Plane Sweeping in Eye-gaze Corrected, Tele-immersive 3D Video Conferencing

Maarten Dumont, Patrik Goorts, Gauthier Lafruit

Abstract A tele-immersive video conferencing system for autostereoscopic 3D displays is presented. Eye contact between participants is restored by synthesizing novel, interpolated views from multiple surrounding cameras, effectively emulating a capturing camera position behind a virtually transparent display. Non-uniform, adaptive plane sweeping with dynamic workload balancing yields real-time performances on low-cost embedded GPU platforms.

1 Introduction

Imagine a world with perfect immersive teleconferencing, where people are able to communicate remotely with an intense sense of human awareness that would be virtually indistinguishable from real-life social communication. People would be able to communicate seamlessly with their friends and family thousands of miles away. Face-to-face meetings would be enabled without the need to travel, and tele-collaboration would reach the holy grail of *The Office Of The Future* [22] with a high level of immersion.

And yet, in spite of all the work already done in immersive teleconferencing and the high level of quality reached in today's video capturing and rendering technology, this ultimate dream remains unrealized. What are the potential causes and how can technology provide a solution to this problem?

Maarten Dumont, e-mail: maarten.dumont@uhasselt.be
Patrik Goorts, e-mail: patrik.goorts@uhasselt.be
Gauthier Lafruit, e-mail: gauthier.lafruit@uhasselt.be

Hasselt University - tUL - iMinds
Expertise Centre for Digital Media
Wetenschapspark 2
3590 Diepenbeek, Belgium

Gaze awareness and stereoscopic perception are actually important factors to provide a high level of realism needed to feel oneself immersed in a natural social environment [33]. Unfortunately, since cameras and displays cannot possibly occupy the same spatial position simultaneously, video conferencing participants are unable to look each other in the eyes: a person who stares at the display will be captured by the cameras as looking away at a slightly diverging angle.

An elegant way to solve this problem is to synthesize a virtual view in between the surrounding camera views, as if it would be rendered by a camera positioned right behind the screen, effectively restoring the correct head position and eye contact [25], as shown in Figure 1.

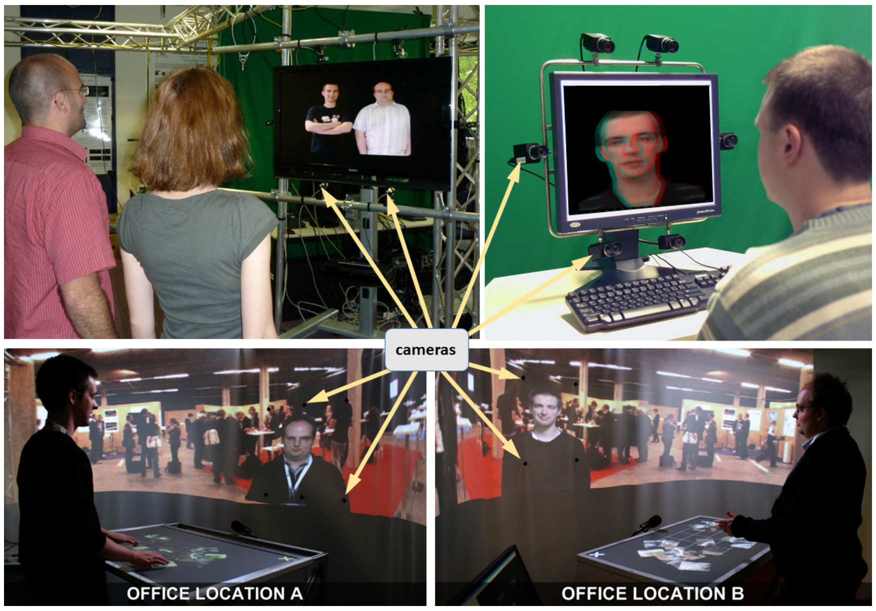


Fig. 1 Restoring eye contact by synthesizing a virtual view in between the surrounding camera views.

Synthesizing a multitude of such nearby views even supports stereoscopic (Figure 1, top right anaglyph) and glasses-free auto-multiscopic 3D displays (Figure 2), where tens of nearby virtual views are projected in different directions, allowing the viewer's eyes to capture two parallax-correct images at any position in space for a natural 3D perception.

This chapter presents a robust method for multi-camera view synthesis for eye gaze correction and natural 3D video rendering, based on seminal work in plane sweeping [7, 8, 13, 14]. Important improvements are proposed to target embedded vision applications on GPU-accelerated platforms.

Fig. 2 A plethora of multi-camera and lenticular display technology, providing all means to natural eye-gaze corrected and 3D video communication, is available on the market today. In auto-multiscopic 3D displays, tens of nearby virtual views are projected in different directions, so that the viewer's eyes always capture two parallax-correct images at any position in space, hence providing glasses-free, natural 3D perception.

2 View Synthesis Prior Art

Early solutions in handling the problem of eye gaze correction in video conferencing applied model-based approaches [28, 31] using a detailed head 3D model, which is projected in the virtual viewpoint direction. This solution often lacks in natural impression with participants talking to humanoid-looking avatars.

More advanced techniques therefore focus on image-based rendering approaches (IBR) [1, 4], where a virtual view is synthesized with depth-based image warping and inpainting techniques [20, 26]. Depth is hereby often recovered from stereo matching [24] on a pair of sanline rectified images. The limited amount of 3D scene parallax and associated depth information often results in visual artifacts in the synthesized views. Solutions such as the ones of Schreer et al. [25] and Baker et al. [2] overcome this limitation at the cost of using expensive dedicated hardware and/or unpractical camera setups.

On the other hand, plane sweeping is a method that uses a multitude of cameras and that by design is much more robust to camera illumination mismatches, misalignments, etc. It recovers scene depth by sweeping over multiple depth plane hypotheses for each pixel in the camera view that is to be reconstructed, hence its name *plane sweeping*. In principle, plane sweeping does not need to explicitly extract depth for handling view synthesis, though depth extraction helps in image post-processing for boosting the natural perception of the virtual views.

Let's first introduce the high-level concepts of these techniques, in order to better understand the advantages of our full system prototype in section 3.

2.1 Stereo Matching

Stereo matching uses a pair of images to estimate the apparent movement of the pixels from one image to the next. This apparent movement is more specifically know as the parallax effect as demonstrated in Figure 3, where two objects are shown, placed at different depths in front of a stereo pair of cameras.

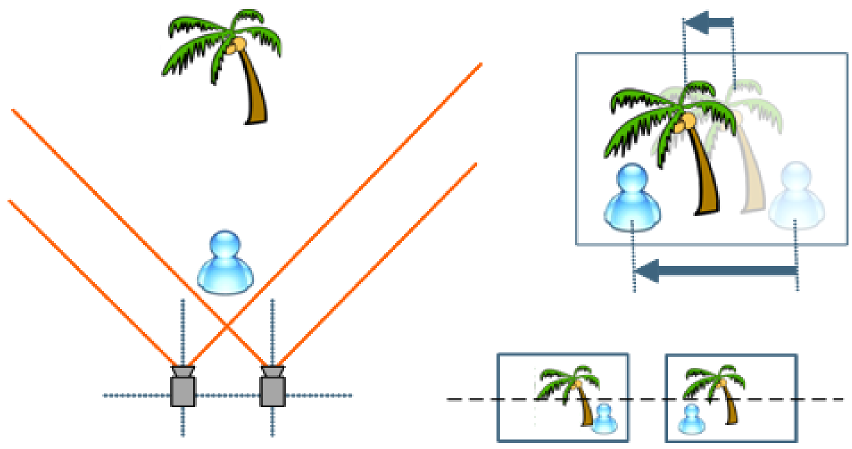


Fig. 3 Concept of stereo vision. A scene is captured using 2 rectified cameras. Stereo matching attempts to estimate the apparent movement of the objects across the images. A large apparent movement (i.e. parallax) corresponds to close objects (a low depth value).

When moving from the left to the right camera view, an object undergoes a displacement – called the disparity – which is inversely proportional to the object's depth in the scene. Objects in the background (the palm tree) has a smaller disparity in comparison to objects in the foreground (the blue buddy). The goal of stereo matching is to compute a dense disparity map by estimating each pixel's displacement.

2.2 Plane Sweeping

To conceptually grasp the concept of plane sweeping, let us take a look at Figure 4. Here cameras C_1 to C_3 are real cameras in an arbitrary configuration, whereas camera C_v is the virtual camera view of which we wish to reconstruct the color image and the scene depth.

Any voxel f in 3D space at e.g. depth plane D_1 is projected onto the 2D pixels p_i on the image plane of the respective camera views. Conversely, inversely projecting (i.e. deprojecting) the pixels p_i into 3D space will have them match at one single voxel f at the corresponding depth plane D_1. On all other depth planes D_j ($j \neq 1$) the pixels p_i deproject on points f_i that do not coincide, as illustrated on depth plane D_2. Visually, reprojecting these points f_i back into the virtual camera view C_v causes a non-focused ghosting artifact in the resulting image, as can be observed in the projected images I_{D2}.

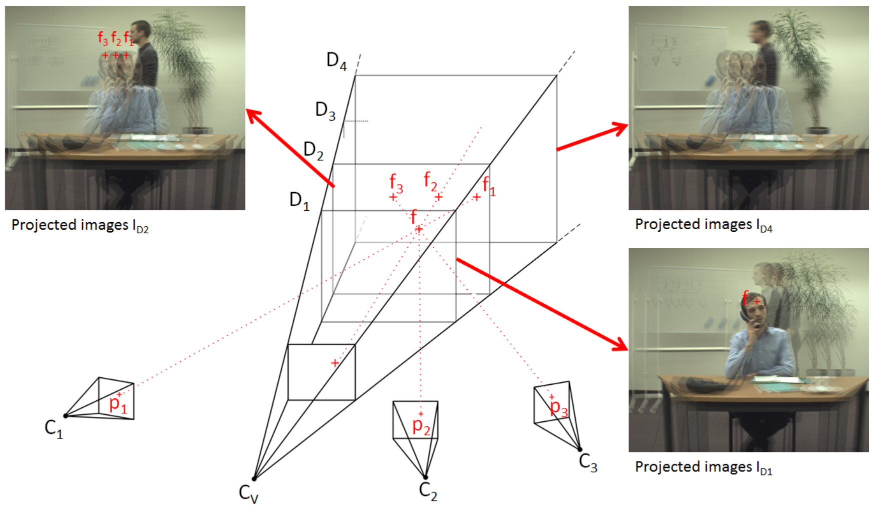

Projected images I_{D2}
Projected images I_{D4}
Projected images I_{D1}
D_4
D_3
D_2
D_1
f_3 f_2 f_1
f
p_1
p_2
p_3
C_1
C_V
C_2
C_3

Fig. 4 Plane sweeping conceptually: deprojecting real cameras C_i ($i = 1 \ldots 3$) on different depth planes D_i for virtual camera C_v causes a non-focused ghosting artifact, depending on whether or not the scene object in question is present at depth D_i.

For instance, for the two-person scene of Figure 4, the person answering the phone at the desk in the foreground is in focus at depth D_1 (as can be seen in the projected images I_{D1}), whereas the person walking by in the background is out of focus at the same hypothesized depth D_1. The background person in turn is in focus on depth plane D_2 (as can be seen in the projected images I_{D2}), hence suggesting that his corresponding voxels are indeed at depth D_2. Looking even deeper into the scene, the whiteboard in the far background is de- and reprojected in focus at its corresponding depth plane, say D_4, and is now in fact readable (projected images I_{D4}).

3 A System Prototype

We present a fully functional prototype that corrects the eye gaze of the video-conferencing peers by using multiple cameras, using the plane sweeping method reviewed in the previous section. The proposed six-fold camera setup is easily integrated into the monitor frame of Figure 1 [7, 8].

Our software framework harnesses the powerful computational resources inside the Graphics Processing Unit (GPU), achieving over real-time performance for Full HD resolution images. Furthermore, although depicted as such in Figure 4, the distribution of the depth planes is not required to be uniform, which we elaborate on in section 4 for further computational complexity savings.

In comparison, competitive solutions such as the system of Criminisi et al. [6] implement their framework on commodity CPUs, resulting in a very low frame rate when sufficient visual quality is required. Others optimize only parts of the application, such as multi-camera video coding [5, 16] for efficient data communication and real-time view synthesis [9, 21, 29] on graphics hardware, but neither of them integrate and optimize the end-to-end performance for eye gaze-corrected video chat.

The core functionality of our system is visualized in Figure 5 and consists out of five consecutive processing modules that are completely running on the GPU. In an initial step (section 3.1), the camera sensor Bayer patterns $\iota_1, \ldots, \iota_N$ are captured from a total of N cameras $C_1, \ldots, C_N$ that are fixed on a custom built metal frame which closely surrounds the screen (see Figure 1). The first module computes the RGB-images $I_1, \ldots, I_N$, based on the method of Malvar et al. [19] [12], and performs lens correction and image segmentation, as a form of preprocessing. The preprocessing module is specifically designed to enhance both the quality and speed of the consecutive view interpolation, and to ensure a high arithmetic intensity in the overall performance.

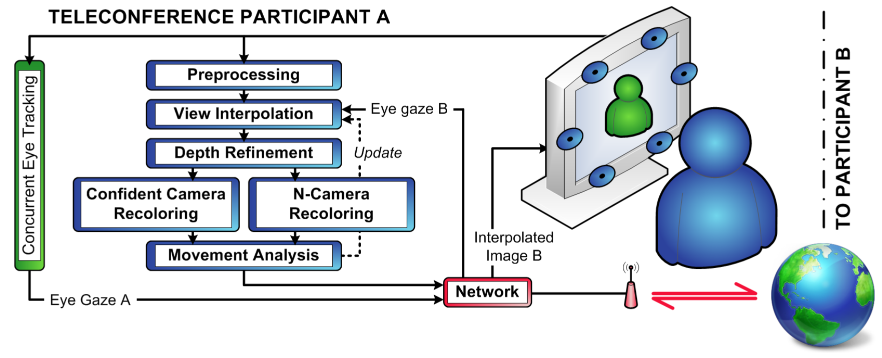


Fig. 5 Data flow and overview of our system architecture.

The second module (section 3.2) interpolates an image I_v, as it would be seen with a virtual camera C_v that is positioned behind the screen. The image I_v is computed as if camera C_v captures the image through a completely transparent screen. Furthermore, the view interpolation module produces a joint depth map Z_v, providing dense 3D information of the captured scene.

The synthesized image still contains a number of noticeable artifacts in the form of erroneous patches and speckle noise. The third module (section 3.3) is therefore specifically designed to tackle these problems by detecting photometric outliers based on the generated depth map.

Following the depth refinement, the image pixels are recolored using the filtered depth information (section 3.4). Our system currently supports recoloring of the synthesized image using all N cameras, or selecting the color from the camera which has the highest confidence of accurately capturing the required pixel.

In a final step, the depth map Z_v is also analyzed to dynamically adjust the system and thereby avoiding heavy constraints on the user's movements. This optimization is performed in the plane distribution control module (*movement analysis*) and, as previously mentioned, discussed in its dedicated section 4.

Besides the main processing on the graphics hardware that synthesizes I_v, the virtual camera C_v needs to be correctly positioned to restore eye contact between the participants. An eye tracking module (section 3.5) thereby concurrently runs on CPU and determines the user's eye position that will be used for correct placement of the virtual camera at the other peer.

By sending the eye coordinates to the other peer, the input images $\iota_1, \ldots, \iota_N$ do not have to be sent over the network (section 3.6), but can be processed at the local peer. This results in a minimum amount of required data communication – i.e. the eye coordinates and the interpolated image – between the two participants.

3.1 Preprocessing

Our system inputs Bayer patterns $\iota_1, \ldots, \iota_N$, i.e. the direct camera sensor inputs (see Figure 6(a)). The RGB-colored images $I_1, \ldots, I_N$ are consistently computed by using the method of Malvar et al. [19], which is based on linear FIR filtering. This is depicted in Figure 6(b). Uncontrolled processing that would normally be integrated into the camera electronics is therefore avoided, guaranteeing the system's optimal performance.

Camera lenses, certainly when targeting the low-budget range, induce a radial distortion that is best corrected. Our system relies on the use of the *Brown-Conrady* distortion model [3] to easily undistort the input images on the GPU.

Each input image I_i with $i \in \{1, \ldots, N\}$ is consequently segmented into a binary foreground silhouette S_i (see Figure 6(c)), to allow the consecutive view interpolation to adequately lever the speed and quality of the synthesis process. Two methods of segmentation are supported; Green screening according to Equation 1, where R_{I_i}, G_{I_i} and B_{I_i} are the red, green and blue components of I_i. For clarity the pixel location (x, y) has been omitted.

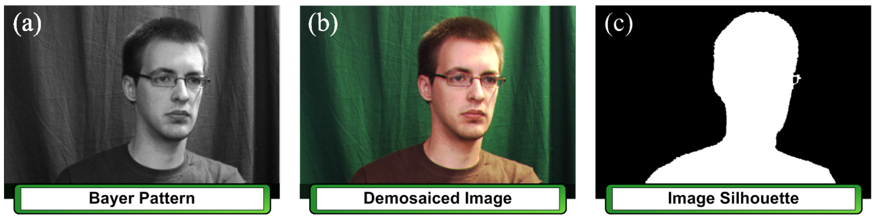


Fig. 6 The preprocessing module performs demosaicing, undistortion and segmentation of the input images.

$$S_i = \begin{cases} 1 : G_{I_i} > \tau_g \cdot (R_{I_i} + G_{I_i} + B_{I_i}) \\ 0 : G_{I_i} \leq \tau_g \cdot (R_{I_i} + G_{I_i} + B_{I_i}) \end{cases} \tag{1}$$

The second method is able to subtract a real-life background [18] according to Equation 2, where I_{B_i} is the static background picture and τ_g, τ_f, τ_b, τ_a are experimentally determined thresholds which are subjected to parameter fine tuning. For shadow removal, the cosine of the angle $\widehat{I_i I_{B_i}}$ between the color component vectors of the image pixel $I_i(x,y)$ and the static background pixel $I_{B_i}(x,y)$ is determined. As a final step, the silhouette is further enhanced by a single erosion and dilation [30].

$$S_i = \begin{cases} 1 : \|I_i - I_{B_i}\| > \tau_f \text{ or} \\ \quad\; \|I_i - I_{B_i}\| \geq \tau_b \text{ and } \cos(\widehat{I_i I_{B_i}}) \leq \tau_a \\ 0 : \|I_i - I_{B_i}\| < \tau_b \text{ or} \\ \quad\; \|I_i - I_{B_i}\| \leq \tau_f \text{ and } \cos(\widehat{I_i I_{B_i}}) > \tau_a \end{cases} \tag{2}$$

Both methods are evaluated on a pixel basis and require very little processing power, while still being robust against moderate illumination changes.

3.2 View Interpolation

To interpolate the desired viewpoint we adopt and slightly modify a plane sweeping approach based on the method of Yang et al. [32]. As depicted in Figure 7(a), the 3D space is discretized into M planes $\{D_1, \ldots, D_M\}$ parallel to the image plane of the virtual camera C_v. For each plane D_j, every pixel f_v of the virtual camera image I_v is back-projected on the plane D_j by Equation 3, and reprojected to the input images I_i according to Equation 4. Here $\mathbf{T}_j$ is a translation and scaling matrix that defines the depth and extent of the plane D_j in world space. The relationship between these coordinate spaces is represented in Figure 7(b).

$$f = \mathbf{V}_v^{-1} \times \mathbf{P}_v^{-1} \times \mathbf{T}_j \times f_v \tag{3}$$

$$f_i = \mathbf{P}_i \times \mathbf{V}_i \times f \tag{4}$$

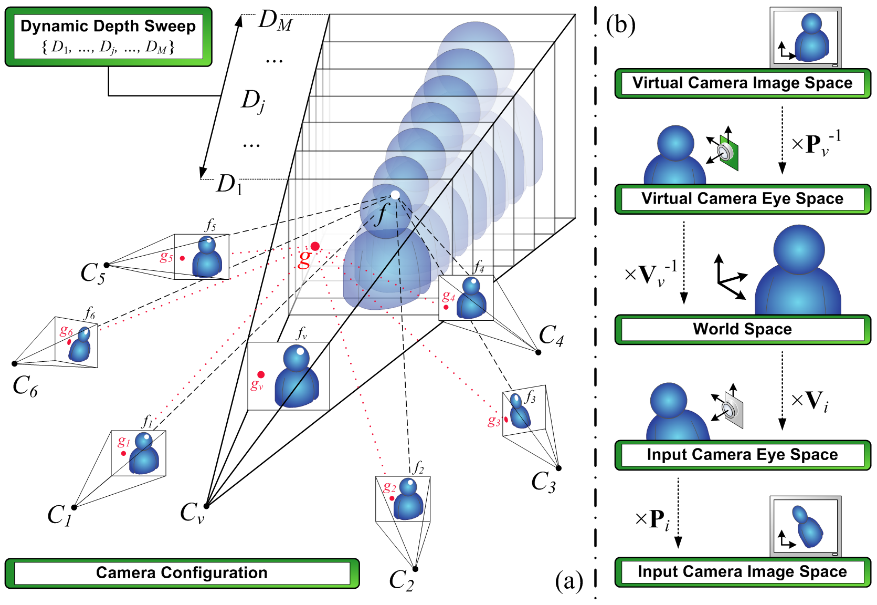


Fig. 7 Concept of the plane sweep algorithm.

Points on the plane D_j that project outside a foreground silhouette in at least one of the input images, are immediately rejected – e.g. point g in Figure 7(a) – and all further operations are automatically discarded by the GPU hardware. This provides a means to lever both speed and quality because noise in the segmentation masks will, with a high probability, not be available in all N cameras. Otherwise, the mean (i.e. interpolated) color ψ and a jointly defined custom matching cost κ are computed as in Equation 5.

$$\psi = \sum_{i=1}^{N} \frac{I_i}{N}, \quad \kappa = \sum_{i=1}^{N} \frac{\|\psi - I_i\|^2}{3N} \tag{5}$$

As opposed to Yang et al. [32], we propose the use of all input cameras to compute the matching cost. The plane is swept for the entire search range $\{D_1, \ldots, D_M\}$, and the minimum cost – together with the corresponding interpolated color – is per pixel selected on a winner-take-all basis, resulting in the virtual image I_v (see Figure 8(a)) and a joint depth map Z_v (see Figure 8(b)).

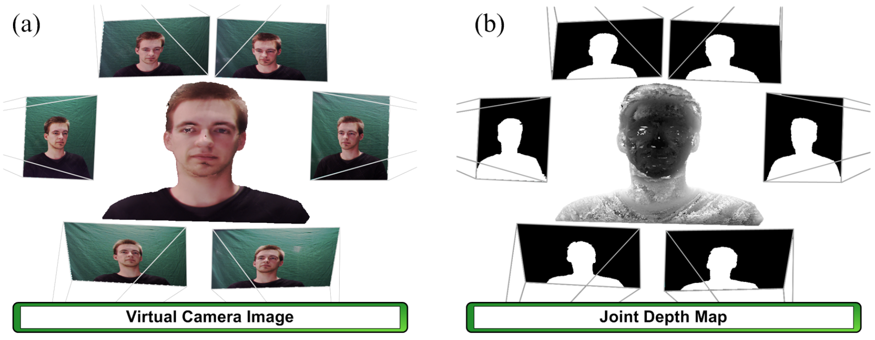


Fig. 8 The view interpolation module generates a virtual image and joint depth map.

3.3 Depth Refinement

The interpolated image calculated in the previous section still contains erroneous patches (see Figure 9(a)) and speckle noise due to illumination changes, partially occluded areas and natural homogeneous texturing of the face. These errors are even more apparent in the depth map Z_v and we therefore propose a photometric outlier detection algorithm that detects and restores the patches in Z_v.

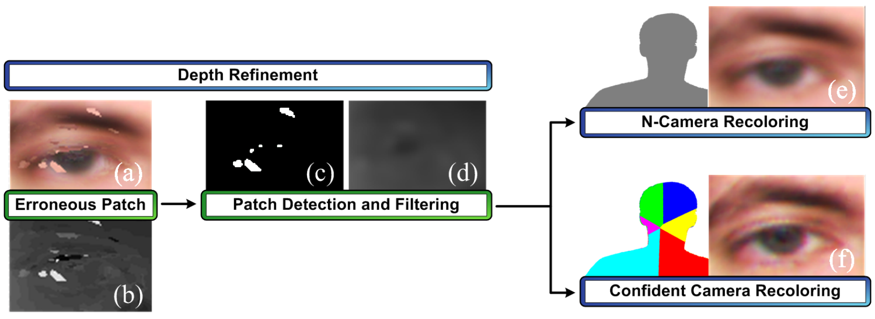


Fig. 9 Depth refinement and recoloring module concept.

To suppress the spatial high frequency speckle noise, we finally run a low-pass Gaussian filter over the depth map.

3.3.1 Erroneous Patch Filtering

To detect erroneous patches, we propose a spatial filter kernel λ, as depicted in Figure 10(a). For every pixel z_v of depth map Z_v, a two dimensional depth consistency check is performed according to Equation 6, where ε is a very small constant to represent the depth consistency. λ thereby defines the maximum size of patches that can be detected.

$$\begin{aligned} &\|Z_v(x-\lambda,y) - Z_v(x+\lambda,y)\| < \varepsilon \text{ or} \\ &\|Z_v(x,y-\lambda) - Z_v(x,y+\lambda)\| < \varepsilon \end{aligned} \tag{6}$$

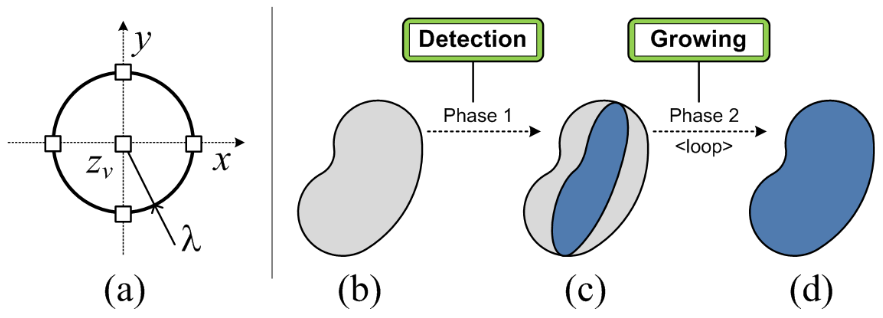


Fig. 10 (a) The proposed filter kernel, and (b–d) the outlier detection concept.

If the area passes the consistency check in one of the dimensions, the depth pixel z_v – and therefore the joint image pixel f_v – is flagged as an outlier if z_v does not exhibit the same consistency by exceeding a given threshold τ_o. Equation 7 shows the outlier test when a depth consistency is noticed in the X-dimension, an analogous test is used in case of consistency in the Y-dimension.

$$\left\| Z_v(x,y) - \frac{Z_v(x-\lambda,y) + Z_v(x+\lambda,y)}{2} \right\| > \tau_o \tag{7}$$

After performing the proposed filter kernel, the patch centers are detected, as conceptually represented in Figure 10(b) and Figure 10(c). Consistently, a standard morphological grow algorithm is executed, which causes the detected center to grow only if the neighbouring pixels exhibit the same depth consistency as the initial outliers. As depicted in Figure 9(c) and 10(d), the complete patch is thereby detected. As a final step for the patch filtering, the morphological grow is reversed and the detected patch is filled with reliable depth values from its neighbourhood. Since all of these operations are implemented on a pixel basis, they are inherently appropriate for implementation on a GPU, achieving a tremendous speedup compared to a generic CPU algorithm.

3.3.2 Speckle Noise Filtering

Due to the nature of the human face, a significant amount of large homogeneous texture regions are present. As indicated by Scharstein and Szeliski [24] these areas cause the depth map to contain spatial high frequency speckle noise. The noise is most effectively filtered by a low-pass filter, but this eliminates the geometrical correctness of the depth map. A standard 2D isotropic Gaussian filter is applied on the depth map and thanks to its separable convolution properties, it can even be highly optimized on graphics hardware [11].

3.4 Recoloring

All of the previous refinement steps involve changing the depth map Z_v, which is normally – due to the plane sweep – jointly linked to the image color in I_v. To restore this link, the refined depth map is used to recolor the interpolated image with the updated depth values. As opposed to other more geometrically correct approaches [17], we thereby significantly enhance the subjective visual quality. The system is currently able to recolor the image in two different approaches, each having their particular effect on the resulting quality.

3.4.1 N-camera Recoloring

The simplest and fastest recoloring solution is similar to the plane sweeping mechanism because it recomputes each pixel of the image I_v with an updated $\mathbf{T}_j$ matrix (Equation 3) according to the refined depth information. The interpolated pixel color is then again obtained by averaging all N cameras.

This approach generates very smooth transitions of the input images in the synthesized result, at the expense of loss of detail (see Figure 9(e)).

3.4.2 Confident Camera Recoloring

For each pixel f_v of the image I_v, the second recoloring solution determines which input camera C_i is closest in angle to the virtual camera C_v, and stores the camera index in a color map H_v according to Equation 8, where $\mathbf{h}_i = \overline{fC_i}$ is the vector from f to C_i, and $\mathbf{h}_v = \overline{fC_v}$.

$$H_v = \arg \max_{i \in \{1 \ldots N\}} \cos(\widehat{\mathbf{h}_v\, \mathbf{h}_i}) \tag{8}$$

We assume C_i to represent the optical image center of the camera, and f is the image point f_v back-projected to world space according to Equation 3, again with an updated $\mathbf{T}_j$ matrix. This recoloring scheme is illustrated in Figure 9(f), with depicted color map H_v.

Selecting the color from a single camera defined in H_v, ensures a sharply detailed synthesized image. However, the quality is sensitive to deviating colors between input cameras due to variations in illumination and color calibration (see Figure 9(f)).

3.5 Concurrent Eye Tracking

To restore the eye contact between the video chat participants, the camera C_v needs to be correctly positioned. Eye tracking can be performed robustly and more efficiently on CPU, and is therefore executed concurrently with the main processing of the system.

The 3D eye position is then mirrored toward the screen, resulting in the correct virtual viewpoint that is needed to restore the eye contact between the system users. The two screens are placed in a common coordinate space, as if they were pasted against each other. Hence, this creates the immersive effect of a virtual window into the world of the other participant.

3.6 Networking

Our prototype system sends the eye coordinates over the network, and therefore the requested image I_v can be computed locally at the peer that captures the relevant images. These cross computations bring the required network communication to a minimum, by avoiding the transfer of N input images. The total peer-to-peer communication thereby exists out of the synthesized images and the eye coordinates.

4 Complexity Control

The previously discussed plane sweeping method is an efficient method to create novel viewpoints. Nevertheless, we can increase performance even more by reducing the number of depth hypotheses. We propose 2 methods: an adaptive uniform plane distribution method where the nearest and farthest depth values are adapted to the scene, and an adaptive non-uniform plane distribution method, where the depth planes themselves are redistributed in space to move computational power to the places where there are actually objects. Both methods have other applications besides video conferencing and are discussed below.

4.1 Adaptive Uniform Plane Distribution

Most of the time, the head of a single person is visible in the camera views. To avoid heavy constraints on the participant's movement, a large depth range has to be scanned to keep the complete head in the virtual view. This actually infers a lot of redundant computations, since the head of the user only spans a small depth range. We therefore propose to dynamically limit the effective depth range to $\{D_{min}, \ldots, D_{max}\}$ (similar to Geys et al. [10], Rogmans et al. [23]) through a movement analysis on the normalized depth map histogram. This implicitly causes a quality increase of the plane sweep, as the probability of a mismatch due to homogeneous texture regions is significantly reduced. Moreover, all M depth planes can be focused as $\{D_1 = D_{min}, \ldots, D_M = D_{max}\}$, which leverages the dynamic range and thereby significantly increases the accuracy of the depth scan. Three separate cases can be distinguished, as the user moves in front of the screen:

- *Forward*: If the user moves forward, he will exit the active scanning range. Therefore, the histogram will indicate an exceptionally large number of detected depth pixels toward D_{min}.
- *Stable*: The histogram indicates a clear peak in the middle, this resolves to the fact that the user's head remains in the same depth range.
- *Backward*: Analog to forward movement of the user, the depth histogram will indicate a peak toward D_{max}.

As depicted in Figure 11(a) and Figure 11(b), we fit a Gaussian distribution function $G(\mu, \sigma)$ with center μ and standard deviation σ on the histogram. The effective depth range is updated according to Equation 9 and Equation 10, where b_1, b_2 are constant forward and backward bias factors that can be adopted to the inherent geometry of the scanned object. D'_{min} represents the previous minimal depth, and $\Delta = D'_{max} - D'_{min}$ for denormalization.

$$D_1 = D_{min} = D'_{min} + (\mu - b_1 \cdot \sigma)\Delta \tag{9}$$

$$D_M = D_{max} = D'_{min} + (\mu + b_2 \cdot \sigma)\Delta \tag{10}$$

As the user performs forward or backward movement, the center μ of the Gaussian fit changes and dynamically adapts the effective scan range of the system. The image will briefly distort in this unstable case, but will quickly recover as the depth scan is adapted for every image iteration. A real-time high frame rate therefore increases the responsiveness of the system, and is able to achieve fast restabilization. Normal moderate speed movement will thereby not be visually noticed by the participants.

4.2 Adaptive Non-Uniform Plane Distribution

This method of changing the nearest and farthest depth plane is very powerful for video conferencing with one person. There are, however, situations where this method will not work, for example, when multiple people are standing and walking around. Therefore, we present an optimization where the distribution of the planes is adapted to the actual scene content, instead of only the nearest and farthest depth.

A histogram is calculated of the resulting depth map. This histogram guides the plane distribution for the next temporal frame. This will redistribute computational

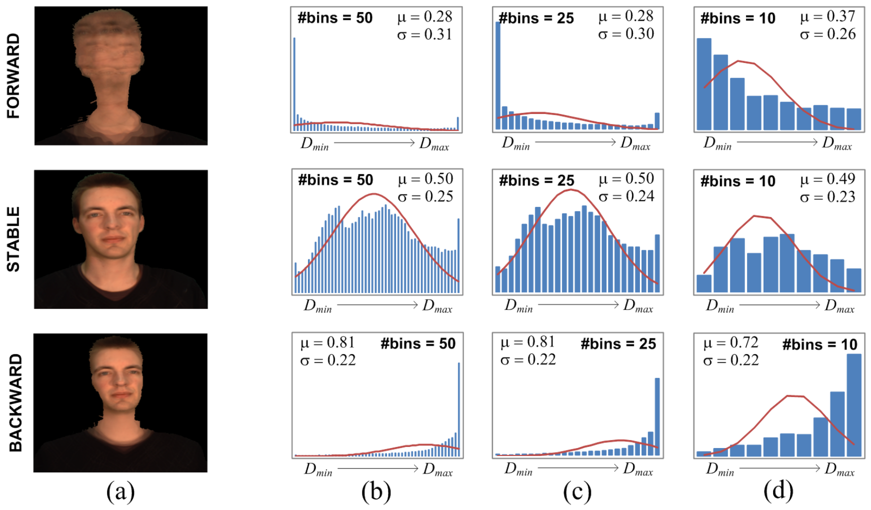


Fig. 11 The depth histogram-based movement analysis in normalized coordinates.

power to the more dense regions of the scene, and consequently increase the quality of the interpolation by reducing mismatches and noise.

When the scene consists of a limited range of depths between D_{min} and D_{max}, some processing resources are allocated to depth planes where no objects are present. This can be in between other objects. This is demonstrated in Figure 12(a). Here, a lot of planes are placed in the scene where no objects are positioned. This wastes resources and introduce more noise due to mismatches between the cameras. Therefore, we rearrange the distribution of the depth planes to provide less planes in depth ranges with less objects, and more, dense planes in scene regions with more objects. We determine the interest of a depth by analyzing the previous frame in a temporal sequence. The method works best when the movement of the scene is limited, such as moving people or scenes with many static objects.

After the interpolation step, we generate the histogram of the depth map using the well-known occlusion querying method [15] on GPU, allowing fast processing. The histogram can be seen in Figure 12(b). The occurrence of every depth value, as determined by the depth of the depth planes in the depth map, is counted. The histogram has discrete depth values between D_{min} and D_{max}, represented by the depth plane numbers. Scene depths of high interest will contain more depth values than depths of low interest. If there are depths in the scene where no objects are present, few of this depth values will be available in the depth map and this is reflected in the histogram. In the next frame, we want to provide more planes in depth ranges where a lot of depth values can be found, thus where there are large values in the depth histogram. The depth planes are not necessarily uniformly distributed, thus the histogram uses the depth plane number as the bin value, instead of the depth directly.

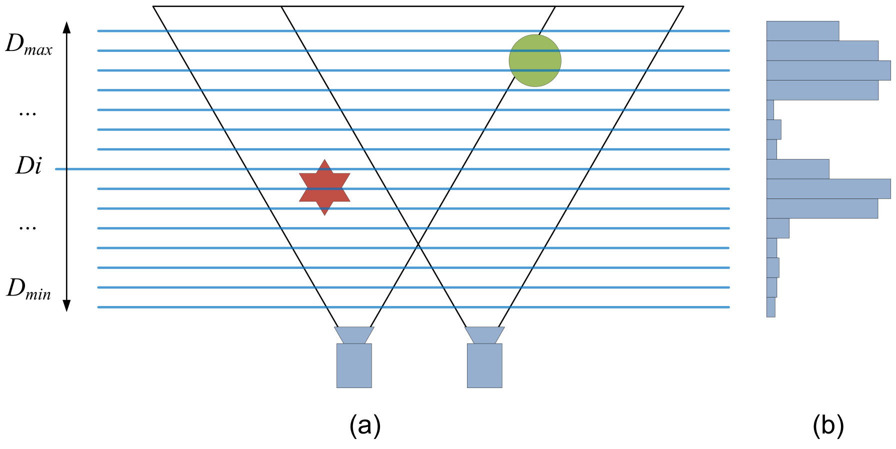


Fig. 12 (a) Uniform plane distribution (b) Histogram of the depth values.

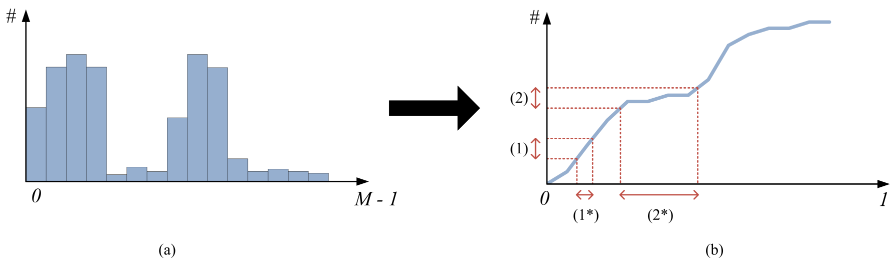


Fig. 13 (a) Resulting histogram (b) Corresponding cumulative histogram $H(x)$.

To use the depth distribution information, we convert the histogram to its cumulative version, as shown in Figure 13. Here, we do not count the number of occurrences per depth value, but we rather include the number of occurrences lower than this depth. Furthermore, we rescale the depth values from $[D_{min}, D_{max}]$, as represented by the depth plane numbers, to $[0,1]$. This transforms the non-uniform distribution of the depth planes to actual normalized depth values between 0 and 1. This transformation generates a monotonically increasing function $H(x) = y$, where $x \in [0,1]$ is a normalized depth value and y is the number of values in the rescaled depth map smaller or equal to x. For values of x where there are a lot of corresponding values in the depth map, $H(x)$ will be steep. For values of x with a low number of occurrences, $H(x)$ will be flat. Because of the non-uniform depth-plane distribution as input, $H(x)$ will be constant at some points where there were no depth planes for the corresponding normalized depth value.

We use the cumulative histogram to determine a mapping of a plane number m with $0 \leq m < M$ to a depth value D_m with $D_{min} \leq D_m \leq D_{max}$. For a uniform distribution, this would be:

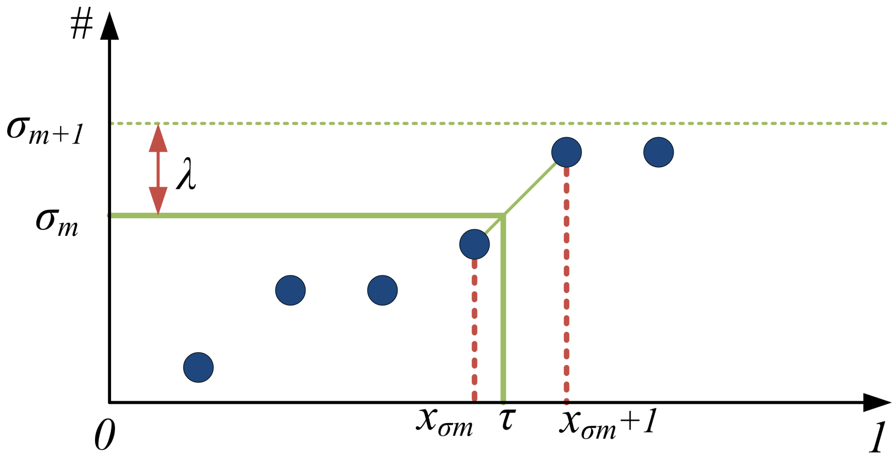


Fig. 14 Detail of the cumulative histogram with discrete values. τ is calculated by determining $x_{\sigma m}$ and $x_{\sigma m}+1$, such that $H(x_{\sigma m}) \leq \sigma_m$ and $H(x_{\sigma m}+1) > \sigma_m$, where σ_m represents a depth plane number.

$$D_m = D_{min} + \frac{m}{M}(D_{max} - D_{min}) \tag{11}$$

We adapt this uniform distribution method. When using the cumulative histogram to determine the distribution, we calculate a fraction $\tau_m \in [0,1]$ based on the plane number m, applied as follows:

$$D_m = D_{min} + \tau_m(D_{max} - D_{min}) \tag{12}$$

The fraction τ_m is determined by the cumulative histogram. The Y axis is divided in M cross sections, with a distance λ from each other, where $\lambda = max(H)/M$. Each cross section represents a depth plane m. The actual depth fraction τ_m for each cross section σ_m , i.e. a depth plane, is calculated by first determining the depth value $x_{\sigma m}$ where $H(x_{\sigma m}) \leq \sigma_m$ and $H(x_{\sigma m}+1) > \sigma_m$. This is demonstrated in Figure 14. Because the depth values x in the cumulative histogram are discrete, finding a value $x_{\sigma m}$ where $H(x_{\sigma m}) = \sigma_m$ is unlikely, and not desirable when generating planes that are dense, i.e. closer together, than the depth values provided in the cumulative histogram.

Once $x_{\sigma m}$ is determined, τ_m is calculated as follows:

$$\xi = \frac{m\lambda - H(x_{\sigma m})}{H(x_{\sigma m}+1) - H(x_{\sigma m})} \tag{13}$$

$$\tau_m = \xi(x_{\sigma m}+1) + (1-\xi)(x_{\sigma m}) \tag{14}$$

Figure 13(b) shows the transformation from a uniform depth-plane distribution to a non-uniform distribution based on the cumulative histogram. In region (1), where

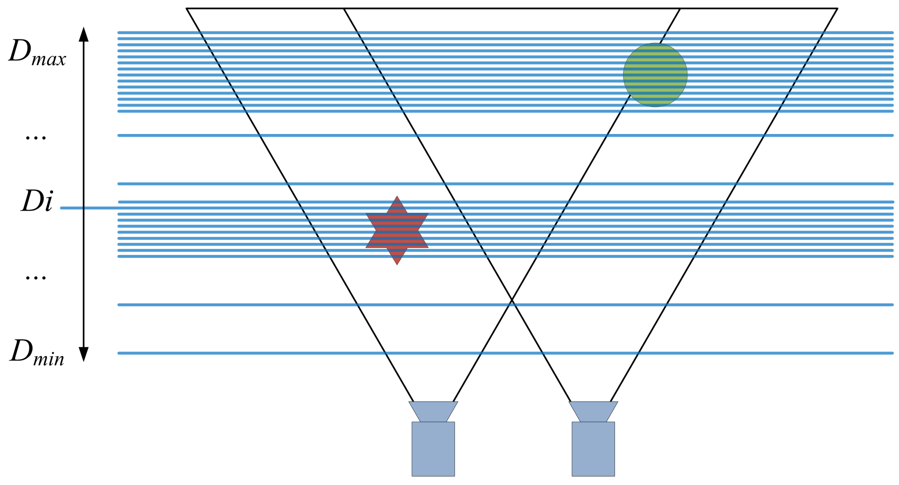


Fig. 15 Redistributed depth planes.

the cumulative histogram is steep, there is a dense plane distribution, as can be seen at (1*). When the cumulative histogram is flat, a sparse plane distribution is acquired, as can be seen at (2*).

Using τ_m, an actual depth for every plane m ($0 \leq m < M$) is determined and used in the plane sweeping step:

$$D_m = D_{min} + \tau_m(D_{max} - D_{min}) \tag{15}$$

This is depicted in Figure 15. Here, the planes are redistributed using the cumulative histogram of Figure 13(b). More planes are available for determining the depth of the objects, and less planes are available in empty space. It is desirable to include some planes in the empty spaces between objects to allow the appearance of objects in dynamic scenes. To allow this, all the values in the histogram are increased with a fixed number, based on the number of pixels. This way, the cumulative histogram is less flat in less interesting regions, allowing some planes here.

5 Implementation and Optimizations

The use of carefully selected and adapted algorithms allows us to exploit the GPU for general-purpose computations, a technique that is often referred to as general-purpose GPU computing. Our framework harnesses the powerful computational resources of the graphics hardware, and maximizes the arithmetic intensity of the algorithm to ensure real-time performance.

The algorithm execution is further accelerated by elevating the processing granularity from pixels to tiles, configured in a set of well-defined granularity parameters.

The processing is hereby only performed on the vertices (i.e. corner points) of the tiles, and therefore approximates – by inherent linear interpolation – the result of pixels inside the tile.

5.1 Improved Camera Data Transfer

Experimental profiling shows that downloading RGB-colored input images to the graphics card causes a three-fold increase in the data transfer time due to the memory bandwidth bottleneck of PCI-express, i.e. the bus connection between the motherboard northbridge controller and the GPU. This severely reduces the frame rate to two-third of its maximum capacity.

This bottleneck is effectively tackled by transferring Bayer-pattern images directly to the video memory. By inserting an additional demosaicing processing step, more computations are introduced, but only one third of input image data has to be sent, effectively increasing the performance over 30 percent. The reason is that graphics hardware benefit high arithmetic intensity kernels, as they process computations significantly faster than transferring data.

5.2 Acceleration by Elevated Granularity

By elevating the processing granularity from pixels to tiles, the algorithm execution speed can be drastically accelerated. In general, the computational complexity of the processing is inverse proportional to the granularity of the tessellation. If the tile size is chosen wisely, the speed can be significantly increased without noticeable visual quality impact. Our system uses three optimization schemes based on these speed-versus-quality trade-offs.

Tiled Undistortion Standard lens distortion is generally corrected on a pixel-basis level, but can be approximated by applying an equivalent geometrical undistortion to small image tiles using a resolution factor $0 < \rho_{tu} \leq 1$. Since a GPU pipeline exists out of a geometry and pixel processing stage, the lens correction can hence be ported from the pixel to the geometry stage. The pixel processing stage becomes clear to perform the consecutive segmentation processing in a single pipeline pass, which significantly leverages the GPU utilization.

Tiled Tallying in Reduced Bins For the movement analysis, the multi-resolution capabilities of the GPU are used to tally tiles instead of pixels in the histogram bins. This sampling resolution is expressed by a factor $0 < \rho_s \leq 1$, where ρ_s is proportional to the granularity of the tiles.
Evidently, it is of no use to have more histogram bins than the number of planes in the sweep. However, the essential part is deriving the parameters μ and σ to adjust the dynamic range of the depth scan. As depicted in Figure 11(c) and

Figure 11(d), we are able to approximate the histogram by reducing the number of bins, without a large impact on the Gaussian parameters. Therefore the number of bins are defined proportional to the number of planes M, with factor $0 < \rho_b \leq 1$. Heavily reducing the number of bins (see Figure 11(d)) causes the center μ to become less accurate, as it is shifted toward the center of the effective scan range. An optimal trade-off point can therefore be defined, since the accuracy loss will cause the responsiveness of the system to decrease.

Tiled Splatting Identical to the tiled undistortion, the depth map can be tessellated with a factor $0 < \rho_{ts} \leq 1$ to form a mesh for splatting tiles instead of pixels. This technique can significantly accelerate the confident camera recoloring, by interpolating angles between the tile corners.

6 Results

We demonstrate our previously discussed methods using a prototype setup. Our prototype setup is built with $N = 6$ auto-synchronized Point Grey Research Grasshopper cameras mounted on an aluminum frame, which closely surrounds the screen (see Figure 1, top right). The presented camera setup avoids large occlusions, and has the potential to generate high quality views since no image extrapolation is necessary. We have used the *Multi-Camera Self-Calibration* toolbox [27] to calibrate the camera setup offline, but a built-in camera setup into the screen would avoid this procedure due to fixed calibration parameters. Our software framework runs on an Intel Xeon 2.8GHz, equipped with 2GB system memory and an NVIDIA GeForce 8800GTX graphics card. Communication with the GPU is done through OpenGL, and it is programmed with the high level GPU language Cg.

6.1 Visual Quality

Final quality results using N-camera view recoloring are shown in Figure 16(a), and the results using confident camera recoloring are depicted in Figure 16(b). These results are generated under moderate variable illumination conditions, but with a fixed set of fine tuned practical system parameters summarized in Table 1. In Figure 16(a), some small artifacts along the ears and chin, together with minor ghosting around the neck, can still be noticed due to limitations of the depth refinement. The results generated with the confident camera recoloring are much more detailed and sharp, however some minor artifacts can be noticed due to abrupt camera transitions in the color map H_v. Nevertheless, the images maintain their integrity and are regarded as high subjective visual quality, while they convincingly seem to be making eye contact with the reader.

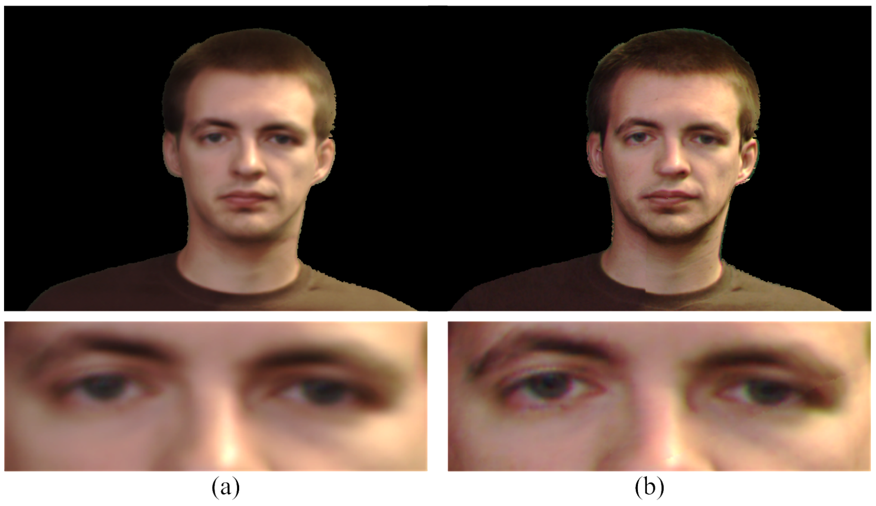


Fig. 16 Eye-gaze corrected images using (a) N-camera versus (b) confident camera recoloring.

Module	Parameter	Value
Preprocessing	τ_g	0.355
	τ_f	0.010
	τ_b	0.002
	τ_a	0.998
	ρ_{tu}	0.2
View Interpolation	N	6
	M	35
Depth Refinement	λ	20
	ε	0.2
	τ_o	0.3
Confident Camera Recoloring	ρ_{ts}	0.2
Movement Analysis	b_1	2.0
	b_2	2.0
	ρ_b	0.4
	ρ_s	0.5

Table 1 Set of optimized system parameters.

6.2 Performance

A detailed workload profiling for the main processing modules – using confident camera recoloring – can be seen in Figure 17, with input cameras and output resolutions of 800×600 pixels. By using on-line demosaicing, the arithmetic intensity can be kept relatively high, and even results in a higher execution speed than our previous implementation [7] using N-camera recoloring. The adaptive uniform plane distribution method was used to reduce computational complexity.

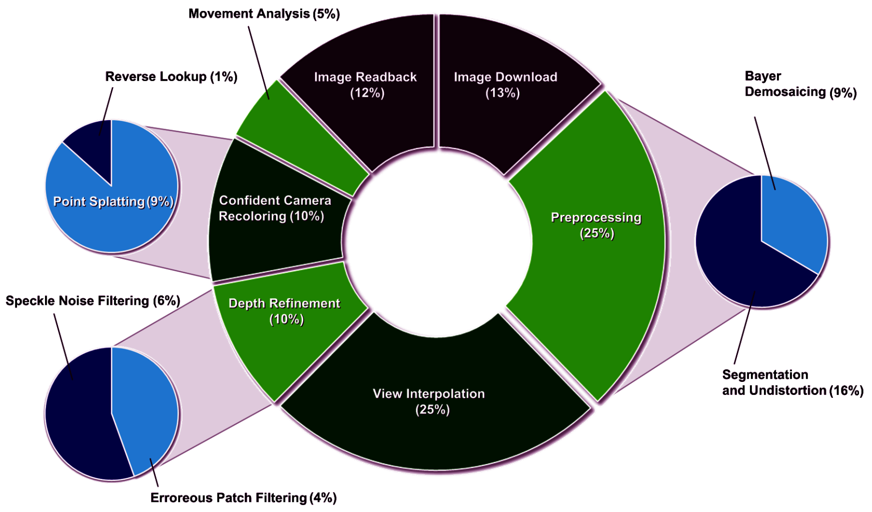


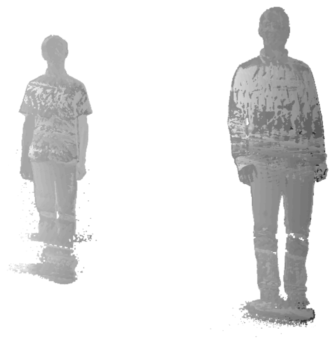
Fig. 17 Detailed workload profiling of the end-to-end optimized processing chain.

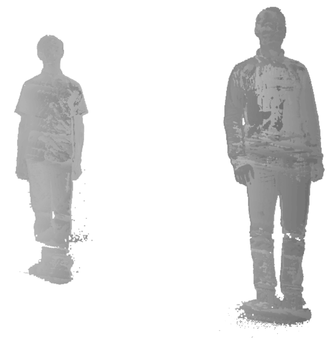
Summing up the different timings of the individual modules, we reach a confident speed of 27 fps for Full HD resolution, but our experimental setup is limited by 15 Hz support in the cameras and Firewire controller hardware. The current implementation speed allows for further quality optimization by advancing the algorithm and computational complexity.

6.3 Adaptive Non-uniform Plane Distribution

To demonstrate the validity of the adaptive non-uniform plane distribution system in multiple scenes, we created separate datasets with moving persons. We tested the method on different scenes and compared image quality and planes required.

The experiment shows that the quality is higher when a low number of planes is available, compared to the same number of planes using a uniform plane distribution. To increase the overall quality in both methods, we use foreground-background segmentation. Figure 18(a) shows the result for a uniform depth plane distribution. Artifacts caused by the sparse plane distribution can be clearly seen; the depth map shows clear outliers. The depth map when using a non-uniform plane distribution, based on the histogram of the first depth map, can be seen in Figure 18(b). Less noise and outliers in the depth values can be perceived. Furthermore, the silhouette is more distinct and the features of the persons are clearer. Using the non-uniform


(a)


(b)

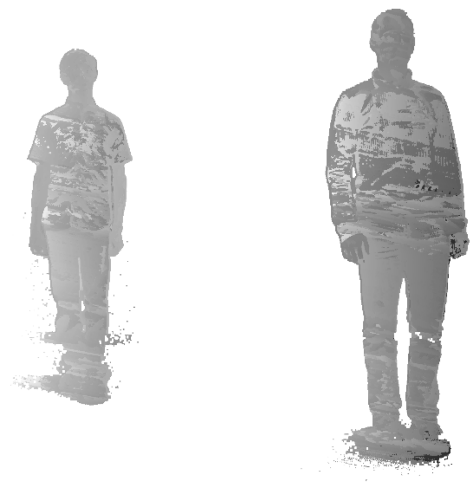

(c)

Fig. 18 (a) Depth map with a uniform depth plane distribution. A low number of planes (50) is used. (b) Depth map with a non-uniform depth plane distribution. A low number of planes (50) is used. (c) Depth map with a uniform depth plane distribution. A high number of planes (256) is used.

plane distribution increases the quality of the depth map using a low number of planes, therefore increasing overall performance.

Figure 18(c) shows the result for a high number of planes. Here, some noise and unclear edges can be perceived. These artifacts are effectively filtered out using the non-uniform plane distribution. The depth planes generating vague edges and noise are not used and cannot contribute to the depth map, and therefore to the noise and artifacts.

To demonstrate the effect of the cumulative histograms, Figure 19 shows an input image of a video sequence (a), the corresponding cumulative histogram of the depth map of the preceding frame (b) and the corresponding fraction τ from Equation 14 (c). When only one dominant depth can be perceived, such as in Figure 19(top), one steep section in the cumulative histogram is visible. This part is transformed to a flat value of τ, thus increasing the density of the planes in the corresponding region in the sweeping space. Flat sections of the cumulative histogram correspond to steep values in the graph of τ, resulting in a sparse plane distribution.

When multiple dominant depths are available in the scene, the cumulative histogram show multiple steep sections (see Figure 19, bottom). This results in multiple dense regions in the plane distribution, as reflected by the values of τ.

7 Conclusions

In this chapter, we presented a prototype for 3D video conferencing with eye gaze correction. A virtual camera is placed behind the display, and its image is synthesized with image-based rendering from multiple real cameras around the display, using a modified plane sweeping algorithm. A GPU implementation yields real-time performances at high visual quality.

Performance is even further increased thanks to algorithmic complexity reduction approaches. We presented a method to reduce the computational requirements by adapting the depth range and reducing the number of hypothesis depth planes in the search space, redistributing them to places with a high object density. This results in a substantial processing performance increase without impeding on the visual quality of the view synthesis.

References

[1] Shai Avidan and Amnon Shashua. Novel View Synthesis in Tensor Space. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1034–1040, Washington, DC, USA, 1997. IEEE.

[2] Harlyn Baker, Nina T. Bhatti, Donald Tanguay, Irwin Sobel, Dan Gelb, Michael E. Goss, W. Bruce Culbertson, and Thomas Malzbender. The Coliseum Immersive Teleconferencing System. In *Proceedings of the International Workshop on Immersive Telepresence*, Juan-les-Pins, France, 2002.

[3] Duane C. Brown. Decentering Distortion of Lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[4] Shenchang Eric Chen and Lance Williams. View Interpolation for Image Synthesis. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 279–288, Anaheim, CA, USA, 1993. ACM.

[5] Shao-Yi Chien, Shu-Han Yu, Li-Fu Ding, Yun-Nien Huang, and Liang-Gee Chen. Efficient Stereo Video Coding System for Immersive Teleconference

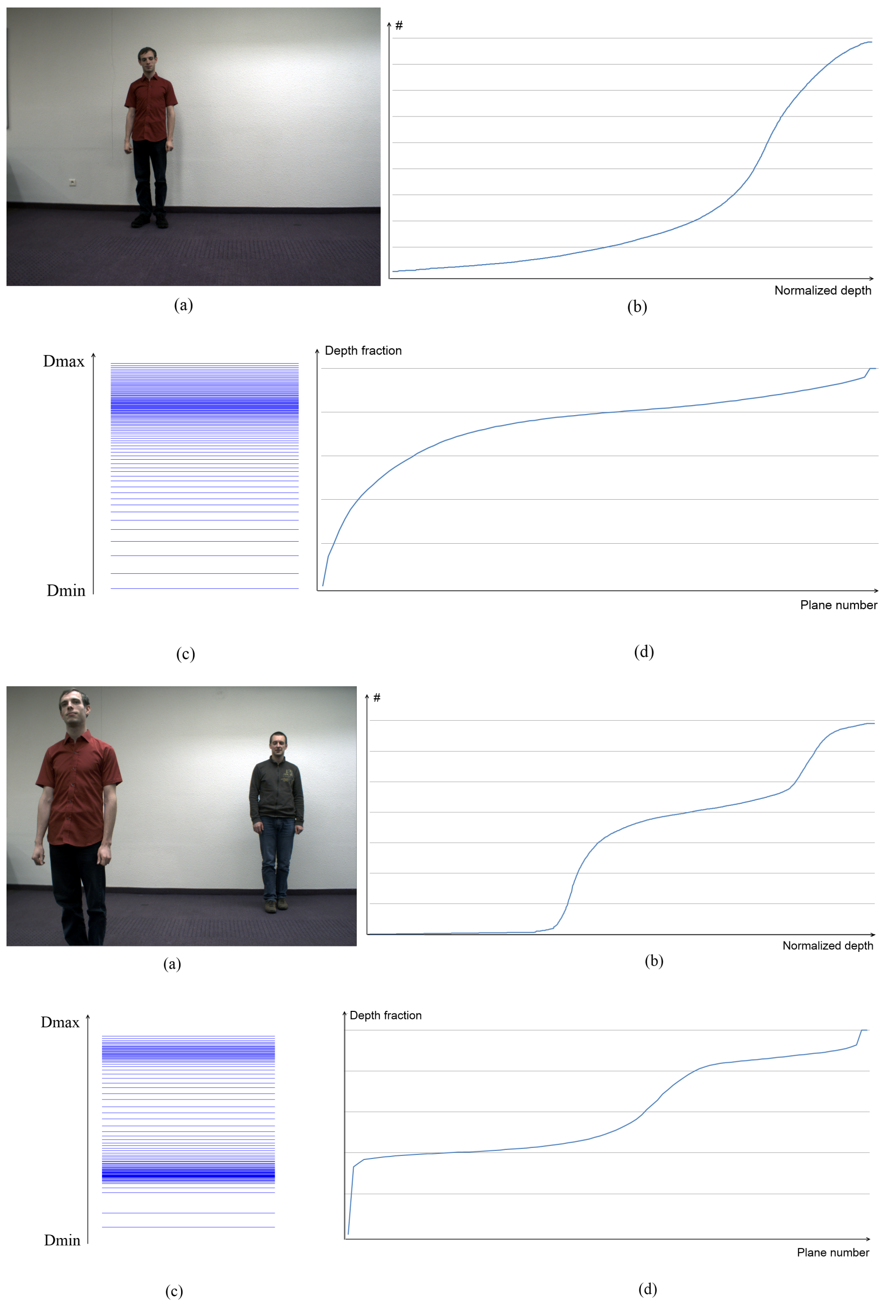


Fig. 19 Results for 1 person (top) and 2 persons (bottom). (a) Input image with one person. (b) Cumulative histogram of the depth map. (c) New depth plane distribution. (d) Corresponding fraction τ for a given plane number.

with Two-stage Hybrid Disparity Estimation Algorithm. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 749–752, 2003.

[6] Antonio Criminisi, Jamie Shotton, Andrew Blake, and Philip H.S. Torr. Gaze Manipulation for One-to-one Teleconferencing. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV)*, pages 191–198, Washington, DC, USA, 2003. IEEE.

[7] Maarten Dumont, Steven Maesen, Sammy Rogmans, and Philippe Bekaert. A Prototype for Practical Eye-Gaze Corrected Video Chat on Graphics Hardware. In *Proceedings of the International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, pages 236–243, Porto, Portugal, July 2008. INSTICC.

[8] Maarten Dumont, Sammy Rogmans, Steven Maesen, and Philippe Bekaert. Optimized Two-Party Video Chat with Restored Eye Contact using Graphics Hardware. *Springer Communications in Computer and Information Science*, 48(11):358–372, November 2009.

[9] Indra Geys and Luc Van Gool. Extended View Interpolation by Parallel use of the GPU and the CPU. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference: Videometrics VIII*, volume 5665, pages 96–107, 2004.

[10] Indra Geys, Thomas P. Koninckx, and Luc Van Gool. Fast Interpolated Cameras by Combining a GPU based Plane Sweep with a Max-Flow Regularisation Algorithm. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, pages 534–541, Washington, DC, USA, 2004. IEEE.

[11] Patrik Goorts, Sammy Rogmans, and Philippe Bekaert. Optimal Data Distribution for Versatile Finite Impulse Response Filtering on Next-Generation Graphics Hardware Using CUDA. In *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS)*, pages 300–307, Shenzhen, China, December 2009.

[12] Patrik Goorts, Sammy Rogmans, and Philippe Bekaert. Raw Camera Image Demosaicing using Finite Impulse Response Filtering on Commodity GPU Hardware using CUDA. In *Proceedings of the Tenth International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, Rome, Italy, 2012. INSTICC.

[13] Patrik Goorts, Cosmin Ancuti, Maarten Dumont, and Philippe Bekaert. Real-time Video-Based View Interpolation of Soccer Events using Depth-Selective Plane Sweeping. In *Proceedings of the Eight International Conference on Computer Vision Theory and Applications (VISAPP 2013)*, pages 131–137, Barcelona, Spain, 2013. INSTICC.

[14] Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. Optimization of Free Viewpoint Interpolation by Applying Adaptive Depth Plane Distributions in Plane Sweeping. In *Proceedings of the Tenth International Conference on Signal Processing and Multimedia Applications (SIGMAP 2013)*, Reykjavik, Iceland, 2013. INSTICC.

[15] Simon Green. Image Processing Tricks in OpenGL. Presentation at GDC, 2005.

[16] Xun Guo, Wen Gao, and Debin Zhao. Motion Vector Prediction in Multiview Video Coding. In *Proceedings of the 2005 International Conference on Image Processing (ICIP)*, pages 337–344, Genoa, Italy, 2005.

[17] B. J. Lei and E. A. Hendriks. Real-time Multi-step View Reconstruction for a Virtual Teleconference System. *EURASIP Journal on Applied Signal Processing*, 2002(1):1067–1087, 2002.

[18] Marcus Magnor, Marc Pollefeys, German Cheung, Wojciech Matusik, and Christian Theobalt. Video-based Rendering. In *Cources of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH cources)*, Los Angeles, California, 2005.

[19] Henrique S. Malvar, Li-wei He, and Ross Cutler. High-quality Linear Interpolation for Demosaicing of Bayer-patterned Color Images. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 485–488, Montreal, Canada, 2004. IEEE.

[20] Leonard McMillan. *An Image-based Approach to Three-dimensional Computer Graphics*. PhD thesis, University of North Carolina, 1997.

[21] Vincent Nozick, Sylvain Michelin, and Didier Arquès. Real-time Plane-Sweep with Local Strategy. *Journal of WSCG*, 14:121–128, jan 2006.

[22] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The Office of the Future: a Unified Approach to Image-based Modeling and Spatially Immersive Displays. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 179–188, New York, NY, USA, 1998. ACM.

[23] Sammy Rogmans, Maarten Dumont, Tom Cuypers, Gauthier Lafruit, and Philippe Bekaert. Complexity Reduction of Real-Time Depth Scanning on Graphics Hardware. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 547–550, Lisbon, Portugal, February 2009.

[24] Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[25] Oliver Schreer, Nicole Brandenburg, Serap Askar, , and M. Trucco. A Virtual 3D Video-Conference System Providing Semi-Immersive Telepresence: A Real-Time Solution in Hardware and Software. In *Proceedings of the eBusiness-eWork Conference*, pages 184–190, Venice, Italy, 2001.

[26] Steven M Seitz and Charles R Dyer. View Morphing: Uniquely Predicting Scene Appearance from Basis Images. In *Proceedings of the Image Understanding Workshop*, pages 881–887, New Orleans, LA, USA, 1997.

[27] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A Convenient Multicamera Self-calibration for Virtual Environments. *PRESENCE: teleoperators and virtual environments*, 14(4):407–422, 2005.

[28] Thomas Vetter. Synthesis of Novel Views from a Single Face Image. *International Journal on Computer Vision*, 28(2):103–116, 1998.

[29] Ruigang Yang and Marc Pollefeys. Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 211–220, Madison, WI, USA, june 2003. IEEE.

[30] Ruigang Yang and Greg Welch. Fast Image Segmentation and Smoothing using Commodity Graphics Hardware. *Journal of Graphics Tools*, 7(4):91–100, 2002.

[31] Ruigang Yang and Zhengyou Zhang. Eye Gaze Correction with Stereovision for Video-Teleconferencing. In *Proceedings of the 7th European Conference on Computer Vision Part II (ECCV)*, pages 479–494, London, UK, UK, 2002. Springer-Verlag.

[32] Ruigang Yang, Greg Welch, and Gary Bishop. Real-Time Consensus-Based Scene Reconstruction Using Commodity Graphics Hardware. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications (PG)*, pages 225–234, Washington, DC, USA, 2002. IEEE.

[33] Zhenyu Yang, Bin Yu, Wanmin Wu, Ross Diankov, and Ruzena Bajscy. Collaborative Dancing in Tele-immersive Environment. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 723–726. ACM, 2006.