Fast and flexible GPU implementation of the view-dependent error diffusion algorithm

Antoine Lagrange, Antonin Gilles, Kevin Heggarty, Bruno Fracasso

Fast and flexible GPU implementation of the view-dependent error diffusion algorithm

Antoine Lagrange[1,3,*] Antonin Gilles[1] Kevin Heggarty[1,2] Bruno Fracasso[1,3]

[1] IRT b<>com
Cesson-Sévigné
France

[2] IMT Atlantique
Brest
France

Abstract

During the quantization of the hologram values part of the information is lost, which introduces quantization noise in the reconstruction. To enhance the hologram quality, we developed in a previous work a view-dependent error diffusion algorithm which rejects the quantization noise in specific views. However, this algorithm is slow which prevents its use in real-time applications. In this paper, we present a GPU implementation of this technique which uses the inherent parallelism of the error diffusion algorithm. Moreover, we introduce a technique to select diffusion weights, which allows a trade-off between computation time and reconstruction quality. Experimental results demonstrate the ability of our approach to quickly generate high-quality holograms.

1 Introduction

Immersive technologies are increasingly being studied as the acquisition and the visualization of 3D information are becoming easier [1]. Stereoscopy is the most commonly used technique and consists in displaying two different views of the scene to each of the observer's eye [2]. However, this method suffers from the vergence-accommodation conflict because stereoscopy does not provide all the depth cues of the Human Visual System (HVS), which causes eyestrain and headaches [3]. At the opposite, holography provides all the depth information required by the HVS which makes this immersive technology one of the most promising [4]. Holographic approaches rely on the diffraction of an incoming light beam to reproduce the waves scattering from a 3D scene, which allows a viewer to see objects as if they were physically there.

However, some limitations still remain such as the quantization of the hologram values that are encoded in floating point format with conventional generation methods. Indeed, in order to be visualized on current holographic displays, Computer-Generated Holograms (CGHs) must be converted to pure amplitude or phase signals with quantized values [5]. During the conversion process, part of the signal information is lost and quantization noise appears in the reconstructed scene. To enhance the visualization quality, one can use the well-known error diffusion algorithm, which rejects the conversion noise outside the reconstruction area by propagating the quantization error of each pixel to its neighbours according to a set of fixed diffusion weights [6, 7]. In a previous work, we proposed a new view-dependent error diffusion (VDED) algorithm [8] which is based on the signal window error diffusion (SWED) technique [9]. Our approach rejects the quantization noise in specific views by applying the SWED technique on hologram blocks and relies on a large set of diffusion weights. However, the Central Processing Unit (CPU) implementation of the VDED algorithm is slow, which prevents its use in real-time applications.

Fast Graphics Processing Unit (GPU) implementations of the conventional error diffusion method have been proposed in the literature. In [10, 11] the authors proposed the pinwheel error diffusion technique which can be easily parallelized but creates periodical artefacts. At the opposite, the method proposed in [12] exploits the parallelism inherent to the Floyd-Steinberg approach [13]. Even if the parallelism of this method is limited by the recursive nature of the error diffusion algorithm, it does not suffer from specific artefacts since it provides the same result as a sequential approach. Thus, to further reduce the execution time and keep a high image quality, several optimizations based on the Compute Unified Device Architecture (CUDA) were presented in [14, 15]. However, these approaches rely on four fixed diffusion weights and are therefore incompatible with the larger set of varying weights of the VDED algorithm.

*This work has been achieved within the Research and Technology Institute b<>com, dedicated to digital technologies. It has been funded by the French government through the National Research Agency (ANR) Investment referenced ANR-A0-AIRT-07. Corresponding author: antoine.lagrange@b-com.com.

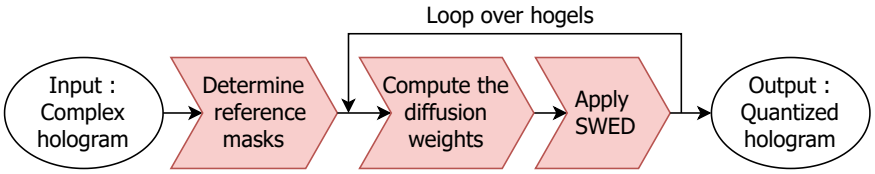


Figure 1: Block diagram of the VDED algorithm.

In this paper, we propose a GPU implementation of the VDED algorithm in order to reduce its computation time. This approach relies on the decomposition of the hologram blocks in parallelograms as in [14], the use of the Single Kernel Soft Synchronisation (SKSS) technique [15] which allows fast inter-block communication on GPU [16] and the optimisation of the memory accesses. Although these techniques speed up the process by several orders of magnitude, the spatial distribution of the diffusion weights limits the parallelisation capacity of the VDED approach. Thus, to further decrease the execution time, we propose a new weight selection technique which allows more parallelism and a better use of the GPU resources in exchange for a quality loss. Experiments demonstrate that the GPU implementation is up to 117 times faster than the CPU implementation. Moreover, the quality of the reconstructions are evaluated subjectively with numerical simulations.

The following of this paper is organized as follows: Section 2 is a reminder of the view-dependent error diffusion algorithm and Section 3 presents the techniques which allow a fast and flexible GPU implementation of our method. Finally, experimental results are analyzed in Section 4.

2 Viex-dependent error diffusion

In a previous work [8], we identified two limitations that restrict the numerical generation of holograms: the huge computational load at high resolution and the loss of quality during the quantization process. To overcome these issues, we propose a novel view-specific layered-based stereogram approach including a view-dependent error diffusion algorithm. This method selects the light waves that are scattered to a specific visualisation window and uses the VDED technique to reject the quantization noise outside the viewing area. In this way, the computation time was reduced up to 94% and quality improvements were observed especially on the contrast of the scene objects. Moreover, our generation method has also two additional quality enhancements: the imperceptibility of the conjugate order and the increased brightness of the reconstructed scene.

The VDED algorithm is based on the SWED technique which puts the quantization noise in specific regions of the reconstruction plane by means of quantization noise masks. The main difference between these two approaches is the hologram partitioning in the VDED algorithm. Indeed, in this method the hologram is first divided in several blocks and for each of them a set of diffusion weights is computed. On the other hand, during the quantization process, the SWED uses a single set of coefficients for the entire hologram. Hence, in the VDED algorithm the quantization errors are diffused differently according to their position in the hologram which allows to reject the quantization noise in specific views. In other words, the quantization noise is only visible for specific positions of the viewer. Conversely, in the SWED, the quantization noise is always located in the same area regardless of the observer viewpoint.

The input of the VDED technique is a complex hologram decomposed in several blocks called hogels. The first step is to determine two binary reference masks, $m_{\mathrm{s}}(\xi,\eta)$ and $m_{\mathrm{hf}}(\xi,\eta)$. The aim of $m_{\mathrm{hf}}(\xi,\eta)$ is to limit the high frequencies in the hologram. Therefore, this mask corresponds to the area on the edges of the reconstruction plane. The mask $m_{\mathrm{s}}(\xi,\eta)$ is associated to the region seen by an observer through a hogel and its symmetric area in the reconstruction plane. Indeed, the quantization noise should be rejected from the area observes by a viewer. Moreover, the conjugate order scattering from the symmetric area reaches the visualisation window explaining that it should also stay free of any quantization noise. These regions are specific to a hogel and are determined by translating $m_{\mathrm{s}}(\xi,\eta)$ by $(\xi_{\mathrm{tr}},\eta_{\mathrm{tr}})$ and $(-\xi_{\mathrm{tr}},-\eta_{\mathrm{tr}})$, respectively. Here $m_{\mathrm{s}}(\xi,\eta)$ is considered to be centered on the reconstruction plane. The VDED technique iterates over the hogels and computes each set of diffusion weights by applying a Fourier transform on the quantization noise masks. Thus, the diffusion weights are computed according to the shift theorem of the Fourier transform:

$$\begin{aligned} w(x,y) &= M_{\mathrm{s}}(x,y)(e^{-j2\pi(\xi_{\mathrm{tr}}x+\eta_{\mathrm{tr}}y)} + e^{j2\pi(\xi_{\mathrm{tr}}x+\eta_{\mathrm{tr}}y)}) + M_{\mathrm{hf}}(x,y) \\ &= 2M_{\mathrm{s}}(x,y)\cos(2\pi(\xi_{\mathrm{tr}}x+\eta_{\mathrm{tr}}y)) + M_{\mathrm{hf}}(x,y), \end{aligned} \tag{1}$$

where $M_{\mathrm{s}}(x,y)$ and $M_{\mathrm{hf}}(x,y)$ are the Fourier transform of $m_{\mathrm{s}}(\xi,\eta)$ and $m_{\mathrm{hf}}(\xi,\eta)$, respectively. To reduce the computational load, $M_{\mathrm{s}}(x,y)$ and $M_{\mathrm{hf}}(x,y)$ are pre-computed. During the last stage, the SWED algorithm is applied on the current hogel. This process is summed up in Fig. 1.

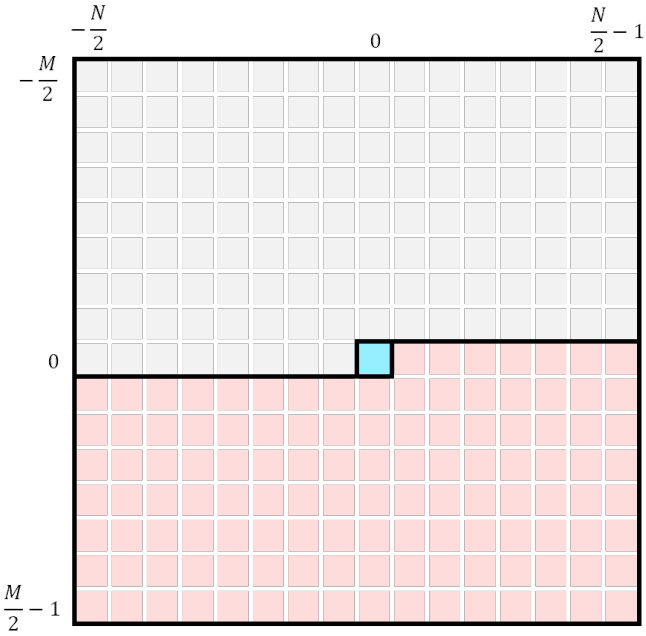


Figure 2: Localisation of the selected diffusion weights with a lexicographic order. The blue square corresponds to the position of the processed pixel, its coordinates in $w(x, y)$ are (0,0). The red squares are the selected diffusion weights.

3 GPU implementation

3.1 Weight selection

As presented in Eq. (1), for each hogel a set of $N \times M$ diffusion weights are available, where N and M are respectively the width and the height of a hogel. However, the quantization error of a pixel is only diffused to its neighbours that have not already been processed. Thus, according to the order in which the pixels are accessed, the weights selected for the diffusion vary. Several processing orders have been presented such as the random order, the lexicographic order and the bidirectional order [17, 18]. To maximize the parallelism of our approach, the pixels are accessed in a lexicographic order, which means that lines are processed one by one from left to right. Moreover, as explained in [18], the value of coordinate (0,0) in the weight function $w(x, y)$ corresponds to the processed pixel. The other values of this function weight the quantization error diffused to the corresponding neighbours. Therefore, the coordinates of the weights used during the diffusion respect either:

$$0 < y < \frac{M}{2} \quad \text{and} \quad \frac{-N}{2} \leq x < \frac{N}{2}, \tag{2}$$

or

$$y = 0 \quad \text{and} \quad 0 < x < \frac{N}{2}. \tag{3}$$

These inequalities are illustrated in Fig. 2.

Furthermore, in order to speed up the calculation, only the coefficients with the highest moduli are kept for the diffusion process. However, $|w(x, y)|$ varies according to $(\xi_{\text{tr}}, \eta_{\text{tr}})$ which implies that this selection has to be done for each hogel. This operation might take time for a large set of coefficients. Therefore, by observing that

$$\max_{\xi_{\text{tr}},\eta_{\text{tr}}} |w(x, y)| = 2|M_{\text{s}}(x, y)| + |M_{\text{hf}}(x, y)|, \tag{4}$$

all the weights that are at least significant for one hogel are pre-selected before iterating over them. Indeed, after determining the reference masks, their spectrum $M_{\text{s}}(x, y)$ and $M_{\text{hf}}(x, y)$ are known, which allows to compute $\max_{\xi_{\text{tr}}\eta_{\text{tr}}} |w(x, y)|$ and to limit the calculation in the loop.

To sum up, before looping over the hologram blocks, two types of coefficients are chosen: useful weights according to the processing order and weights significant for at least one hogel. Then, a second selection is done for each hogel over this restricted set of weights. This pipeline is presented in Fig. 3. Moreover, the set of coefficients associated to each hogel is unknown before the calculation starts because they vary according to the computation parameters.

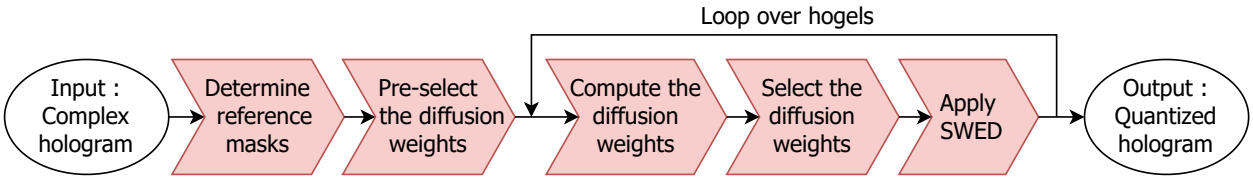
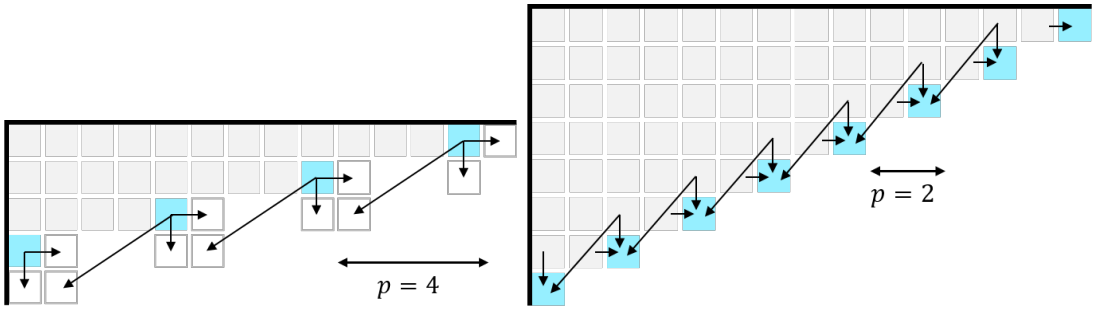


Figure 3: Block diagram of the VDED algorithm completed with the selection operations.



(a) Error diffusion (b) Error collection

Figure 4: Inherent parallelism of (a) the error diffusion and (b) the error collection algorithms according to a random set of weights. Blue squares are the pixels being quantized and gray pixels are already processed. The quantization errors are diffused and collected according to the black arrows.

3.2 Inherent parallelism

In this section, we generalize the work presented in [14, 15] for a set of weights whose size and layout are unknown ahead of compile time. Since the hologram pixels are accessed in a lexicographic order, to parallelize the computation each thread is assigned to a line of the signal. In CUDA, threads are split into blocks and are executed concurrently. The thread k assigned to line k starts processing the pixel i when thread $k-1$ is quantizing pixel $p+i$, where $k \in [\![1, M-1]\!]$ and $i \in [\![0, N-p-1]\!]$. Hence, the variable p is the pixel offset between two consecutive lines. This value is defined by the set of diffusion weights which depends on the quantization masks. In other words, p cannot be set directly by the user. Moreover, the smaller p is, the more parallelism is allowed. An example with a random set of weights is illustrated in Fig. 4a, where $p = 4$. In [14], the authors introduced a new diffusion process called error collection. Instead of quantizing a pixel and transmitting its quantization error as in the classic algorithm, in this method a pixel being quantized collects the errors of its neighbours as shown in Fig. 4b. This approach allows for more parallelism than the error diffusion technique because the hologram pixels can be concurrently read but not concurrently updated.

CUDA is a Single Instruction Multiple Threads (SIMT) programming model which means that groups of n_w threads, called wraps, are executed simultaneously. To leverage this inherent synchronization, authors of [14] proposed to create groups of n_w hologram lines that run synchronously. Each set of lines quantizes d pixels sequentially and then checks the progress of the other groups to process the d following pixels. Thus, each wrap executes $\frac{N+pn_\mathrm{w}}{d}$ tasks before exiting. For the algorithm to run correctly, the set of lines k starts computing task i only when the group $k-1$ is executing its task $\frac{pn_\mathrm{w}}{d} + i + 1$, where $k \in [\![1, \frac{M}{n_\mathrm{w}} - 1]\!]$ and $i \in [\![0, \frac{N}{d} - 2]\!]$. In Fig. 5, an example illustrates this operation.

The synchronization of the wraps relies on the SKSS approach [15] which uses the inter-block communication technique presented in [16]. This method uses a while loop and an array of bits in the global memory: threads update the array as soon as a task is finished which unlocks other threads that start a new task. The SKSS technique launches a single parallel function, called kernel in CUDA, which minimizes the accesses to the global memory. To execute a task, a warp needs to access the hologram pixels and their associated quantized values. Both data arrays have a parallelogram shape in the global memory and are divided into four parts: current pixels, top neighbours, left neighbours and unused pixels. An example in Fig. 6a illustrates this data layout when $p = 2$. The size of this parallelogram varies according to the distance between the farthest top/left neighbour and the current pixel, h and w respectively. This block has a height of

$$P_h = n_\mathrm{w} + h, \tag{5}$$

and a width of

$$P_w = d + w + p \times h. \tag{6}$$

At the beginning of a new task, the two parallelograms are loaded in the shared memory in order to limit memory transfers. The data layout in the shared memory corresponds to a row shift of Fig. 6a with a padding area to

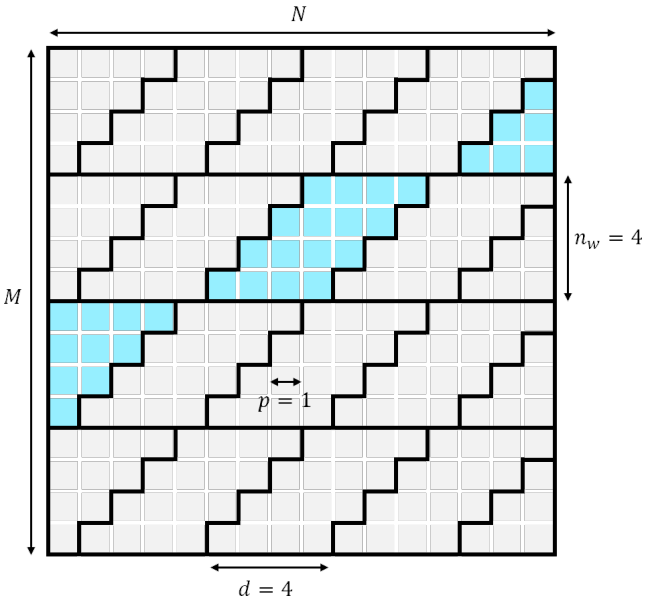


Figure 5: Dependency link between tasks of each set of lines. Each task of d pixels is delimited by black lines. Blue tasks are executed concurrently.

avoid memory bank conflicts as shown in 6b. The hologram pixels of the left neighbours and their quantized values are already in the shared memory as they are a product of the previous task. The top neighbours information are transferred by the upper wrap across the global memory. Finally, the hologram values of the pixels being quantized are also loaded from the global memory. In this way, the required information within a wrap is only loaded once from the global memory, which is a major time saving.

Several optimizations presented in [14] were not generalizable to a set of undefined weights such as the register caching technique. Indeed, in our approach, the access pattern is not known at compile time which forbids its use. Furthermore, this algorithm is embedded into a hologram generation method, imposing input and output data types which are not encoded on 32 bits and prevent all the global memory accesses to be coalesced.

3.3 Increased parallelism

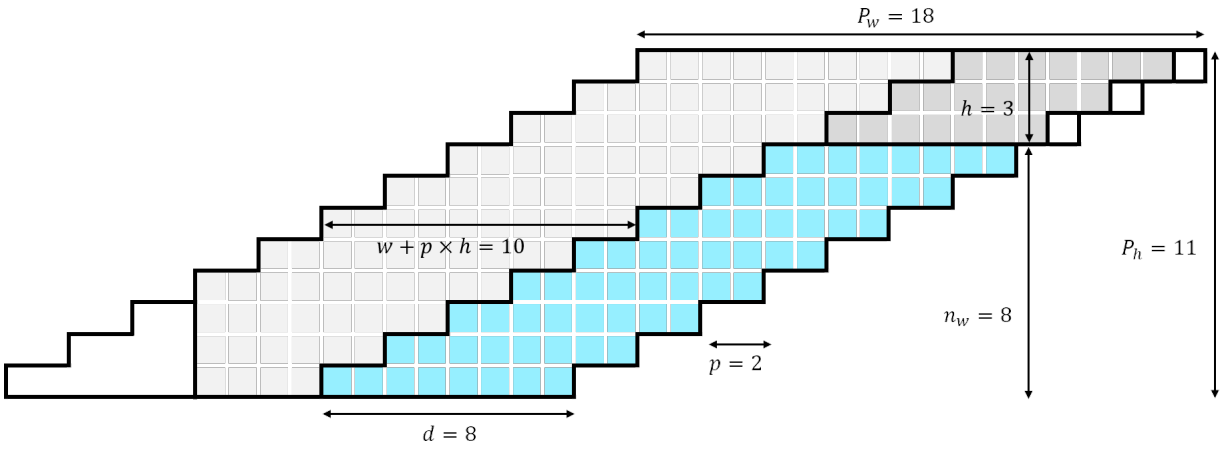
In the method presented in the previous section, p depends on the quantization masks and cannot be set directly. In this way, the parallelism of the algorithm cannot be adjusted. To overcome this issue, we propose to restrict the pixels from which a quantization error is collected. Hence, the set of available coefficients for the collection process is limited. The area in which diffusion weights are selected is defined as follows:

$$0 \leq y < \frac{M}{2} \quad \text{and} \quad -yp < x < \frac{N}{2}. \tag{7}$$

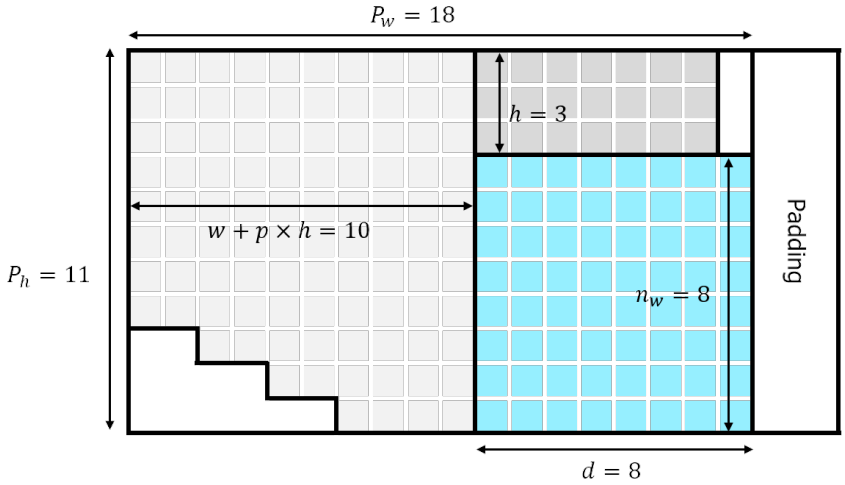
Fig 7 illustrates this region for $p = 2$. This process allows a trade-off between the computation time and the quality of the reconstructions. Indeed, by restricting the selection area of the diffusion weights, some coefficients with a high modulus might be excluded, decreasing the reconstruction quality. Nevertheless, this selection guarantees a specific parallelism capacity and the possibility to speed up calculations if needed.

For this technique to work, at least a part of the coefficients with a high modulus must be located in the region of selection. To check in which conditions this requirement is met, the weight function, $w(x, y)$, is studied. By considering $m_{\mathrm{s}}(\xi, \eta)$ and $m_{\mathrm{hf}}(\xi, \eta)$ as squared binary masks, their spectrum are expressed as follows:

$$\begin{aligned} M_{\mathrm{s}}(x, y) &= \mathcal{F}\left\{m_{\mathrm{s}}(\xi, \eta)\right\}(x, y) \\ &= \mathcal{F}\left\{\mathrm{rect}\left(\frac{\xi}{\zeta_x}, \frac{\eta}{\zeta_y}\right)\right\}(x, y) \\ &= |\zeta_x \zeta_y| \,\mathrm{sinc}(\zeta_x \pi x)\,\mathrm{sinc}(\zeta_y \pi y), \end{aligned} \tag{8}$$



(a) Global memory



(b) Shared memory

Figure 6: Representation of a parallelogram in (a) the global memory and in (b) the shared memory when $p = 2$. Blue squares are the pixels being quantized, light gray pixels are the left neighbours, dark gray pixels are the top neighbours and white areas are unused.

and

$$\begin{aligned} M_{\text{hf}}(x,y) &= \mathcal{F}\left\{m_{\text{hf}}(\xi,\eta)\right\}(x,y) \\ &= \mathcal{F}\left\{1 - \text{rect}\left(\frac{\xi}{\kappa_x}, \frac{\eta}{\kappa_y}\right)\right\}(x,y) \\ &= \begin{cases} \delta(x,y) & \text{if } (x,y) = (0,0) \\ -|\kappa_x \kappa_y| \,\text{sinc}(\kappa_x \pi x)\,\text{sinc}(\kappa_y \pi y) & \text{otherwise,} \end{cases} \end{aligned} \tag{9}$$

where (ζ_x, ζ_y) is the size of the area seen by an observer through a hogel and (κ_x, κ_y) defines the region which limits the highest frequencies. The function $w(x,y)$ is a difference of real sinc functions everywhere except in $(0,0)$ where it is equal to a Dirac pulse. Hence, the coefficients with the highest moduli are located around the pixel $(0,0)$. Therefore, at least some significant weights are kept regardless of the parameter p and of the size of the quantization masks, which demonstrates the proper functioning of our method.

4 Experimental results

4.1 Scene and hologram calculation parameters

We consider phase-only holograms whose generation parameters are described in Table 1. The fringe size controls the maximum diffraction angle separately from the pixel pitch. After several tests, a hogel size of 2048 × 2048 pixels appears as a good balance between angular and spatial resolutions. In the experiments, all the view-specific holograms are quantized on 8 bits except the reference which is a complex hologram with floating point values. All the VDED techniques pre-select 27 weights before the collection process. The approach

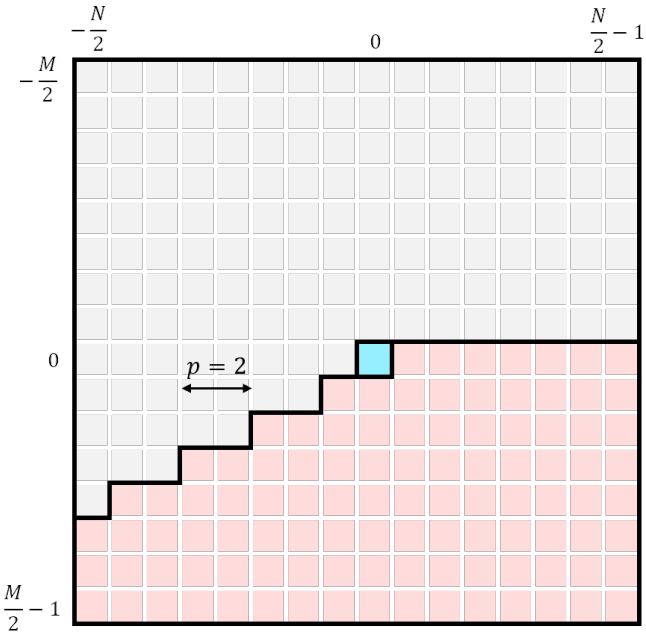


Figure 7: Localisation of the diffusion weights for $p = 2$. The blue square corresponds to the position of the processed pixel in $w(x, y)$ and red squares are the available diffusion weights.

Parameter	Value
Pixel pitch	750nm
Fringe size	1.5µm
Field of view	25°
Hogel size	2048 × 2048 pixels
Wavelength	635nm

Table 1: Parameters of the generated holograms.

without VDED quantizes the hologram pixels with a truncation method. Moreover, the test scene is composed of 3 dices and a checkerboard as described in Table 2.

4.2 Numerical computation

Our method was implemented as a C++ Unity plugin on a PC system including an Intel Core i9-9900X operating at 3.5GHz, a 32GB Random Access Memory (RAM) and a NVIDIA GeForce RTX 2080 Ti graphics card. To evaluate the speed up brought by our approach, we compared it with the CPU implementation presented in [8].

Table 3 shows that the execution time of the GPU approach is up to 117 times faster than the CPU implementation. Hence, as each method uses the same number of weights to diffuse the quantization errors, it proves the efficiency of our GPU implementation. Moreover, Table 3 demonstrates that our approach is able to adjust its execution time by imposing a specific p. Indeed, the VDED technique with $p = 1$ is 3 times faster than the one with $p = 6$.

4.3 Numerical reconstruction

To reconstruct the 3D scene from the hologram, the light waves are first propagated to the viewer plane with the angular spectrum. Then, the light field is cropped to fit the observer viewpoint and is back-propagated to the reconstruction plane. The VDED reconstructions are less bright than the one without VDED because the quantization noise is rejected outside of the visualisation window. Hence, the reconstructions of Fig. 8 are normalized by the sum of their mean value and their standard deviation.

Fig. 8 shows reconstructions of the 3D scene for different quantization methods. The reconstructions from Fig. 8c to Fig. 8f come from holograms quantized with the VDED technique by using the same number of weights. However, their values and coordinates vary according to the approach which has an impact on the reconstruction quality. The contrast in Fig. 8d is poor as the light intensity is mainly located on the edges of the image. At the opposite, for $p = 3$, the contrast on the bottom dice is greatly improved. The reconstructions of the VDED CPU and the VDED $p = 6$ are similar to the reference image which proves their high-quality. As the variable p decreases, the quality of the reconstructions is also reducing. Indeed, the selection region of the

Object	Size (mm)	Depth (cm)
Hologram	18.4 × 18.4	0
Viewing area	9.2 × 9.2	-3.6
Checkerboard	18.4 × 18.4	10.3
Top left dice	6 × 6 × 6	9.7
Bottom dice	8 × 8 × 8	9.1
Top right dice	6 × 6 × 6	8.1

Table 2: Parameters of the 3D scene.

Approach	Value of p	Time (s)
GPU VDED	1	0.164
GPU VDED	3	0.304
GPU VDED	6	0.507
CPU VDED [8]	/	19.24

Table 3: Execution time of the different VDED techniques.

diffusion weights becomes tighter and some significant coefficients cannot be used anymore. Moreover, since the quality of Fig. 8e is similar to the quality of Fig. 8f and Fig.8c, it demonstrates that a restricted selection region still allows to enhance remarkably the reconstruction quality. Furthermore, the worst reconstruction is the one without VDED which proves the usefulness of our technique.

5 Conclusion

In this paper, we propose a GPU implementation of the view-dependent error diffusion algorithm. Our method exploits the inherent parallelism of the error diffusion algorithm and generalizes the work presented in [14, 15] to a set of diffusion weights unknown before compile time. Moreover, we introduce a new weight selection approach which allows a trade-off between execution time and reconstruction quality.

Experimental results show that our GPU implementation is up to 117 times faster than the CPU algorithm. Moreover, numerical reconstructions demonstrate that the VDED GPU algorithm is able to quickly generate high-quality holograms by a relevant choice of diffusion weights.

The register caching technique could not be used in the VDED algorithm because the coefficient access pattern is not known ahead of compile time. Furthermore, our implementation of the VDED technique is embedded in a hologram generation plugin which prevents several memory accesses to be coalesced due to incompatible data types.

References

[1] Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., and Ivkovic, M., "Augmented reality technologies, systems and applications," *Multimedia Tools and Applications* **51**, 341–377 (Jan. 2011).

[2] Wheatstone, C., "Contributions to the Physiology of Vision.–Part the First. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision," *Philosophical Transactions of the Royal Society of London* **128**, 371–394 (Jan. 1838).

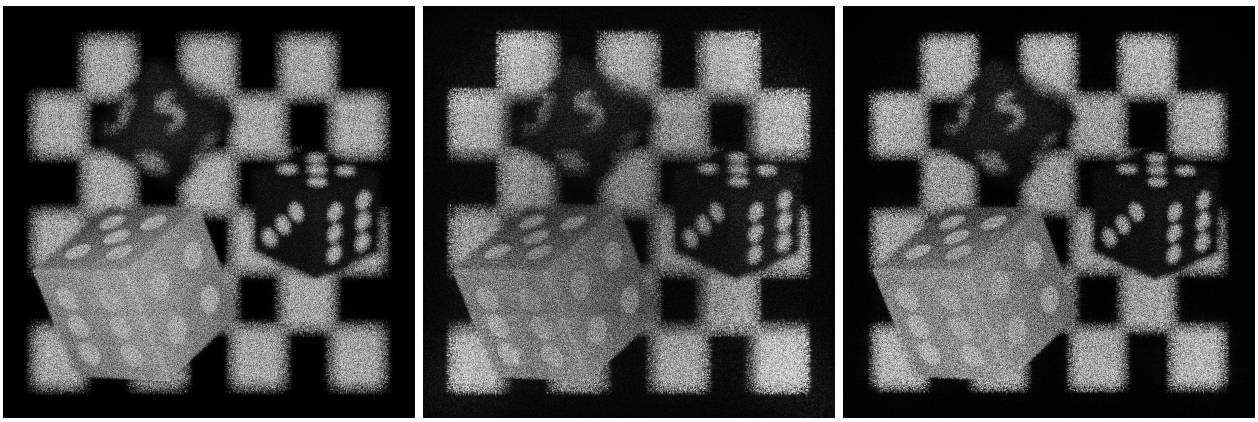
[3] Hoffman, D. M., Girshick, A. R., Akeley, K., and Banks, M. S., "Vergence–accommodation conflicts hinder visual performance and cause visual fatigue," *Journal of Vision* **8**, 33 (Mar. 2008).

[4] Schnars, U. and Jüptner, W., [*Digital Holography: Digital Hologram Recording, Numerical Reconstruction, and Related Techniques*], Springer Science & Business Media (Dec. 2005).

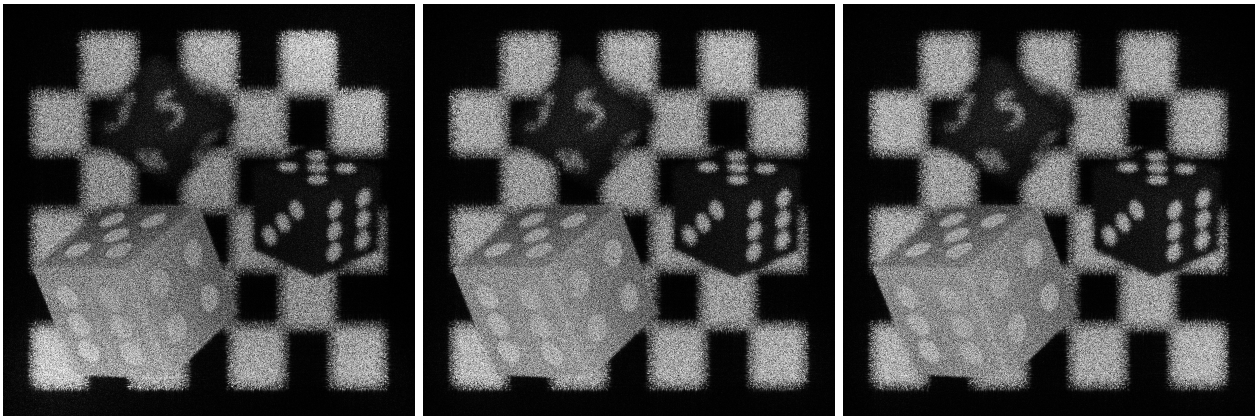
[5] Yaraş, F., Kang, H., and Onural, L., "State of the Art in Holographic Displays: A Survey," *Journal of Display Technology* **6**, 443–454 (Oct. 2010).

[6] Pang, H., Wang, J., Zhang, M., Cao, A., Shi, L., and Deng, Q., "Non-iterative phase-only Fourier hologram generation with high image quality," *Optics Express* **25**, 14323 (June 2017).

[7] Liu, K., He, Z., and Cao, L., "Pattern-adaptive error diffusion algorithm for improved phase-only hologram generation," *Chinese Optics Letters* **19**(5), 050501 (2021).



(a) Reference (b) w/o VDED (c) VDED CPU



(d) VDED $p = 1$ (e) VDED $p = 3$ (f) VDED $p = 6$

Figure 8: Reconstructions of the 3D scene according to different quantization techniques.

[8] Lagrange, A., Gilles, A., Heggarty, K., and Fracasso, B., "Fast view-specific generation of high definition holograms with enhanced quality," *submitted to Optics Express* (2024).

[9] El Bouz, M. and Heggarty, K., "Signal window minimum average error algorithm for multi-phase level computer-generated holograms," *Optics Communications* **180**, 21–28 (June 2000).

[10] Allebach, J. P., "Block interlaced pinwheel error diffusion," *Journal of Electronic Imaging* **14**, 023007 (Apr. 2005).

[11] Zhang, Y., Recker, J. L., Ulichney, R., Beretta, G. B., Tastl, I., Lin, I.-J., and Owens, J. D., "A parallel error diffusion implementation on a GPU," 78720K (Jan. 2011).

[12] Metaxas, P. T., "Parallel digital halftoning by error-diffusion," in [*Proceedings of the Paris C. Kanellakis memorial workshop on Principles of computing & knowledge: Paris C. Kanellakis memorial workshop on the occasion of his 50th birthday*], 35–41, ACM, San Diego California USA (June 2003).

[13] Floyd, R. W. and Steinberg, L., "An adaptative algorithm for spatial grayscale," *Proceedings of the Society for Information Display* **17**, 75–77 (1976).

[14] Kasagi, A., Nakano, K., and Ito, Y., "Parallelization Techniques for Error Diffusion with GPU Implementations," in [*2015 Third International Symposium on Computing and Networking (CANDAR)*], 30–39, IEEE, Sapporo, Hokkaido, Japan (Dec. 2015).

[15] Funasaka, S., Nakano, K., and Ito, Y., "Single Kernel Soft Synchronization Technique for Task Arrays on CUDA-enabled GPUs, with Applications," in [*2017 Fifth International Symposium on Computing and Networking (CANDAR)*], 11–20, IEEE, Aomori (Nov. 2017).

[16] Shucai Xiao and Wu-chun Feng, "Inter-block GPU communication via fast barrier synchronization," in [*2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*], 1–12, IEEE, Atlanta, GA (Apr. 2010).

[17] Tsang, P. W. M. and Poon, T.-C., "Novel method for converting digital Fresnel hologram to phase-only hologram based on bidirectional error diffusion," *Optics Express* **21**, 23680–23686 (Oct. 2013). Publisher: Optica Publishing Group.

[18] Heggarty, K. and Chevallier, R., "Signal window minimum average error algorithm for computer-generated holograms," *JOSA A* **15**, 625–635 (Mar. 1998). Publisher: Optica Publishing Group.