A prototype for practical eye-gaze corrected video chat on graphics hardware

Peer-reviewed author version

A PROTOTYPE FOR PRACTICAL EYE-GAZE CORRECTED VIDEO CHAT ON GRAPHICS HARDWARE

Maarten Dumont[1], Steven Maesen[1], Sammy Rogmans[1,2], and Philippe Bekaert[1]
[1]*Hasselt University – tUL – IBBT, Expertise centre for Digital Media, Wetenschapspark 2, 3590 Diepenbeek, Belgium*
[2]*Multimedia Group, IMEC, Kapeldreef 75, 3001 Leuven, Belgium*
{*maarten.dumont, steven.maesen, sammy.rogmans, philippe.bekaert*}*@uhasselt.be*

Keywords: prototype, practical, video chat, eye-gaze correction, GPU, real-time

Abstract: We present a fully functional prototype to convincingly restore eye contact between two video chat participants, with a minimal amount of constraints. The proposed six-fold camera setup is easily integrated into the monitor frame, and is used to interpolate an image as if its virtual camera captured the image through a transparent screen. The peer user has a large freedom of movement, resulting in system specifications that enable genuine practical usage. Our software framework thereby harnesses the powerful computational resources inside graphics hardware, to achieve real-time performance up to 30 frames per second for 800×600 resolution images. Furthermore, an optimal set of finetuned parameters are presented, that optimizes the end-to-end performance of the application, and therefore is still able to achieve high subjective visual quality.

1 INTRODUCTION

Peer-to-peer interactive video chat is becoming increasingly popular, but still has some major drawbacks which prevent a definitive breakthrough in the common public. One of the most important reasons is that the participant is not able to simultaneously look at the screen and the camera, leading to the loss of eye contact (**?**). We therefore present a fully functional prototype (see Fig. 1) that corrects the eye gaze of the peers by using multiple cameras, solving typical problems for genuine practical usage.

Previous solutions such as (**?**) either implement their framework on commodity CPUs, resulting in a very low framerate when sufficient visual quality is required. On the opposite side, solutions such as (**?**; **?**) involve the use of expensive dedicated hardware, or have an unpractical camera setup. Others optimize parts of the application, such as multi-camera video coding (**?**; **?**) for efficient data communication and real-time view synthesis (**?**; **?**; **?**) on graphics hardware, but neither of them integrate and optimize the end-to-end performance for eye gaze-corrected video chat. The end-to-end performance for camera interpolation is optimized in (**?**), but they assume a rectified two-fold camera setup and no illumination changes due to the use of the Middlebury dataset (**?**).

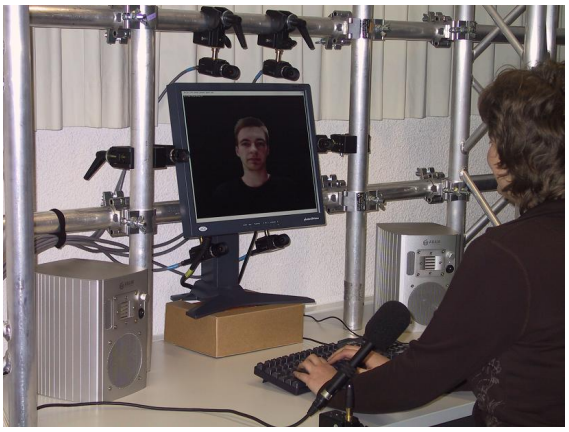

Figure 1: Peer setup of our prototype.

Our prototype uses a practical camera setup that can be easily integrated in the monitor frame, and its framework harnesses the powerful computational resources inside the Graphics Processing Unit (GPU) to achieve real-time performance on commodity PCs. Section 2 of the paper describes the system architecture in detail, which proposes the use of several carefully selected and slightly adapted algorithms that are appropriate for implementation on graphics hard-

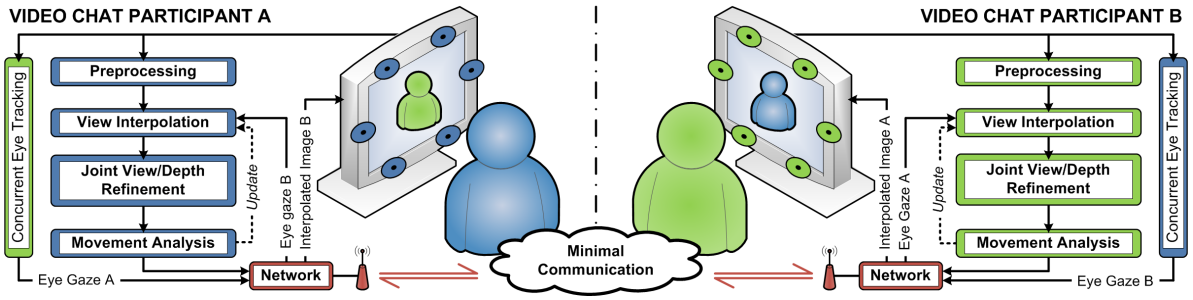

Figure 2: Data flow and overview of our system architecture.

ware. The end-to-end system thereby achieves both high speed and quality with only few constraints. We also provide a number of GPU specific optimizations in Section 3 to ensure real-time application performance. Section 4 discusses the results of the prototype, Section 5 ultimately concludes the paper, and Section 6 deals with future work.

2 SYSTEM ARCHITECTURE

As depicted in Fig. 2, the main functionality of our system consists of four consecutive processing modules that are completely running on a GPU. In an initial step, images $I_1,\ldots,I_N$ are fetched from N cameras $C_1,\ldots,C_N$ that are closely aligned along the screen. The first module performs lens correction and image segmentation, as a form of preprocessing, to enhance both the quality and speed of the consecutive view interpolation.

This second module interpolates an image I_v, as seen with a virtual camera C_v that is positioned behind the screen and produces a consistent depth map Z_v. The image I_v is computed as if camera C_v captured the image through a completely transparent screen.

However, the synthesized image still has a number of noticeable artifacts in the form of erroneous patches and speckle noise. The third refinement module is therefore specifically designed to tackle these problems by detecting photometric outliers based on the accompanying depth map.

In a final step, the depth map Z_v is also analyzed to dynamically adjust the system and thereby avoids heavy constraints on the user's movements.

Next to the main processing on graphics hardware that synthesizes I_v, the camera C_v needs to be correctly positioned to restore eye contact between the participants. An eye tracking module 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 $I_1,\ldots,I_N$ do not have to be sent over the network, 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.

2.1 Preprocessing

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 (**?**) to easily undistort the input images pixel-based on the GPU.

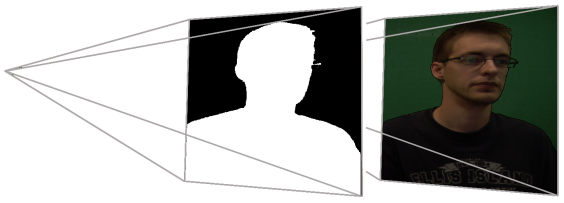

Figure 3: The preprocessing module segments the input camera image.

Each input image I_i with $i \in \{1,\ldots,N\}$ is consequently segmented into a binary foreground silhouette S_i (see Fig. 3), to allow the consecutive view interpolation to adequately lever the speed and quality of the synthesis process. Two methods of segmentation are supported; Greenscreening according to (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.

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

The second method is able to subtract a real-life background (**?**) according to (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 finetuning. 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

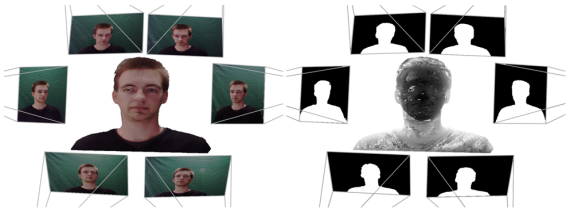

Figure 4: The view interpolation module generates a virtual image and joint depth map.

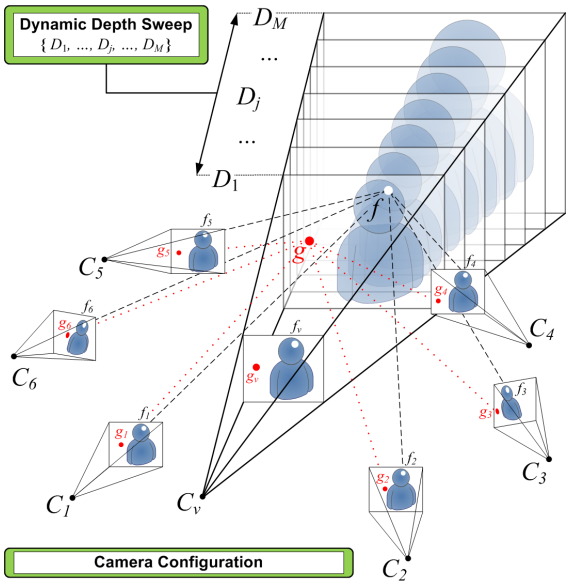

Figure 5: Concept of the plane sweep algorithm.

is further enhanced by a single erosion and dilation (**?**).

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

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

2.2 View Interpolation

To interpolate the desired viewpoint (see Fig. 4) we adopt and slightly modify a plane sweeping approach based on (**?**). As depicted in Fig. 5, 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 (3), and reprojected to the input images I_i according to (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

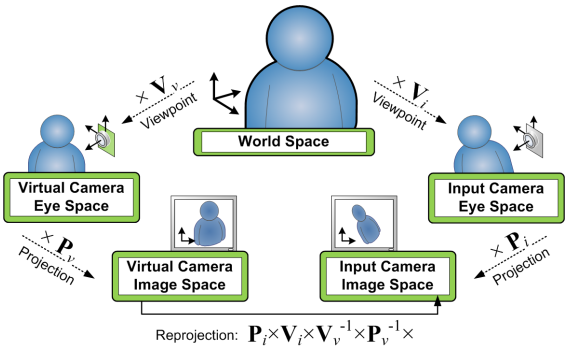

Figure 6: Reprojection from virtual to input images.

relationship between these coordinate spaces is represented in Fig. 6.

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

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

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 Fig. 5 – and all further operations are automatically discarded by the GPU hardware. This provides a means to lever both speed and quality because segmentation noise 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 (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 (**?**), 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-Takes-All* basis, resulting in the virtual image I_v and a joint depth map Z_v (see Fig. 4).

2.3 Joint View/Depth Refinement

The interpolated image calculated in the previous section still contains erroneous patches (see Fig. 4, magnified in Fig. 7) and speckle noise due to illumination changes, partially occluded areas and natural homogeneous texturing of the human 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.

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

In a final step, the refined depth map is used to recolor the interpolated image with the updated depth

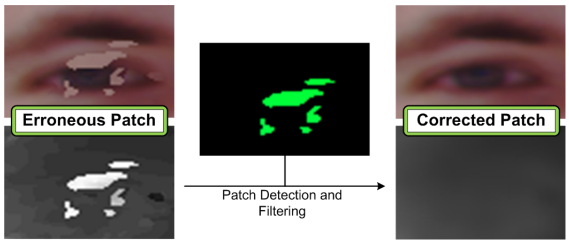

Figure 7: Joint view/depth refinement module concept.

values. As opposed to other geometrically correct approaches (**?**), we thereby significantly enhance the subjective visual quality.

2.3.1 Erroneous Patch Filtering

To detect erroneous patches, we propose a filter kernel as depicted in Fig. 8a. For every pixel z_v of depth map Z_v, a two dimensional depth consistency check is performed for its neighbourhood λ according to (6), where ε is a very small constant to represent the depth consistency. λ thereby defines the radius of the filter kernel, and the maximum size of patches that can be detected.

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

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. Eq. (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.

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

After performing the proposed filter kernel, the center of patches – as conceptually represented in Fig. 8b and Fig. 8c – are detected. Consistently, a standard morphological grow algorithm is executed in a loop, which causes the detected center to grow only if the neighbouring pixels exhibit the same depth consistency as the initial outliers. As depicted in Fig. 8d, 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.

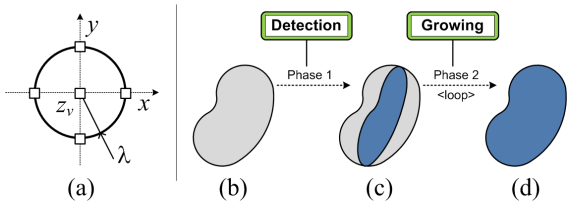

Figure 8: (a) The proposed filter kernel, and (b–d) the outlier detection concept.

2.3.2 Speckle Noise Filtering

Due to the nature of the human face, a significant amount of large homogeneous texture regions are present. As indicated by (**?**) 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 eliminates the geometrical correctness of the depth map. Therefore as opposed to methods such as (**?**), we rather enhance the subjective visual quality instead of geometrical accuracy.

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.

2.3.3 Recoloring

All of the previous 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 image I_v is recomputed with an updated $\mathbf{T}_j$ matrix, according to the filtered depth information. Since erroneous patches and speckle noise are now filled or leveled with consistent – rather than geometrically correct – depth values, the recolored image is interpreted as plausible and thereby subjectively regarded as higher quality.

2.4 Movement Analysis

To avoid heavy constraints on the participant's movement, a large depth range has to be scanned. This actually infers a lot of redundant computations, since the head of the user only spans a small range. We therefore propose to dynamically limit the effective depth range to $\{D_{min},\ldots,D_{max}\}$ similar to (**?**), 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 towards 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 towards D_{max}.

As depicted in Fig. 9a to f, 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 (8) and (9), 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.

$$D_1 = D_{min} = D'_{min} + (\mu - b_1 \cdot \sigma)\Delta \quad (8)$$

$$D_M = D_{max} = D'_{min} + (\mu + b_2 \cdot \sigma)\Delta \quad (9)$$

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 (see extreme cases in Fig. 9a and c), but will quickly recover as the depth scan is adapted for every image iteration. A real-time high framerate 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.

2.5 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 more robust and efficiently on CPU, and is therefore executed concurrently with the main processing of the system. Afterall, the N input images are implicitly available on system memory before they are sent to the video memory of the GPU.

Face and eye candidates are detected in every input image as described in (**?**), but we optimized the detection process by predicting the eye positions based on previous frames and using a hierarchical approach to search for eye pixels. An epipolar cross check is consequently used to estimate the most reliable eye candidates in all images. The two best eye candidates are ultimately used to triangulate their 3D positions.

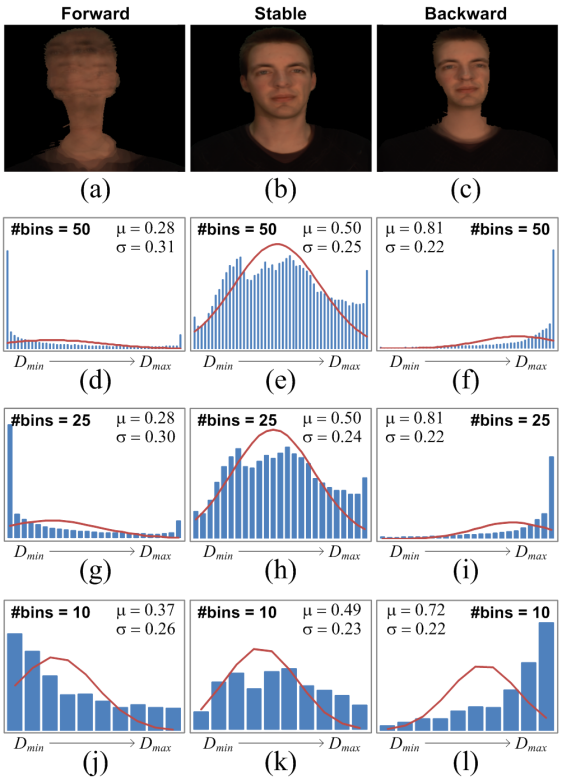

Figure 9: Overview of the depth histogram-based movement analysis in normalized coordinates.

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

2.6 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.

Real-time speed can therefore easily be achieved over various types of networks without the need of any compression. Due to the segmentation result S_v, the system can be efficiently equipped with *Region-of-Interest* coding, to enable the use over low-data rate

networks as well.

3 OPTIMIZATIONS

Our framework harnesses the powerful computational resources of the graphics hardware to ensure real-time performance. 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 (GPGPU).

GPGPU is traditionally only capable of utilizing a part of the GPU resources due to its computer graphics nature (**?**). We optimized the preprocessing to bypass this constraint, which leverages the GPU utilization and execution speed without any noticeable loss of quality.

Being an additional advantage, the histogram for movement analysis is approximated by reducing its number of bins, since only the correct Gaussian distribution fit is required.

As this approximation infers dynamic looping inside the GPU, we have built-in support for dynamic GPU programming to be able to optimize and unroll loops at run-time.

3.1 Improved Utilization

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. 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. Initially, the computational complexity of the undistortion is inverse proportional to the granularity of the tessellation, resulting in a speedup without any visual quality loss if the tile size is chosen correctly. Next to this fact, the pixel processing stage is clear to perform the consecutive segmentation processing in a single pipeline pass, which significantly leverages the utilization of the GPU.

3.2 Accelerated Histogramming

For the movement analysis, the essential part is deriving the parameters μ and σ to adjust the dynamic range of the depth scan. As depicted in Fig. 9d to l, we are able to approximate the histogram by reducing the number of bins, without a large impact on the Gaussian parameters. Heavily reducing the number of bins (see Fig. 9j to l) causes the center μ to become less accurate, as it is shifted towards the center of the effective scan range. An optimal trade-off point

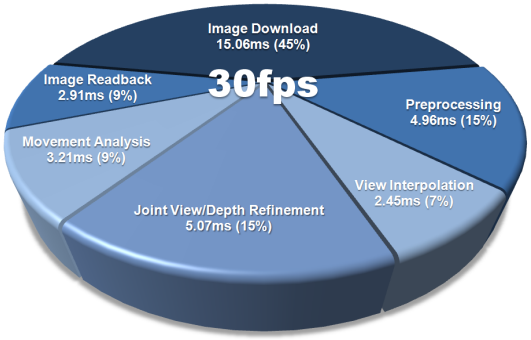

Figure 11: Workload profiling.

can therefore be defined, since the accuracy loss will cause the responsiveness of the system to decrease.

3.3 Dynamic GPU Programming

Similar to the dynamic looping needed to variate the number of bins in the histogramming, looping inside the GPU is very often required. To enhance the execution speed, our framework has built-in support to generate and compile the GPU programming code at run-time. Hereby, the loops can always be unrolled and internally fully optimized.

4 RESULTS

Our optimal prototype setup is built with $N = 6$ auto-synchronized Point Grey Research Grasshopper cameras mounted on an aluminum frame, so they can be closely aligned to the screen (See Fig. 1). 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 (**?**) 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 language Cg.

Final quality results are shown in Fig. 10, under moderate variable illumination conditions, but with a fixed set of finetuned practical system parameters grouped in Table 1. Some small artifacts along the ears and chin, together with minor ghosting around the neck, can still be noticed due to limitations of the joint view and depth refinement. Nonetheless, the images maintain their integrity and are regarded

Figure 10: Eye-gaze corrected images with variable illumination and fixed optimal parameters.

as high subjective visual quality, while they convincingly seem to be making eye contact with the reader.

Module	Parameter	Value
Preprocessing	τ_g	0.355
	τ_f	0.010
	τ_b	0.002
	τ_a	0.998
View Interpolation	N	6
	M	35
Joint View/Depth	λ	20
Refinement	ε	0.2
	τ_o	0.3
Movement Analysis	b_1	2.0
	b_2	2.0
	$\#bins$	15

Table 1: Set of optimized system parameters.

A detailed workload profiling for the main processing modules can be seen in Fig. 11, with image and camera resolutions of 800×600 pixels. A large amount of time is needed to perform the six input image download and synthesized image readback to and from the GPU. Hence, the typical data locality importance is illustrated, and the core reason to implement all main processing steps on graphics hardware is motivated. The rather large weight for the preprocessing is mainly due to the $N = 6$ times execution on all input images. Furthermore, a significant amount of processing is concentrated in the refinement and movement analysis, as it levers the quality independent of the amount of input images.

Summing up the different timings of the individual modules, we can reach a confident speed of 30fps, but in our experimental setup is limited by 15Hz support in the cameras and Firewire controller hardware. We foresee these specifications as for genuine practical usage, since the end-to-end system is optimized for minimum practical constraints.

5 CONCLUSION

We have presented a system prototype for practical eye-gaze correction between two video chat participants, with a minimal amount of constraints. A six-fold camera setup that is closely aligned along the screen, is proposed for possible integration into the monitor frame. Besides the convenient camera placement, the presented setup also avoids large occlusions, and has the potential to generate higher quality views compared to camera setups that require constant image extrapolation.

Our software framework harnesses the computational resources of the graphics hardware through GPGPU, and is able to achieve real-time performance. A framerate of 30fps can be achieved for 800×600 image resolutions, but our experimental setup is limited by 15 Hz cameras. Thanks to a movement analysis, the system provides the user with a large freedom of movement.

We have improved the end-to-end performance of the system, by introducing optimizations that have no noticeable loss of visual quality. A fixed set of fine-tuned parameters is able to generate interpolated images with high subjective visual quality – rather than being geometrically correct – under variable conditions. Moreover, the system specifications thereby enable genuine practical usage of convincing eye-gaze corrected video chat.

6 FUTURE WORK

Future work will focus on improving the movement analysis. Furthermore, multiple objects can be intelligently distinguished by combing silhouette information, enabling eye-gaze corrected multi-party video conferencing.

In the largest extent, background objects can be recognized and interpolated with correct motion parallax, providing a true immersive experience for the participants.

ACKNOWLEDGEMENTS

We would like to explicitly thank Tom Haber for the development of the underlying software framework, his contributions and suggestions.

Part of the research at EDM is funded by the European Regional Development Fund (ERDF) and the Flemish government.

We would also like to acknowledge the Interdisciplinary Institute for Broadband Technology (IBBT) for funding this research in the scope of the ISBO Virtual Individual Networks (VIN) project.

Finally, co-author Sammy Rogmans is funded outside this project, by a specialization bursary from the IWT, under grant number SB071150.