3-D cinematography with approximate and no geometry

Martin Eisemann, Timo Stich and Marcus Magnor

Abstract 3-D cinematography is a new step towards full immersive video, allowing complete control of the viewpoint during playback both in space and time. One major challenge towards this goal is precise scene reconstruction, either implicit or explicit. While some approaches exist which are able to generate a convincing geometry proxy, they are bound to many constraints, e.g., accurate camera calibration and synchronized cameras. This chapter is about methods to remedy some of these constraints.
This is the author version of the paper. The original publication is available at www.springerlink.com and is part of the book *Image and Geometry Processing for 3-D Cinematography* published by Springer.

1 Introduction

As humans we perceive most of our surroundings through our eyes and visual stimuli affect all of our senses, drive emotion, arouse memories and much more. That is one of the reasons why we like to look at pictures. A major revolution occured with the introduction of moving images, or videos. The dimension of time was suddenly added, which gave incredible freedom to film- and movie makers to tell their story to the audience.

Martin Eisemann
TU Braunschweig, Mühlenpfordtstr. 23, 38102 Braunschweig, Germany,
e-mail: eisemann@cg.cs.tu-bs.de

Marcus Magnor
TU Braunschweig, Mühlenpfordtstr. 23, 38102 Braunschweig, Germany,
e-mail: magnor@cg.cs.tu-bs.de

Timo Stich
TU Braunschweig, Mühlenpfordtstr. 23, 38102 Braunschweig, Germany,
e-mail: stich@cg.cs.tu-bs.de

With more powerful hardware, computation power and clever algorithms we are now able to add a new dimension to videos, namely the third spatial dimension. This will give users or producers the possibility to change the camera viewpoint on the fly.

The standard 2-D video production pipeline can be split into two parts, acquisition and display. Both are well understood. 3-D cinematography adds a new intermediate step, the scene reconstruction. This could be a 3-D reconstruction of the scene combined with retexturing for displaying a new viewpoint or a direct estimation of a plausible output image for a new viewpoint without the need for reconstruction. Both methods have their benefits and drawbacks and will be further discussed in the next sections. Purely image-based approaches as Lightfields [36] or the Lumigraph [27] might not need any sophisticated reconstruction as the sheer amount of images allows for smooth interpolation. But more practical systems need to rely on some clever reconstruction as the number of cameras is seldomly larger than a dozen. These are the systems we are interested in this chapter.

2 3-D Reconstruction

Despite the acquired image or video footage one of the most important factors for 3-D cinematography is arguably a 3-D model of the scene. Depending on the task only a 3-D model of the foreground or a complete scene reconstruction is needed. Most methods aim at the reconstruction of the foreground in controlled environments, e.g. the reconstruction of an actor. For proper reconstruction the camera parameters need to be known in advance. These can be acquired by several methods and the choice depends on the task [58, 29, 53].

Usually the most simple camera, a frontal pinhole camera, is assumed. With this model every point $\mathbf{x}_w$ in a 3-D scene can be simply projected to its image space position $\mathbf{x}_{ip}$ using a projection matrix $\mathbf{P}$. Given this dependency between the 3-D world and its 2-D image equivalent reconstruction of the scene geometry is then possible. There are basically three established methods that can compute 3-D information from images alone. These are parameterized template model matching, shape-from-silhouette and depth-from-stereo.

2.1 Model Reconstruction

In Hollywood movies it is quite common to replace actors with hand-made, rendered 3-D models, which are crafted by some designers using modelling tools such as Maya, 3DS Max, Blender, Cinema4D, etc. Using marker-based motion capture methods the model animation can be brought into congruence with the motion of the actor. Nevertheless this is very laborious work and only suitable for studios that are willing to spend a tremendous amount of money on skilled artists and designers,

additionally in some situations, like sports, or public events it might be impossible to place markers on the object of interest.

The Free-Viewpoint Video System of Carranza *et al.* [15] combines motion capture and 3-D reconstruction by using a single template model. In a first step the silhouettes of the object of interest are extracted in all input images. A generic human body model consisting of several segments, i.e. submeshes, and a corresponding bone system is then adapted to resemble the human actor and fitted to the silhouettes of each video frame by an analysis-by-synthesis approach.

A single parameterized template model can seldom represent all possibilities of human shapes sufficiently, therefore the result can be improved by identifying multi-view photo-inconsistent regions and fine-tuning the mesh in these regions by enforcing a color-consistency criterion [3].

Small details usually cannot be sufficiently recovered by these methods, as the underlying mesh is quite coarse, usually only up to 21k triangles [15]. An improvement can be achieved by acquiring a detailed mesh beforehand. One way to do this would be to use a structured light approach [46, 50, 33]. Anguelov *et al.* [7] make use of detailed laser scans of an actor in different poses, from which they learn a pose deformation model and a model of variation for the body shape in order to simulate realistic muscle behaviour on the model. Using a set of markers on the human, their system can use the motion capture data to animate the model. De Aguiar *et al.* also make use of detailed laser scans of the actor which they deform in order to maximize the congruence with the multi-view recordings [5]. Their system is not aiming at muscle behaviour but is more focussed on arbitrary inputs, as e.g. humans wearing different kinds of apparel, and markerless tracking, which is less intrusive. Similar to Carranza *et al.* [15] a template model is fitted to the videos first. In a next step the laser scan is deformed to fit the template model by specifying correspondence points between the two meshes.

To capture non-rigid surface deformations occuring from wearing wide apparel for example, optical flow-based 3-D correspondences are estimated and the laser scanned model is deformed using a Laplacian mesh deformation such that it follows the motion of the actor over time [4]. Fusing of efficient volume- and surface-based deformation schemes, a multi-view analysis-through-synthesis procedure, and a multi-view stereo approach, cf. Section 2.3, can lead to an even higher correspondence match of the mesh with the input video [2]. This allows to capture performances of people wearing a wide variety of everyday apparel and performing extremely fast and energetic motions.

While this approach delivers an evidentiary high quality it is not so well suited for situations in which a high-quality laser scan of the actor cannot be acquired beforehand. For such situations more general methods are needed that are described in the following sections.

2.2 Shape-From-Silhouettes

The shape-from-silhouettes approach by Laurentini *et al.* [34] uses the extracted silhouettes of the object to define its visual hull. In 2-D the visual hull is equivalent to the convex hull, in 3-D the visual hull is a subset of the convex hull including hyperbolic regions. It is the maximal volume constructed from reprojecting the silhouettes cones of each input image back into 3-D space and computing their intersection. As the number of input images is limited, only an approximation of the visual hull, sometimes called inferred visual hull, can be reconstructed, ref. Figure 1. As this

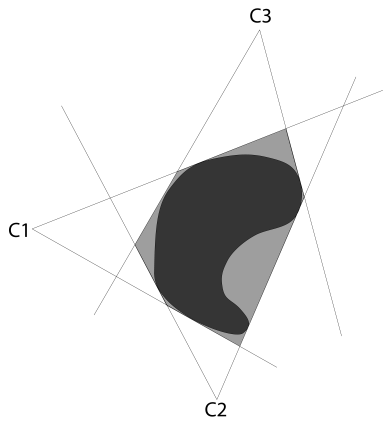

Fig. 1 The inferred visual hull (light gray) of an object (dark gray) is estimated by reprojecting each silhouette cone and computing the intersection.

method rather conservatively estimates the real geometry, results can become relatively coarse. On the other hand this algorithm can be easily implemented on graphics hardware to achieve real-time frame rates, e.g. [40], and can even be calculated in image-space rather than 3-D space [41]. An improvement can be achieved by adding color constraints in order to detect concavities as well [52, 32], also known as space-carving, or to employ an optimization process, as it is done by Starck *et al.* [54]. Their approach combines cues from the visual hull and stereo-correspondences, cf. Section 2.3, in an optimization framework for reconstruction. The visual hull is used as an upper bound on the reconstructed geometry. Contours on the underlying surface are extracted using a wide-baseline feature matching, which serve as further constraints. In a final step an optimization framework based on graph cuts reconstructs the smooth surface within the visual hull that passes through the features while reproducing the silhouette images and maximizing the consistency in appearance between views.

2.3 Depth-From-Stereo

Sometimes a whole scene has to be reconstructed, in which case all previously mentioned methods fail (except for [32]), as they are based on silhouettes, which can no longer be extracted. In this case depth-from-stereo systems perform better, as they extract a depth map for each input image, which can then be used for 3-D rendering. The basic principle of depth-from-stereo is triangulation. Given two corresponding points in two images and the camera parameters, the exact position of this point in 3-D can be reconstructed, see Figure 2. Finding these correspondences can be

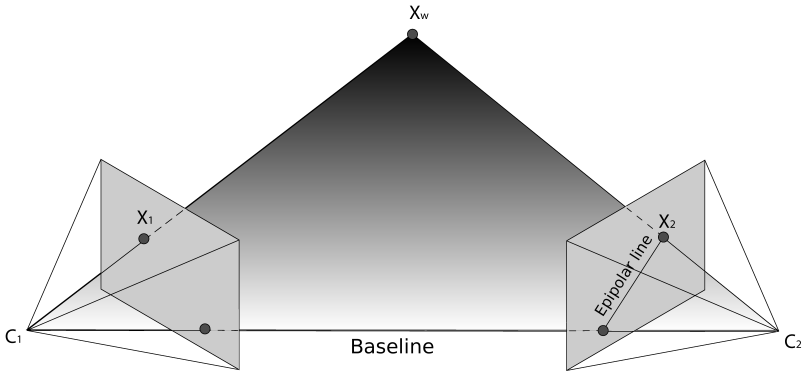

Fig. 2 Using epipolar constraints and triangulation the 3-D position of any scene point can be reconstructed.

arbitrarily hard and sometimes even ambiguous. To relax the problem of doing an exhaustive search for similarity over the whole image, one usually makes use of the epipolar constraint. By employing epipolar constraints the search can be reduced to a 1D line search along the conjugate epipolar lines, Figure 2. Usually a rectification preceeds the line search so that the search can be performed along the same scanline, i.e. the input images are projected onto a plane parallel to the line between the optical centers of the input cameras. The correspondence finding can be performed for example by a window-based matching, i.e. the matching is performed on a $n \times n$ window instead of a single pixel. If further knowledge about the scene is given or scene characteristics are assumed, as for example local smoothness, more sophisticated methods based on energy minimization can be employed, e.g. [11, 10]. If more than two images should be used for depth estimation a plane sweep algorithm can be used [18]. In this approach a plane is placed at different depths, the input images are projected onto it and the plane is rendered from the virtual viewpoint. The color variation at every fragment serves as a quality estimate for this depth value. This approach is especially appealing in real-time acquisition systems as it can be computed very efficiently on graphics hardware [62, 37, 26]. Even dedicated hardware is nowadays available for multi-view stereo reconstruction and has already been successfully applied in an image-based rendering system [43].

One of the first systems to achieve high quality interpolation with a relatively sparse camera setup was the approach by Zitnick *et al.* [64]. Instead of matching single-pixels or windows of pixels, they match segments of similar color. As they assume that all pixels inside a segment have similar disparities an oversegmentation of the image is needed. The segments are then matched and the estimated disparities are further smoothed to remove outliers and create smooth interpolation between connected segments belonging to the same object.

Methods based on this matching are commonly applied only for dense stereo, i.e. the distance between the cameras are rather small, only up to a few dozen inches. For larger distances, or fewer cameras, additional information is needed for reconstruction. Waschbüsch *et al.* [60] use video bricks, which consist of a color camera for texture acquisition, and two calibrated grayscale cameras that are used together with a projector to estimate depth in the scene using structured light. The benefit of these bricks is that depth ambiguities are resolved in textureless areas. These depth estimations are used as initialization for a geometry filtering, based on bilateral filtering, for generation of time-coherent models with removed quantization noise and calibration errors.

There are many other 3-D reconstruction methods, like Shape-from-Texture, Shape-from-Shading, etc. but these are commonly not used for multi-view stereo reconstruction for 3-D cinematography and therefore we refer the interested reader to the appropriate literature.

3 Texturing with imprecise geometry and sparse camera setups

In the previous section several approaches have been presented to reconstruct 3-D geometry from a collection of images. All these reconstruction approaches have usually one thing in common: they are imprecise (especially the faster ones, which are essential for real-time acquisition). While this may not be such a big problem when looking at the mesh alone, it will become rather obvious when the mesh is to be textured again using only a set of input images. One way to circumvent this problem is to use a large enough number of input cameras or a camera array [36, 27, 13, 25, 42] and then simply interpolate between the images in ray space. This is of course arguably the way to achieve the best rendering quality, but rather impractical as thousands of images might be needed for sufficient quality and freedom of camera movement [36]. The number can be effectively reduced by making use of a more precise geometry proxy [27]. But reconstruction of scene geometry is seldomly perfect. Another possibility is to make use of dynamic textures [17]. Here a coarse geometry is used to capture large scale variations in the scene, while the residual statistical variability in texture space is captured using a PCA basis of spatial filters. It can be shown that this is equivalent to the analytical basis. New poses and views can then be reconstructed by first synthesizing the texture by modulating the texture basis and then warping it back onto the geometry. However for a good estimation of the basis quite a few input images are needed. Therefore the challenge

Fig. 3 Comparison of standard linear interpolation using the Unstructured Lumigraph weighting scheme [13] (left) and the Floating Textures approach [20] (right) for similar input images and visual hull geometry proxy. Ghosting along the collar and blurring of the shirt's front, noticeable in linear interpolation, are eliminated by the non-linear approach.

is generating a perceptually plausible rendering with only a sparse setup of cameras and a possibly imperfect geometry proxy.

Commonly in image-based rendering (IBR) the full bidirectional reflectance distribution function (i.e. how a point on a surface appears depending on the viewpoint and lighting) is approximated by projective texture mapping [51] and image blending. Assuming the camera parameters of the input cameras are given the recorded images can be reprojected onto the geometry. If more than one camera is used the corresponding projected color values must be somehow combined for the final result. Usually the influence of a cameras' projected color value to the result is based on either its angular deviation to the normal vector at the corresponding mesh position [15] or its angular deviation to the view vector [19, 13]. The first approach is suitable for lambertian surfaces but can result in either cracks in the texture or, even worse, a complete loss of the 3-D impression, this would e.g. happen to a light-field [36] where the geometry proxy is only a simple quad. The second approach is more general but has to deal with ghosting artifacts if the textures do not match on the surface. This is the case if too few input images are available or the geometry is too imprecise, see Figure 3 for an example. Some approaches prevent the ghosting artifacts by smoothing the input images at the cost of more bluriness. The bandlimiting approach, discussed by Stewart *et al.* in [55], chooses the amount of blur based on the disparity. For light field rendering they propose to add back high frequency details from a wide aperture filtered image, but this approach is only suitable to two-plane light field rendering and not for general IBR or even sparse camera setups. Eisemann *et al.* [21] vary the amount of smoothness depending on the current viewpoint, but the constant change of bluriness in the rendered images can cause distractions. Let us take a more detailed look at the underlying problem before dealing with more convincing solutions.

In a slightly simplified version, the plenoptic function $P(x,y,z,\theta,\phi)$ describes radiance as a function of 3-D position in space (x,y,z) and direction (θ,ϕ) [1]. The notion of IBR now is to approximate the plenoptic function with a finite number of discrete samples of P for various (x,y,z,θ,ϕ) and to efficiently re-create novel views from this representation by making use of some sort of object geometry proxy.

Any object surface that one chooses to display can be described as a function $\mathbf{G} : (x,y,z,\theta,\phi) \rightarrow (x_o,y_o,z_o)$, i.e., by a mapping of viewing rays (x,y,z,θ,ϕ) to 3-D coordinates (x_o,y_o,z_o) on the object's surface. Of course, the function $\mathbf{G}$ is only defined for rays hitting the object, but this is not crucial since one can simply discard the computation for all other viewing rays. Let $\mathbf{G_O}$ denote the function of the true surface of the object, and $\mathbf{G_A}$ denote a function that only approximates this surface, Figure 4.

Next, a variety of camera calibration techniques exist to establish the projection mapping $\mathbf{P}_i : (x,y,z) \rightarrow (s,t)$ which describes how any 3-D point (x,y,z) is mapped to its corresponding 2-D position (s,t) in the i-th image [58, 29]. From its projected position (s,t) in image $\mathbf{I}_i$, the 3-D point's color value (r,g,b) can be read out, $\mathbf{I}_i : (s,t) \rightarrow (r,g,b)$. Then, any novel view $\mathbf{I}^V_{\text{Linear}}$ from a virtual viewpoint V synthesized by a weighted linear interpolation scheme, as employed by most IBR systems, can be formulated as

$$\mathbf{I}^V_{\text{Linear}}(x,y,z,\theta,\phi) = \sum_i \mathbf{I}^V_i(x,y,z,\theta,\phi)\ \omega_i \tag{1}$$

with

$$\mathbf{I}^V_i(x,y,z,\theta,\phi) = \mathbf{I}_i(\mathbf{P}_i(\mathbf{G_A}(x,y,z,\theta,\phi))) \tag{2}$$

$$\omega_i = \delta_i(\mathbf{G_A}(x,y,z,\theta,\phi))\ w_i(x,y,z,\theta,\phi) \tag{3}$$

and $\sum_i \omega_i = 1$. The notation $\mathbf{I}^V_i$ is used to denote the image rendered for a viewpoint V by projecting the input image $\mathbf{I}_i$ as texture onto $\mathbf{G_A}$. δ_i is a visibility factor which is one if a point on the approximate surface $\mathbf{G_A}$ is visible by camera i, and zero otherwise. w_i is the weighting function which determines the influence of camera i for every viewing ray, also called weight map.

Note that (1) is the attempt to represent the plenoptic function as a linear combination of re-projected images. For several reasons, weighted linear interpolation cannot be relied on to reconstruct the correct values of the plenoptic function:

1. Typically, $\mathbf{G_O} \neq \mathbf{G_A}$ almost everywhere, so the input to (2) is already incorrect in most places, Figure 4 left.
2. Due to calibration errors, $\mathbf{P}_i$ is not exact, leading to projection deviations and, subsequently, erroneous color values, Figure 4 middle.
3. In any case, only visibility calculations based on the original geometry $\mathbf{G_O}$ can provide correct results. If only approximate geometry is available, visibility errors are bound to occur, Figure 4 right.

In summary, in the presence of even small geometric inaccuracies or camera calibration imprecisions, a linear approach is not able to correctly interpolate from discrete image samples.

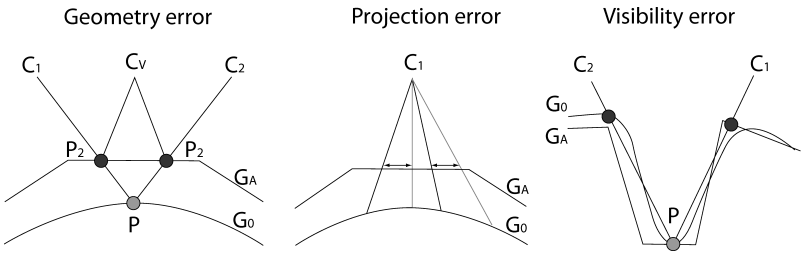

Fig. 4 Left: Geometry inaccuracies cause ghosting artifacts. Point P on the original surface $\mathbf{G_O}$ is erroneously projected to 3-D-position P_1 from camera C_1 and to 3-D-position P_2 from camera C_2 if only the approximate geometry $\mathbf{G_A}$ is available. Middle: Small imprecisions in camera calibration can lead to false pixel projections (dark gray lines, compared to correct projections displayed as light gray lines). This leads to a texture shift on the object surface and subsequently to ghosting artifacts. Right: Visibility errors. Given only approximate geometry $\mathbf{G_A}$, point P is classified as being visible from C_2 and not visible from camera C_1. Given correct geometry $\mathbf{G_O}$, it is actually the reverse, resulting in false projections.

4 Floating Textures

As pointed out in the last section an adaptive, non-linear approach is necessary for high quality texturing in the presence of imprecise geometry and undersampled input data to reduce blurring and ghosting artifacts. Assuming that a set of input images, the corresponding, possibly imprecise, calibration data and a geometry proxy are given (cf. to Section 2 for common 3-D reconstruction methods), the task is to find a way to texture this proxy again without noticeable artifacts and shadowing the imprecision of the underlying geometry.

Without occlusion, any novel viewpoint can, in theory, be rendered directly from the input images by warping, i.e. by simply deforming the images, so that the following property holds:

$$\mathbf{I}_j = \mathbf{W}_{\mathbf{I}_i \to \mathbf{I}_j} \circ \mathbf{I}_i \quad , \tag{4}$$

where $\mathbf{W}_{\mathbf{I}_i \to \mathbf{I}_j} \circ \mathbf{I}_i$ warps an image $\mathbf{I}_i$ towards $\mathbf{I}_j$ according to the warp field $\mathbf{W}_{\mathbf{I}_i \to \mathbf{I}_j}$. The problem of determining the warp field $\mathbf{W}_{\mathbf{I}_i \to \mathbf{I}_j}$ between two images $\mathbf{I}_i$, $\mathbf{I}_j$ is a heavily researched area in computer graphics and vision and several techniques exist which try to solve this problem, the most famous known are optical flow estimations, e.g. [31, 38]. If pixel distances between corresponding image features are not too large, algorithms to robustly estimate per-pixel optical flow are available [12, 45]. The issue here is that in most cases these distances will be too large.

In order to relax the correspondence finding problem, the problem can literally be projected into another space, namely the output image domain. By first projecting the photographs from cameras C_i onto the approximate geometry surface $\mathbf{G_A}$ and rendering the scene from the desired viewpoint V, creating the intermediate images $\mathbf{I}_i^V$, the correponding image features are brought much closer together than they have been in the original input images, Figure 5. This opens up the possibility of using well-established techniques like optical flow estimation to the intermediate images

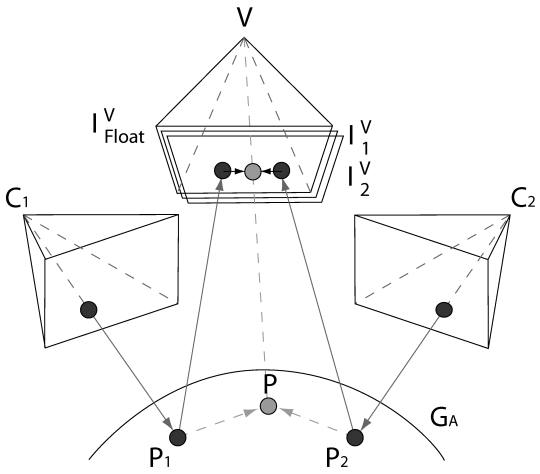

Fig. 5 Rendering with Floating Textures [20]. The input photos are projected from camera positions C_i onto the approximate geometry $\mathbf{G_A}$ and onto the desired image plane of viewpoint V. The resulting intermediate images $\mathbf{I}_i^V$ exhibit mismatch which is compensated by warping all $\mathbf{I}_i^V$ based on the optical flow to obtain the final image $\mathbf{I}_{\text{Float}}^V$.

$\mathbf{I}_i^V$ to robustly determine the pairwise flow fields $\mathbf{W}_{\mathbf{I}_i^V \to \mathbf{I}_j^V}$, i.e. to find the corresponding features in both images. To compensate for more than two input images, a linear combination of the flow fields according to (6) can be applied to all intermediate images $\mathbf{I}_i^V$, which can then be blended together to obtain the final rendering result $\mathbf{I}_{\text{Float}}^V$ [20]. To reduce computational cost, instead of establishing for n input photos $(n-1)n$ flow fields, it often suffices to consider only the 3 closest input images to the current viewpoint. If more than 3 input images are needed, the quadratic effort can be reduced to linear complexity by using intermediate results.

It is important to use an angular weighting scheme as proposed in [13, 19] because it provides smooth changes of the camera influences and therefore prevents snapping problems which could otherwise occur.

The processing steps are summarized in the following functions and visualized in Figure 5:

$$\mathbf{I}_{\text{Float}}^V = \sum_{i=1}^{n} (\mathbf{W}_{\mathbf{I}_i^V} \circ \mathbf{I}_i^V)\,\omega_i \tag{5}$$

$$\mathbf{W}_{\mathbf{I}_i^V} = \sum_{j=1}^{n} \omega_j \mathbf{W}_{\mathbf{I}_i^V \to \mathbf{I}_j^V} \tag{6}$$

$\mathbf{W}_{\mathbf{I}_i^V}$ is the combined flow field which is used for warping image $\mathbf{I}_i^V$. (5) is therefore an extension of (1) by additionally solving for the non-linear part in P.

4.1 Soft Visibility

Up to now only occlusion-free situations can be precisely handled, which is seldomly the case in real-world scenarios. Simple projection of imprecisely calibrated photos onto an approximate 3-D geometry model typically causes unsatisfactory results in the vicinity of occlusion boundaries, Figure 6 top left. Texture information from occluding parts of the mesh project incorrectly onto other geometry parts. With respect to Floating Textures, this not only affects rendering quality but also the reliability of flow field estimation.

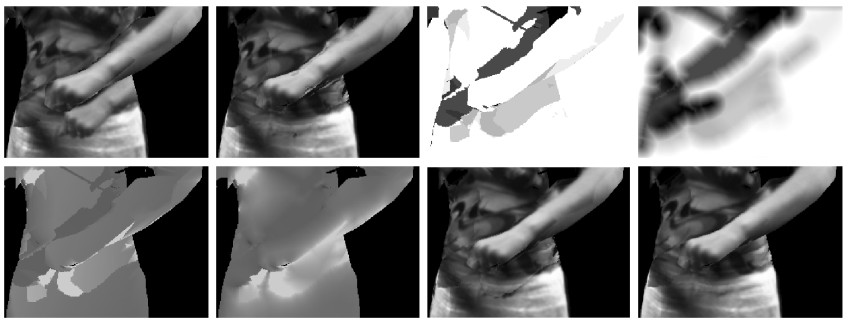

Fig. 6 Top row, left: Projection errors occur if occlusion is ignored. Middle left: Optical flow estimation goes astray if occluded image regions are not properly filled. Middle right: Visualization of a binary visibility map for three input cameras. Right: Visualization of a soft visibility map for three input cameras. Second row, left: Weight map multiplied with the binary visibility map. Middle left: Weight map multiplied with the soft visibility map; note that no sudden jumps of camera weights occur anymore between adjacent pixels. Middle right: Final result after texture projection using a weight map with binary visibility. Right: Final result after texture projection using a weight map with soft visibility. Note that most visible seams and false projections have been effectively removed.

A common approach to handle the occlusion problem is to establish a binary visibility map for each camera, multiply it with the weight map, and normalize the weights afterwards so they sum up to one. This efficiently discards occluded pixels in the input cameras for texture generation. In [15] the camera is slightly displaced several times in order to reliably detect occluded pixels. Lensch *et al.* [35] discard samples which are close to large depth changes, as they cannot be relied on. One drawback of this approach is that it must be assumed that the underlying geometry is precise, and cameras are precisely calibrated. In the presence of coarse geometry, the usage of such binary visibility maps can create occlusion boundary artifacts at pixels where the value of the visibility map suddenly changes, Figure 6 bottom row, middle right.

To counter these effects, a "soft" visibility map Ω for the current viewpoint and every input camera can be generated using a distance filter on the binary map:

$$\Omega(x,y) = \begin{cases} 0 & \text{if } \delta(x,y) = 0 \\ \frac{occDist(x,y)}{r} & \text{if } occDist(x,y) \leq r \\ 1 & \text{else} \end{cases} \tag{7}$$

Here r is a user-defined radius, and $occDist(x,y)$ is the distance to the next occluded pixel. If Ω is multiplied with the weight map, (7) makes sure that occluded regions stay occluded, while hard edges in the final weight map are removed. Using this "soft" visibility map the above mentioned occlusion artifacts effectively disappear, Figure 6 bottom right.

To improve optical flow estimation, occluded areas in the projected input images $\mathbf{I}_t^V$ need to be filled with the corresponding color values from that camera whose weight ω for this pixel is highest, as the probability that this camera provides the correct color is the highest. Otherwise, the erroneously projected part could seriously influence the result of the Floating Texture output as wrong correspondences could be established, Figure 6 top row, middle left. Applying the described filling procedure noticeably improves the quality of the flow calculation, Figure 6 bottom right.

4.2 GPU Implementation

The non-linear optimization before the blending step is computationally very intensive and cannot be sufficiently calculated in advance. Therefore for immediate feedback it is important to compute the whole rendering part on-the-fly exploiting the power of modern graphics hardware. A block diagram is given in Figure 7. The

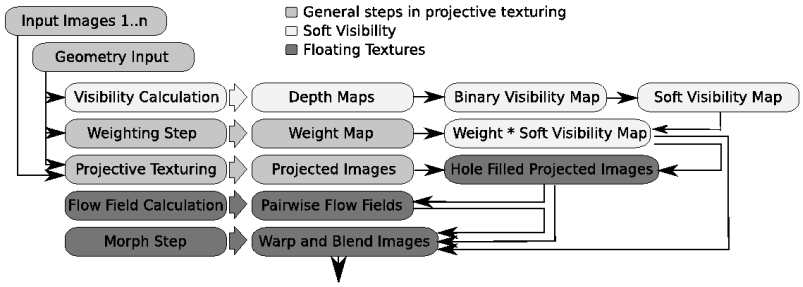

Fig. 7 Complete overview of the Floating Textures algorithm on GPU. See text for details.

geometry representation can be of almost arbitrary type, e.g., a triangle mesh, a voxel representation, or a depth map (even though correct occlusion handling with a single depth map is not always possible due to the $2.5D$ scene representation).

First, given a novel viewpoint, the closest camera positions are queried. For sparse camera arrangements, typically the two or three closest input images are chosen. The geometry model is rendered from the cameras' viewpoints into differ-

ent depth buffers. These depth maps are then used to establish for each camera a binary visibility map for the current viewpoint. These visibility maps are used as input to the soft visibility shader which can be efficiently implemented in a two-pass fragment shader. Next, a weight map is established by calculating the camera weights per output pixel, based on the Unstructured Lumigraph weighting scheme [13]. The final camera weights for each pixel in the output image are obtained by multiplying the weight map with the visbility map and normalizing the result.

To create the input images for the flow field calculation, the geometry proxy is rendered from the desired viewpoint several times into multiple render targets in turn, projecting each input photo onto the geometry. If the weight for a specific camera is 0 for a pixel, the color from the input camera with the highest weight at this position is used instead.

To compute the optical flow between two images efficient GPU implementations are needed [20, 45]. Even though this processing step is computationally expensive and takes approximately 90% of the rendering time, interactive to real-time speedups are possible with modern GPUs. Once all needed computations have been carried out, the results can be combined in a final render pass, which warps and blends the projected images according to the weight map and flow fields. The benefits of the Floating Textures approach are best visible in the images in Figure 8, where a comparison of different image-based rendering approaches is given.

4.3 Static Correspondence Finding

Under some circumstances it might be important to prewarp the textures, not for each viewpoint but once for each time step. One application in this direction would be the estimation of the BRDF of the model. Therefore reflectance information is needed for every point on a surface throughout the whole sequence.

Ahmed *et al.* [6] incorporate BRDF estimation into the free-viewpoint video system [15]. They specifically solve two problems. First, due to the underlying parameterized model a consistent image to surface correspondence for each frame must be found. This is done by reprojecting the input images onto the geometry and back into the views of the other cameras. Then they optimize the projected image for each camera to create a multi-view video texture. For every point on the surface they estimate the camera for which the surface point to the camera deviates the least from the normal vector at that position and use this projected color as reference value. They then warp the input image so that it most resembles this view.

A second registration problem is the model change over time. A parameterized model cannot directly cope with changes of the recorded object, as e.g. shifting clothes. This would invalidate the assumption that a constant set of BRDF parameters could be assigned to each location on the object. To deal with this the texture is transformed into a square domain, similar to geometry images [28] and frame to frame correspondences are computed to handle the shift.

During acquisition two recording passes are usually needed, one pass to acquire the reflectance information, where the actor needs to slowly turn himself around and one recording for the actual motion that one wants to capture. For both, calibrated light sources need to be used.

Assigning constant texture information through warping for each vertex is only possible if a mesh with consistent vertex topology is given. In many reconstruction approaches, cf. Section 2, this is not provided. Furthermore assigning constant texture coordinates to each vertex even per frame may lead to wrong results on coarse geometry. This is due to the assumption, that at least one camera projects the correct color value onto the mesh is not always true and the amount of warping must be based on the current viewpoint [20]. That means the correspondences can still be precomputed but the amount of warping during rendering must be scaled depending on the viewpoint to theoretically generate an artifact free image. In our experience the dynamic approach from Section 4 reveals more realistic results and should be preferred if no complete BRDF estimation is needed.

5 View and Time Interpolation in Image Space

Up to now we considered the case where at least an approximate geometry could be reconstructed. In some cases however it is beneficial not to reconstruct any geometry at all, but instead work solely in image space. In some sense reconstructing geometry imposes an implicit quality degradation by creating a 3-D scene from a 2-D video, for the purpose creating a 2-D video out of the 3-D scene again.

While sophisticated methods are still able to create high quality, in controlled studio environments, cf. Section 3, these methods also pose several constraints on the acquisition setup. First of all, many methods only reconstruct foreground objects, which can be easily segmented from the rest of the image. Second, the scene to be reconstructed must be either static or the recording cameras must be synchronized, so that frames are captured at exactly the same time instance, otherwise reconstruction will fail for fast moving parts. Even though it is possible to trigger synchronized capturing for modern state-of-the-art cameras, it still poses a problem in outdoor environments or for moving cameras, due to the amount of cables and connectors. Third, if automatic reconstruction fails, laborious modelling by hand might be necessary. Additionally sometimes even this approach seems infeasible due to fine, complicated structures in the image like e.g. hair.

Working in image-space directly can solve or at least ease most of the aforementioned problems for 3-D cinematography, as the problem is altered from a 3-D reconstruction problem to a 2-D correspondence problem. If perfect correspondences are found between every pixel of two or more images, morphing techniques can create the impression of a real moving camera to the human observer, plus time and space can be treated equally in a common framework. While this enforces some

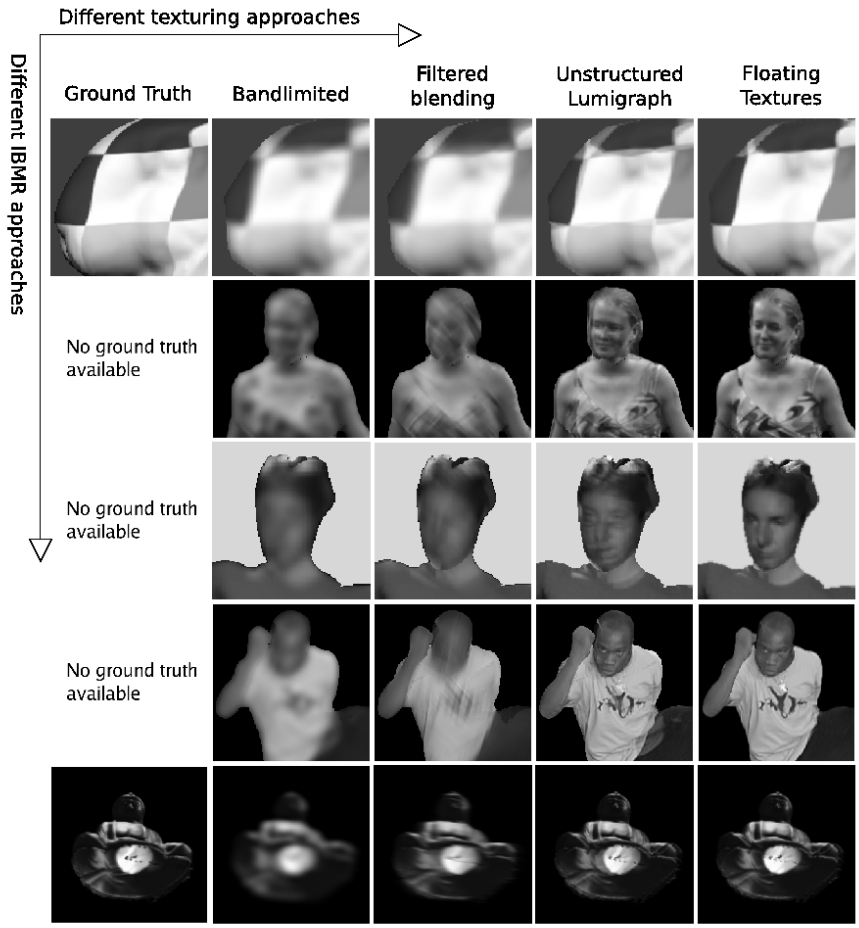

Fig. 8 Comparison of different texturing schemes in conjunction with a number of image-based modelling and rendering (IBMR) approaches. From left to right: Ground truth image (where available), bandlimited reconstruction [16], Filtered Blending [21], Unstructured Lumigraph Rendering [13], and Floating Textures. The different IBMR methods are (from top to bottom): Synthetic data set, Polyhedral Visual Hull Rendering [24], Free-Viewpoint Video [15], SurfCap [54], and Light Field Rendering [36].

constraints, as e.g. limiting the possible camera movement to the camera hull[1], see Figure 9, it also opens up new possibilities as e.g. easier acquisition and rendering of much more complex scenes. In addition rendering quality is better in many cases.

Computing the true motion field from the images alone, however, is a formidable task that, in general, is hard to solve due to inherent ambiguities. For example the

[1] This is not completely true. Extrapolation techniques could be used to go beyond this limitation, but quality will quickly prevail.

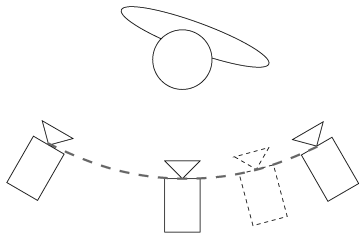

Fig. 9 Image-based interpolation techniques can create the impression of a moving camera along the space spanned by the input cameras (here depicted by the dashed line).

aperture problem and insufficient gradient strength can make it impossible to compute the correct motion field using e.g. optical flow. However, the true motion field is not needed if the goal is to generate perceptually convincing image interpolations. Because a perceptually plausible motion is interpreted as a physically correct motion by a human observer, we can rely on the capabilities of the human visual system to understand the visual input correctly in spite of all ambiguities. It is thus sufficient to focus on the aspects that are important to human motion perception to solve the interpolation problem. Or in other words:

> "The human eye does not care about optimal solutions in a least squares sense, as long as it looks good."

5.1 Image Morphing and Spatial Transformations

Image morphing aims at creating smooth transitions between pairs or arbitrary numbers of images. For simplicity of explanation we will stick to two images first. The basic procedure is to warp, i.e. to deform, the input images I_1 and I_2 towards each other depending on some warp functions $W_{I_1 \to I_2}$, $W_{I_2 \to I_1}$ and a time step α, with $\alpha \in [0,1]$ so that $\alpha W_{I_1 \to I_2} \circ I_1 = (1-\alpha) W_{I_2 \to I_1} \circ I_2$ and vice versa in the best case. This optimal warp function can usually only be approximated, so to achieve more convincing results when warping image I_1 towards I_2, one usually also computes the corresponding warp from I_2 towards I_1 and blends the results together. More mathematically formulated we can write

$$I_{1,2}(\alpha) = B((\alpha \mathbf{W}_{\mathbf{I_1} \to \mathbf{I_2}}) \circ I_1, ((1-\alpha) \mathbf{W}_{\mathbf{I_2} \to \mathbf{I_1}}) \circ I_2, \alpha) \tag{8}$$

where the blending function B is usually a simple linear cross-dissolve. We will have a more detailed look on how to implement a sophisticated warping function in Section 5.4.

5.2 Image Deformation Model for Time and View Interpolation

Analyzing properties of the human visual system shows that it is sensitive to three main aspects [59, 48, 47, 30]. These are:

1. Edges
2. Coherent motion for parts belonging to the same object
3. Motion discontinuities at object borders.

It is therefore important to pay special attention to these aspects for high-quality interpolation.

Observing our surroundings we might notice that objects in the real world are seldom completely flat, even though many man-made objects are quite flat. However they can be approximated quite well by flat structures, like planes or triangles, as long as these are small enough. Usually this limit is given by the amount of detail the eye can actually perceive. In computer graphics it is usually set by the screen resolution (you may try as hard as you wish, but details smaller than a pixel are simply not visible).

If it is assumed that the world consists of such planes, then the relation between two projections of such a 3-D plane can be directly described via a homography in image space. Such homographies for example describe the relation between a 3-D plane seen from two different cameras, the 3-D rigid motion of a plane between two points in time seen from a single camera or a combination of both. Thus, the interpolation between images depicting a dynamic 3-D plane can be achieved by a per pixel deformation according to the homography directly in image space without the need to reconstruct the underlying 3-D plane, motion and camera parameters explicitly. Only the assumption that natural images can be decomposed into regions, for which the deformation of each element is sufficiently well described by a homography has to be made, which is surprisingly often the case. Stich *et al.* [57, 56] introduced translets which are homographies that are spatially restricted. Therefore a translet is described by a 3×3 matrix H and a corresponding image segment. To obtain a dense deformation, they enforce that the set of all translets is a complete partitioning of the image and thus each pixel is part of exactly one translet, an example can be seen in Figure 10 on the bottom right. Since the deformation model is defined piecewise, it can well describe motion discontinuities as for example resulting from occlusions.

The first step in estimating the parameters of the deformation model is to find a set of point correspondences between the images from which the translet transformation can be derived. This may sound contradictive as we stated earlier that this is the overall goal. However at this stage we are not yet interested in a complete correspondence field for every pixel. Rather we are looking for a subset for which the transformation can be more reliably established and which convey already most of the important information concerning the apparent motion in the image. As it turns out classic point features such as edges and corners, which have a long history of research in computer vision, are best suited for this task. This is in accordance to the human vision, which measures edge- and corner-features early on.

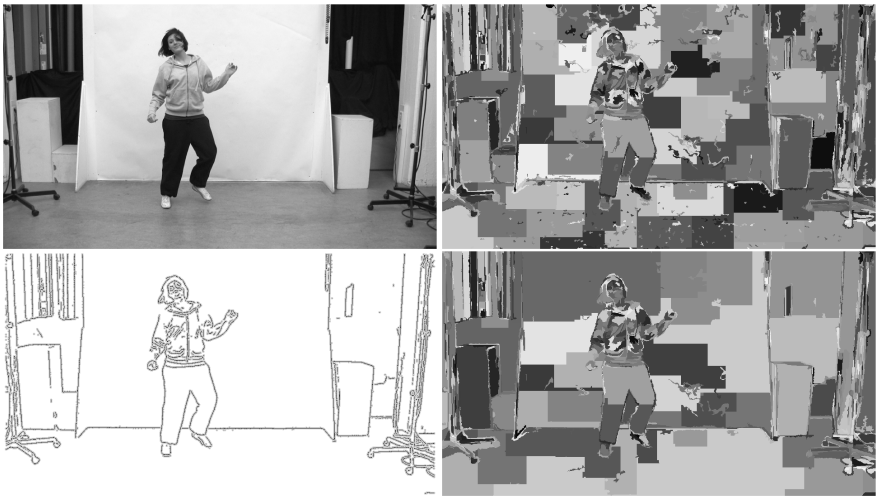

Fig. 10 An image (upper left) and its decomposition into its homogeneous regions (upper right). Since the transformation estimation is based on the matched edglets, only superpixels that contain actual edglets (lower left) are of interest. Stich *et al.* [57, 56] merge superpixels with insufficient edglets with their neighbors (lower right).

Using the Compass operator [49], a set of edge pixels, called edglets, is obtained[2]. Depending on the scene, between 2000 to 20000 pixels are edglets. Having extracted these edges in both images, the task is now to find for each edglet in image I_1 a corresponding edglet in image I_2 and this matching should be an as complete as possible 1-1 matching. This problem can be posed as a maximum weighted bipartite graph matching problem, or in other words, one does not simply assign the best match to each edglet, but instead tries to minimize an energy function to find the best overall solution. Therefore descriptors for each edglet need to be established. The shape context descriptor [9] has been shown to perform very well at capturing the spatial context C_{shape} of edgelets and is robust against the expected deformations. To reduce computational effort and increase robustness for the matching process only the k nearest neighbor edglets are considered as potential matches for each edglet. Also one can assume that edglets will not move from one end of the image I_1 to the other in image I_2 as considerable overlap is always needed to establish a reliable matching. Therefore an additional distance term C_{dist} can be added. One prerequisite for the reformulation is that for each edglet in the first set a match in the second set exists, otherwise the completeness cannot be achieved. While this is true for most edglets, some will not have a correspondence in the other set due to occlusion or small instabilities of the edge detector at faint edges. However, this is easily addressed by inserting virtual occluder edglets for each edglet in the first edglet set. The graph for the matching problem is then build as depicted in Figure 11. Each edge pixel of the first image is connected by a weighted edge to its possibly cor-

[2] Other edge detectors could be used for this step as well, as the Canny operator [14]

responding edge pixels in the second image and additionally to its virtual occluder edglet. The weight or cost function for edglet $\mathbf{e}_i$ in I_1 and $\mathbf{e}'_j$ in I_2 is then defined as

$$C(\mathbf{e_i}, \mathbf{e}'_j) = C_{dist} + C_{shape} \tag{9}$$

where the cost for the shape is the χ^2-test between the two shape contexts and the cost for the distance is defined as

$$C_{dist}(\mathbf{e_i}, \mathbf{e}'_j) = \frac{a}{(1 + e^{-b\,||\mathbf{e}_i - \mathbf{e}'_j||})} \tag{10}$$

with $a, b > 0$ such that the maximal cost for the euclidean distance is limited by a. The cost $C_{occluded}$ to assign an edglet to its occluder edglet is user defined and controls how aggressively the algorithm tries to find a match with an edglet of the second image. The lower $C_{occluded}$ the more conservative the resulting matching will be, as more edges will be matched to their virtual occluder edglets.

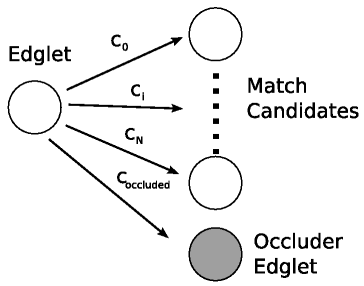

Fig. 11 Subgraph of the weighted bipartite graph matching problem for a single edglet. Each edglet has an edge to its possible match candidates and an additional edge to its virtual occluder edglet.

Now that the first reliable matches have been found this information can be used to find good homographies for the translets of both images. But first the spatial support for these translets need to be established, i.e. the image needs to be segmented into coherent, disjoint regions. From Gestalt theory [61] it is known that for natural scenes, these regions share not only a common motion but in general also share other properties such as similar color and texture. Felzenszwalb and Huttenlocher's superpixel segmentation [22] can be exploited to find an initial partitioning of the image into regions to become translets, based on neighboring pixel similarities. Then from the matching between the edge pixels of the input images, local homographies for each set of edge pixels in the source image that are within one superpixel are estimated. In order to do this four reliable point correspondences need to be found to compute the homography. Since the least-squares estimation based on all matched edglets of a translet is sensitive to outliers and often more than the minimal number of four matched edge pixels is available, a RANSAC approach to obtain a robust solution and filter match outliers is preferred instead [23]. Usually still between 20% to 40% of the computed matches are outliers and thus some translets will have wrongly estimated transformations. Using a greedy iterative approach, the most similar transformed neighboring translets are merged into one, as depicted in

Figure 12, until the ratio of outliers to inliers is lower than a user defined threshold. When two translets are merged, the resulting translet then contains both edglet sets and has the combined spatial support. The homographies are re-estimated based on the new edglet set and the influence of the outliers is again reduced by the RANSAC filtering.

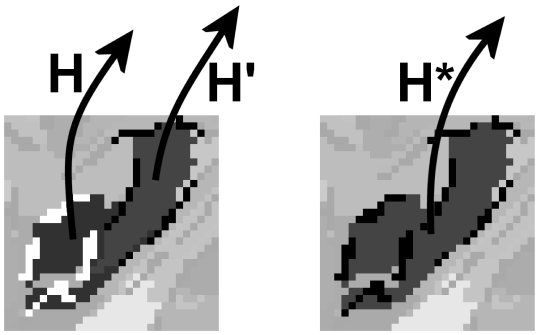

Fig. 12 During optimization, similar transformed neighboring translets are merged into a single translet. After merging, the resulting translet consists of the combined spatial support of both initial translets (mid and dark gray) and their edglets (black and white).

Basically in this last step a transformation for each pixel in the input images towards the other image was established. Assuming linear motion only, the deformation vector $d(\mathbf{x})$ for a pixel $\mathbf{x}$ is thus computed as

$$d(\mathbf{x}) = H_t\mathbf{x} - \mathbf{x}. \tag{11}$$

H_t is the homography matrix of the translet t with $\mathbf{x}$ being part of the spatial support of t. However, when only a part of a translet boundary is at a true motion discontinuity, noticeably incorrect discontinuities still produce artifacts along the rest of the boundary. Imagine for example the motion of an arm in front of the body. It is discontinuous along the silhouette of the arm, while the motion at the shoulder changes continuously. We can then resolve the per pixel smoothing by an anisotropic diffusion [44] on this vector field using the diffusion equation

$$\delta I/dt = div(\, g(min(|\nabla d|, |\nabla I|)\, \nabla I) \tag{12}$$

which is dependent on the image gradient ∇I and the gradient of the deformation vector field ∇d. The function g is a simple mapping function as defined in [44]. Thus, the deformation vector field is smoothed in regions that have similar color or similar deformation, while discontinuities that are both present in the color image and the vector field are preserved. During the anisotropic diffusion, edglets that have an inlier match, meaning they are only slightly deviating from the planar model, are considered as boundary conditions of the diffusion process. This results in exact

edge transformations handling also non-linear deformations for each translet and significantly improves the achieved quality.

5.3 Optimizing the Image Deformation Model

There are three ways to further optimize the image deformation model from the previous section.

1. using motion priors
2. using coarse-to-fine translet estimation
3. using a scale-space hierarchy

Since the matching energy function (Eq. 9) is based on spatial proximity and local geometric similarity, a motion prior can be introduced by pre-warping the edglets with a given deformation field. The estimated dense correspondences described above can be used as such a prior. So the algorithm described in Section 5.2 can be iterated using the result from the i-th iteration as the input to the $(i+1)$-th iteration.

To overcome local matching minima a coarse to fine iterative approach on the translets can be applied. In the first iteration, the number of translets is reduced until the coarsest possible deformation model with only one translet is obtained. Thus the underlying motion is approximated by a single perspective transformation. During consecutive iterations, the threshold is decreased to allow for more accurate deformations as the number of final translets increases.

Additionally, solving on different image resolutions similar to scale-space [63] further improves robustness. Thus a first matching solution is found on the coarse resolution images and is then propagated to higher resolutions. Using previous solutions as motion prior significantly reduces the risk to getting stuck in local matching minima, cf. Figure 13.

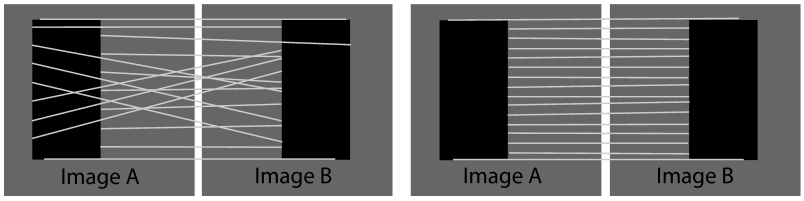

Fig. 13 Local matching minima (left) can be avoided by multiple iterations. In a coarse to fine manner, in each iterations the number of translets increases avoiding local matching minima by using the previous result as prior (right).

In rare cases, some scenes still cannot be matched automatically sufficiently well. For example, when similar structures appear multiple times in the images the matching can get ambiguous and can only be addressed by high level reasoning. To resolve this, a fallback on manual intervention is necessary. Regions can be selected in both

images by the user and the automatic matching is computed again only for the so selected subset of edglets. Due to this restriction of the matching, the correct match is found and used to correct the solution.

5.4 Rendering

Given the pixel-wise displacements from the previous sections the rendering can then be efficiently implemented on graphics hardware to allow for real-time image interpolation. Therefore a regular triangle mesh is placed over the image plane, so that each pixel in the image is represented by two triangles with appropriate texture coordinates. A basic morphing scheme algorithm as presented in Equation (8) would be straight-forward to implement by just displacing the vertices in the vertex shader by the scaled amount of the corresponding displacement vector according to the α-value chosen. However, two problems arise with forward warping at motion discontinuities: Fold-overs and missing regions. Fold-overs occur when two or more pixels in the image end up in the same position during warping. This is the case when the foreground occludes parts of the background. Consistent with motion parallax it is assumed that the faster moving pixel in x-direction is closer to the viewpoint to resolve this conflict. When on the other hand regions get disoccluded during warping the information of these regions is missing in the image and must be filled in from the other image. Mark *et al.* [39] proposed to use a connectedness criterion evaluated on a per-pixel basis after warping. This measure can be computed directly from the divergence of the deformation vector field such that

$$c_{I_1} = 1 - div(d_{I_1 \to I_2})^2. \tag{13}$$

with c_{I_1} being the connectedness and $d_{I_1 \to I_2}$ is the vector field between the images I_1 and I_2 (cf. Figure 14). The connectedness is computed on the GPU during blending to adaptively reduce the alpha values of pixels with low connectedness. Thus, in missing regions only the image which has the local information has an influence on the rendering result.

Opposed to recordings with cameras, rendered pixels at motion boundaries are no longer a mixture of background and foreground color but are either foreground or background color. In a second rendering pass, the color mixing of foreground and background at boundaries can be modelled using a small selective low-pass filter applied only to the detected motion boundary pixels. This effectively removes the artifacts with a minimal impact on rendering speed and without affecting rendering quality in the non-discontinuous regions.

The complete interpolation between two images I_1 and I_2 can then be described as

$$I(\alpha) = \frac{c_{I_1}(1-\alpha)(\alpha d_{I_1 \to I_2} \circ I_1) + c_{I_2}(\alpha)((1-\alpha) d_{I_2 \to I_1} \circ I_2)}{c_{I_1}(1-\alpha) + c_{I_2}(\alpha)} \tag{14}$$

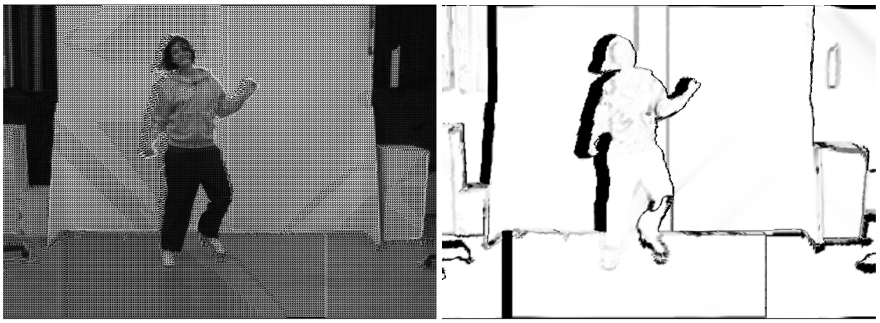

Fig. 14 Left: Per-vertex mesh deformation is used to compute the forward warping of the image, where each pixel corresponds to a vertex in the mesh. The depicted mesh is at a coarser resolution for visualization purposes. Right: The connectedness of each pixel that is used during blending to avoid a possibly incorrect influence of missing regions.

where $c_X(\phi)$ is the locally varying influence of each image on the final result which is modulated by the connectedness

$$c_X(\alpha) = c_X \cdot \alpha \tag{15}$$

Thus, the (possibly incorrect) influence of pixels with low connectedness on the final result is reduced.

The interpolation is not restricted to two images. Interpolating between multiple images is achieved by iteratively repeating the warping and blending as described in (14), where I takes over the role of one of the warped images in the equation. To stay inside the image manifold that is spanned by the images the interpolation factors must sum to one, $\sum_i \alpha_i = 1$.

As can be seen in Table 1 the proposed algorithm produces high-quality results, e.g. using the Middlebury examples [8].

The results have been obtained without user interaction. As can be seen the approach is best when looking at the interpolation errors and best or up to par in the sense of the normalized interpolation error. It is important to point out that from a perception point of view the normalized error is less expressive than the unnormalized error since discrepancies at edges in the image (e.g. large gradients) are dampened. Interestingly, relatively large angular errors are observed with the presented method emphasizing that the requirements of optical flow estimation and image interpolation are different.

6 Acknowledgements

We would like to thank Jonathan Starck for providing us with the SurfCap test data (www.ee.surrey.ac.uk/CVSSP/VMRG/surfcap.htm) and the Stanford Computer Graphics lab for the buddha light field data set.

Venus	Interp.	Norm. Interp.	Ang.
Stich et al.	**2.88**	**0.55**	**16.24**
Pyramid LK	3.67	0.64	14.61
Bruhn et al.	3.73	0.63	8.73
Black and Anandan	3.93	0.64	7.64
Mediaplayer	4.54	0.74	15.48
Zitnick et al.	5.33	0.76	11.42

Dimetrodon	Interp.	Norm. Interp.	Ang.
Stich et al.	**1.78**	**0.62**	**26.36**
Pyramid LK	2.49	0.62	10.27
Bruhn et al.	2.59	0.63	10.99
Black and Anandan	2.56	0.62	9.26
Mediaplayer	2.68	0.63	15.82
Zitnick et al.	3.06	0.67	30.10

Hydrangea	Interp.	Norm. Interp.	Ang.
Stich et al.	**2.57**	**0.48**	**12.39**

RubberWhale	Interp.	Norm. Interp.	Ang.
Stich et al.	**1.59**	**0.40**	**23.58**

Table 1 Interpolation, Normalized Interpolation and Angular errors computed on the Middlebury Optical Flow examples by comparison to ground truth with results obtained by our method and by other methods taken from Baker *et al.* [8].

References

1. Adelson, E.H., Bergen, J.R.: The Plenoptic Function and the Elements of Early Vision. Computational Models of Visual Processing pp. 3–20 (1991)
2. de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.P., Thrun, S.: Performance capture from sparse multi-view video. ACM Transactions on Graphics **27**(3), 1–10 (2008)
3. de Aguiar, E., Theobalt, C., Magnor, M., Seidel, H.P.: Reconstructing human shape and motion from multi-view video. In: European Conference on Visual Media Production, pp. 42–49 (2005)
4. de Aguiar, E., Theobalt, C., Stoll, C., Seidel, H.P.: Marker-less deformable mesh tracking for human shape and motion capture. In: International Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
5. de Aguiar, E., Theobalt, C., Stoll, C., Seidel, H.P.: Rapid animation of laser-scanned humans. In: Virtual Reality, pp. 223–226 (2007)
6. Ahmed, N., Theobalt, C., Magnor, M.A., Seidel, H.P.: Spatio-temporal registration techniques for relightable 3d video. In: International Conference on Image Processing, pp. 501–504 (2007)
7. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: SCAPE: shape completion and animation of people. ACM Transactions on Graphics **24**(3), 408–416 (2005)
8. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: A Database and Evaluation Methodology for Optical Flow. In: International Conference on Computer Vision, pp. 1–8 (2007)
9. Belongie, S., Malik, J., Puzicha, J.: Matching Shapes. In: International Conference on Computer Vision, pp. 454 – 461 (2001)
10. Bhat, P., Zitnick, C.L., Snavely, N., Agarwala, A., Agrawala, M., Curless, B., Cohen, M., Kang, S.B.: Using photographs to enhance videos of a static scene. In: J. Kautz, S. Pattanaik (eds.) Eurographics Symposium on Rendering, pp. 327–338. Eurographics (2007)

11. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. Transactions on Pattern Analysis and Machine Intelligence **23**(11), 1222–1239 (2001)
12. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: European Conference on Computer Vision, pp. 25–36 (2004)
13. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured Lumigraph Rendering. ACM Transactions on Graphics **20**(3), 425–432 (2001)
14. Canny, J.: A Computational Approach To Edge Detection. Transactions on Pattern Analysis and Machine Intelligence **8**, 679–714 (1986)
15. Carranza, J., Theobalt, C., Magnor, M., Seidel, H.P.: Free-viewpoint video of human actors. ACM Transactions on Graphics **22**(3), 569–577 (2003)
16. Chai, J.X., Chan, S.C., Shum, H.Y., Tong, X.: Plenoptic Sampling. ACM Transactions on Graphics **19**(3), 307–318 (2000)
17. Cobzaş, D., Yerex, K., Jägersand, M.: Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. Computer Graphics Forum **21**(3), 493–502 (2002)
18. Collins, R.T.: A space-sweep approach to true multi-image matching. In: Conference on Computer Vision and Pattern Recognition, pp. 358–363 (1996)
19. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. ACM Transactions on Graphics **15**(3), 11–20 (1996)
20. Eisemann, M., Decker, B.D., Magnor, M., Bekaert, P., de Aguiar, E., Ahmed, N., Theobalt, C., Sellent, A.: Floating Textures. Computer Graphics Forum **27**(2), 409–418 (2008)
21. Eisemann, M., Sellent, A., Magnor, M.: Filtered Blending: A new, minimal Reconstruction Filter for Ghosting-Free Projective Texturing with Multiple Images. Vision, Modeling, and Visualization pp. 119–126 (2007)
22. Felzenszwalb, P., Huttenlocher, D.: Efficient Graph-Based Image Segmentation. International Journal of Computer Vision **59**, 167–181 (2004)
23. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
24. Franco, J.S., Boyer, E.: Exact polyhedral visual hulls. In: British Machine Vision Conference, pp. 329–338 (2003). Norwich, UK
25. Fujii, T., Tanimoto, M.: Free viewpoint TV system based on ray-space representation. In: SPIE, vol. 4864, pp. 175–189. SPIE (2002)
26. Gallup, D., Frahm, J.M., Mordohai, P., Yang, Q., Pollefeys, M.: Real-time plane-sweeping stereo with multiple sweeping directions. Computer Vision and Pattern Recognition pp. 1–8 (2007)
27. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The Lumigraph. ACM Transactions on Graphics **15**(3), 43–54 (1996)
28. Gu, X., Gortler, S., Hoppe, H.: Geometry images. ACM Transactions on Graphics **21**(3), 355–361 (2002)
29. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2 edn. Cambridge University Press (2003)
30. Heeger, D., Boynton, G., Demb, J., Seidemann, E., Newsome, W.: Motion opponency in visual cortex. Journal of Neuroscience **19**, 7162–7174 (1999)
31. Horn, B., Schunck, B.: Determining Optical Flow. Artificial Intelligence **17**, 185–203 (1981)
32. Kutulakos, K.N., Seitz, S.M.: A Theory of Shape by Space Carving. International Journal on Computer Vision **38**(3), 199–218 (2000)
33. Lanman, D., Crispell, D., Taubin, G.: Surround structured lighting for full object scanning. In: International Conference on 3-D Digital Imaging and Modeling, pp. 107–116 (2007)
34. Laurentini, A.: The visual hull concept for silhouette-based image understanding. Transactions on Pattern Analysis and Machine Intelligence **16**(2), 150–162 (1994)
35. Lensch, H.P.A., Kautz, J., Goesele, M., Heidrich, W., Seidel, H.P.: Image-based reconstruction of spatial appearance and geometric detail. ACM Transactions on Graphics **22**(2), 234–257 (2003)

36. Levoy, M., Hanrahan, P.: Light Field Rendering. ACM Transactions on Graphics **15**(3), 31–42 (1996)
37. Li, M., Magnor, M., Seidel, H.P.: Hardware-accelerated rendering of photo hulls. Computer Graphics Forum **23**(3), 635–642 (2004)
38. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: International Joint Conference on Artificial Intelligence, pp. 674–679 (1981)
39. Mark, W., McMillan, L., Bishop, G.: Post-Rendering 3D Warping. In: Symposium on Interactive 3D Graphics, pp. 7–16 (1997)
40. Matsuyama, T., Wu, X., Takai, T., Nobuhara, S.: Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video. Computer Vision and Image Understanding **96**(3), 393–434 (2004)
41. Matusik, W., Buehler, C., Raskar, R., Gortler, S.J., Mcmillan, L.: Image-based visual hulls. ACM Transactions on Graphics **19**(3), 369–374 (2000)
42. Matusik, W., Pfister, H.: 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. ACM Transactions on Graphics **19**(3), 814–824 (2004)
43. Naemura, T., Tago, J., Harashima, H.: Real-time video-based modeling and rendering of 3d scenes. Computer Graphics and Applications **22**(2), 66–73 (2002)
44. Perona, P., Malik, J.: Scale-Space and Edge Detection using Anisotropic Diffusion. Transactions on Pattern Analysis and Machine Intelligence **12**(7), 629–639 (1990)
45. Pock, T., Urschler, M., Zach, C., Beichel, R., Bischof, H.: A duality based algorithm for tv-l1-optical-flow image registration. In: International Conference on Medical Image Computing and Computer Assisted Intervention, pp. 511–518 (2007)
46. Posdamer, J., Altschuler, M.: Surface measurement by space-encoded projected beam systems. Computer Graphics and Image Processing **18**(1), 1–17 (1982)
47. Qian, N., Andersen, R.: A physiological model for motion-stereo integration and a unified explanation of Pulfrich-like phenomena. Vision Research **37**, 1683–1698 (1997)
48. Reichardt, W.: Autocorrelation, A principle for the evaluation of sensory information by the central nervous system. In: W. Rosenblith (ed.) Sensory communication, pp. 303–317. New York: MIT Press-Willey (1961)
49. Ruzon, M., Tomasi, C.: Color Edge Detection with the Compass Operator. In: Conference on Computer Vision and Pattern Recognition, pp. 160–166 (1999)
50. Salvi, J., Pags, J., Batlle, J.: Pattern codification strategies in structured light systems. Pattern Recognition **37**, 827–849 (2004)
51. Segal, M., Korobkin, C., van Widenfelt, R., Foran, J., Haeberli, P.: Fast Shadows and Lighting Effects using Texture Mapping. ACM Transactions on Graphics **11**(3), 249–252 (1992)
52. Seitz, S.M., Dyer, C.R.: Photorealistic scene reconstruction by voxel coloring. In: International Journal of Computer Vision, vol. 35, pp. 1067–1073 (1997)
53. Snavely, N., Seitz, S., Szeliski, R.: Photo Tourism: Exploring Photo Collections in 3D. ACM Transactions on Graphics **25**(3), 835–846 (2006)
54. Starck, J., Hilton, A.: Surface capture for performance based animation. Computer Graphics and Applications **27**(3), 21–31 (2007)
55. Stewart, J., Yu, J., Gortler, S.J., McMillan, L.: A New Reconstruction Filter for Undersampled Light Fields. In: Eurographics Workshop on Rendering, pp. 150–156 (2003)
56. Stich, T., Linz, C., Albuquerque, G., Magnor, M.: View and Time Interpolation in Image Space. Computer Graphics Forum **27**(7), 1781–1787 (2008)
57. Stich, T., Linz, C., Wallraven, C., Cunningham, D., Magnor, M.: Perception-motivated Interpolation of Image Sequences. In: Symposium on Applied Perception in Graphics and Visualization, pp. 97–106 (2008)
58. Tsai, R.: An efficient and accurate camera calibration technique for 3d machine vision. In: Conference on Computer Vision and Pattern Recognition, pp. 364–374 (1986)
59. Wallach, H.: Über visuell wahrgenommene Bewegungsrichtung. Psychologische Forschung **20**, 325–380 (1935)
60. Waschbüsch, M., Würmlin, S., Gross, M.: 3D Video Billboard Clouds. Computer Graphics Forum **26**(3), 561–569 (2007)

61. Wertheimer, M.: Laws of organization in perceptual forms. In: W. Ellis (ed.) A Source Book of Gestalt Psychology, pp. 71–88. Kegan Paul, Trench, Trubner & Co. Ltd. (1938)
62. Yang, R., Pollefeys, M.: Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In: Conference on Computer Vision and Pattern Recognition, pp. 211–217 (2003)
63. Yuille, A.L., Poggio, T.A.: Scaling theorems for zero crossings. Transactions on Pattern Analyis and Machine Intelligence **8**(1), 15–25 (1986)
64. Zitnick, C., Kang, S., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. ACM Transactions on Graphics **23**(3), 600–608 (2004)