resolutions or focal lengths, we use linear interpolation to compute virtual camera parameters. For its virtual position and rotation we use Spherical Linear intERPolation (SLERP).

The rendering uses as an underlying geometry a point cloud that provides the necessary depth information but doesn't requires colour which comes from the original camera images. To improve the quality we also use normals and render the point cloud as normal-oriented quads. This has a major impact on the quality of the final rendering as seen on Fig. 12.

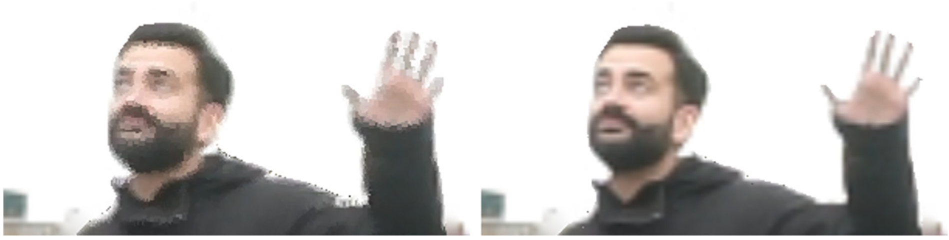


Fig. 12. View synthesis example using a point cloud rendered as billboards (left) and as *normal-oriented quads* (right).

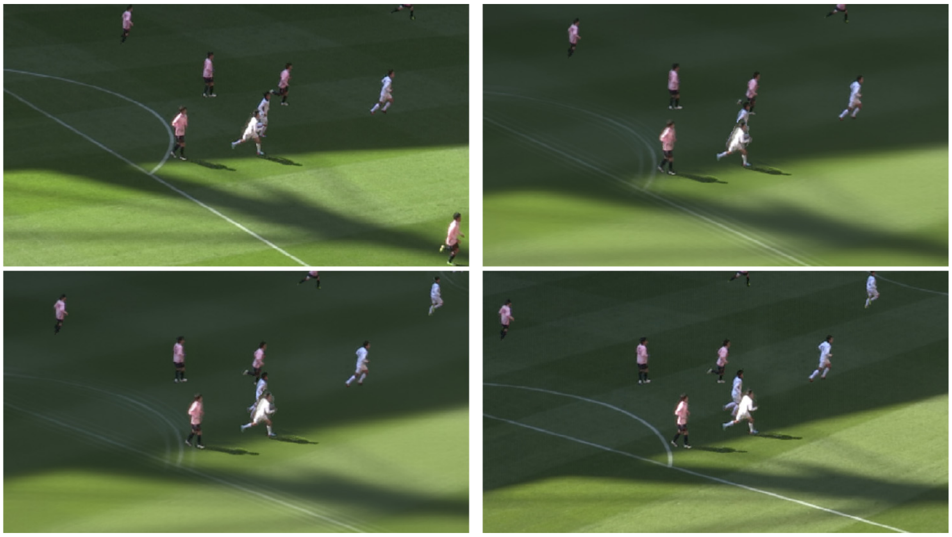


Fig. 13. A view synthesis example with transition frames from football scene.

The pixel's colour is then computed using the Multi-View Colouring stage presented in Section 5.6 adapted to real-time rendering. In our real-time version we consider only the current frame and the two nearest camera images (cameras *A* and *B*). The rendering pipeline is then similar to the traditional pipeline from computer graphics for shadow mapping, where we use cameras instead of lights. In the last rendering pass we apply an optional post-processing effect, a screen-space motion blur on the background geometry during the transition phase to highlight the dynamic content (see Fig. 13).

7. Conclusion

It is highly challenging to create virtual viewpoints of a scene that has only been recorded from a limited number of views. In vision-based 3D reconstruction systems, a large number of overlapping high-speed high-resolution cameras, perfect calibration, synchronisation, foreground segmentation and large computer processing power allow for the acquisition of a, near to perfect quality, 3D reconstructed scene. However, for most people access to these professional studio-quality setups in out of reach. In this paper we presented a novel pipeline for affordable content creation for free-viewpoint video, virtual and augmented reality, which is accessible to everyone. We targeted a scenario where only a small number of consumer-quality cameras are used but also demonstrated the scalability of our method to professional setups, such as in football stadiums or 3D capture studios.

Furthermore, we compared several state-of-the-art segmentation algorithms, showing the shortcomings of the approaches to cope with large shape variations in the human motion. In future work the quality of the foreground segmentation could potentially be improved by incorporating multi-view information. We evaluated the scene reconstruction using a quantitative analysis of several challenging examples, using Microsoft's multi-camera system [1] as ground truth. For the latter, two different camera configurations were presented: maximising the scene coverage, which benefits SfS over MVS; maximising camera overlap, which is the opposite case. These two different setup show the different kind of possible results depending on the input data, and give some examples of the limitations. Our method highlights the limitations when it is pushed to very extreme scenarios with very low scene coverage or no camera overlap. However, as our reconstruction technique is based on the fusion of different kinds of data, it always produces a result, even when some if this data is not available. In a real-world scenario a camera setup would normally fall somewhere in the middle of these two extremes.

We presented an end-to-end system to process and visualise FVV sequences. Our lightweight system is robust and flexible and produces high quality 3D content with a small number of cameras by combining state-of-the-art reconstruction techniques with efficient camera pose estimation, skeletal restraint, colour consistency and multi-view stereo restraint techniques. Multi-source shape-from-silhouette (MS-SfS) combined with fusion of different geometry data are the crucial components for accurate reconstruction in sparse camera settings. Our approach produces flexible outcome for low-cost content creation for VR/AR.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant No. 15/RP/2776.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jvcir.2018.03.012.

References

[1] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, S. Sullivan, High-quality streamable free-viewpoint video, ACM Trans. Graph. (TOG) 34 (4) (2015) 69.

[2] L. Ballan, G.J. Brostow, J. Puwein, M. Pollefeys, Unstructured video-based rendering: interactive exploration of casually captured videos, ACM Trans. Graph. (TOG) (2010) 1–11.

[3] A. Smolic, 3d video and free viewpoint video – from capture to display, Pattern Recogn. 44 (9) (2011) 1958–1968.

[4] S.B. Kang, R. Szeliski, P. Anandan, The geometry-image representation tradeoff for rendering, International Conference on Image Processing, vol. 2, IEEE, 2000, pp. 13–16.

[5] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, ACM Trans. Graph. (TOG) 23 (2004) 600–608. ACM.

[6] C. Lipski, C. Linz, K. Berger, A. Sellent, M. Magnor, Virtual video camera: image-based viewpoint navigation through space and time, Computer Graphics Forum, vol. 29, Wiley Online Library, 2010, pp. 2555–2568.

[7] F. Angehrn, O. Wang, Y. Aksoy, M. Gross, A. Smolic, Mastercam fvv: robust registration of multiview sports video to a static high-resolution master camera for free viewpoint video, IEEE International Conference on Image Processing (ICIP), IEEE, 2014, pp. 3474–3478.

[8] K. Hayashi, H. Saito, Synthesizing free-viewpoing images from multiple view videos in soccer stadiumadium, IEEE International Conference on Computer Graphics, Imaging and Visualisation (CGIV), IEEE, 2006, pp. 220–225.

[9] T. Popa, M. Germann, R. Ziegler, R. Keiser, M. Gross, Geometry reconstruction of players for novel-view synthesis of sports broadcasts, Computer Vision in Sports, Springer, 2014, pp. 133–160.

[10] S. Croci, S. Aljoscha, O. Wang, Dynamic time warping based 3d contours, in: M. Bronstein, J. Favre, K. Hormann (Eds.), Vision, Modeling and Visualization, The Eurographics Association, 2013.

[11] K.N. Kutulakos, S.M. Seitz, Theory of shape by space carving, Int. J. Comput. Vision 38 (2000) 199–218.

[12] J. Starck, G. Miller, A. Hilton, Volumetric stereo with silhouette and feature constraints, in: British Machine Vision Conference, 2006, pp. 1189–1198.

[13] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, A comparison and evaluation of multiview stereo reconstruction algorithms, in: IEEE Conference on Computer Vision and Pattern Recogn., vol. 1, 2006, pp. 519–528.

[14] L. Yebin, D. Qionghai, X. Wenli, A point-cloud-based multiview stereo algorithm for free-viewpoint video, IEEE Trans. Visual. Comput. Graph. 16 (3) (2010) 407–418.

[15] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Trans. Graph. (TOG) 32 (3) (2013) 29.

[16] A. Mustafa, H. Kim, J.Y. Guillemaut, A. Hilton, Temporally coherent 4d reconstruction of complex dynamic scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[17] C. Lipski, F. Klose, M. Magnor, Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video, IEEE Trans. Circuits Syst. Video Technol. 24 (6) (2014) 942–951.

[18] M. Grogan, M. Prasad, R. Dahyot, L2 registration for colour transfer, in: 23rd European Signal Processing Conference (EUSIPCO), 2015, pp. 1–5.

[19] M. Grogan, R. Dahyot, Robust registration of gaussian mixtures for colour transfer, ArXiv e-prints, vol. abs/1705.06091, 2017.

[20] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision (IJCV) 60 (2004) 91–110.

[21] M. Muja, D. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: VISAPP International Conference on Computer Vision Theory and Applications, 2009, pp. 331–340.

[22] P. Moulon, P. Monasse, R. Marlet, Others, Openmvg. An open multiple view geometry library, in: Reproducible Research in Pattern Recognition – First International Workshop, 2016, pp. 60–74.

[23] V. Lepetit, F. Moreno-Noguer, P. Fua, Epnp: an accurate o(n) solution to the pnp problem, Int. J. Comput. Vision (IJC) (2008).

[24] S. Caelles, K.K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, L. Van Gool, One-shot video object segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2017.

[25] Z. Hengshuang, S. Jianping, Q. Xiaojuan, X. Wang, J. Jiaya, Pyramid scene parsing network, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[26] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, S. Su, D. Du, C. Huang, P.H. S. Torr, Conditional random fields as recurrent neural networks, in: International Conference on Computer Vision (ICCV), 2015.

[27] Y.H. Tsai, M.H. Yang, M.J. Black, Video segmentation via object flow, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[28] N. Maerki, F. Perazzi, O. Wang, A. Sorkine-Hornung, Bilateral space video segmentation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[29] R. Hamming, Error detecting and error correcting codes, Bell Syst. Tech. J. 26 (1950) 147–160.

[30] A. Teichman, J. Lussier, S. Thrun, Learning to segment and track in RGBD, IEEE Trans. Autom. Sci. Eng. (2013).

[31] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, A. Sorkine-Hornung, A benchmark dataset and evaluation methodology for video object segmentation, in: Computer Vision and Pattern Recognition, 2016.

[32] Y. Furukawa, J. Ponce, Accurate, dense, and robust multiview stereopsis, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 32 (8) (2010) 1362–1376.

[33] D. Berjón, R. Pagés, F. Morán, Fast feature matching for detailed point cloud generation, in: 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), 2016, pp. 1–6.

[34] P. Alcantarilla, A. Bartoli, A. Davison, Kaze Feat. (2012) 214–227.

[35] P. Alcantarilla, J. Nuevo, A. Bartoli, Fast explicit diffusion for accelerated features in nonlinear scale spaces, Proceedings of the British Machine Vision Conference, BMVA Press, 2013.

[36] D.J. Jobson, Z. Rahman, G.A. Woodell, A multiscale retinex for bridging the gap between color images and the human observation of scenes, IEEE Trans. Image Process. 6 (7) (1997) 965–976.

[37] J.L. Schönberger, E. Zheng, M. Pollefeys, J.M. Frahm, Pixelwise view selection for unstructured multi-view stereo, in: IEEE European Conference on Computer Vision (ECCV), 2016.

[38] C. Zhe, S. Tomas, S.E. Wei, Y. Sheikh, Realtime multi-person 2D pose estimation using part affinity fields, in: CVPR, 2017.

[39] S.E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines, in: CVPR, 2016.

[40] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, second ed., Cambridge University Press, 2004.

[41] G. Slabaugh, R. Schafer, M. Hans, Image-based photo hulls for fast and photo-realistic new view synthesis, Real-Time Imag. 9 (5) (2003) 347–360.

[42] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3d surface construction algorithm, ACM Siggraph Computer Graphics, vol. 21, ACM, 1987, pp. 163–169.

[43] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Lévy, Polygon Mesh Processing, CRC Press, 2010.

[44] R. Pagés, D. Berjón, F. Morán, N. García, Seamless, static multi-texturing of 3D meshes, Comput. Graph. Forum 34 (2015) 228–238.

[45] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, Computer Graphics Forum, vol. 17, Wiley Online Library, 1998, pp. 167–174.