Variable *k*-buffer using Importance Maps

A. A. Vasilakis*†[1] and K. Vardis†[2] and G. Papaioannou†[2] and K. Moustakas[3]

[1]Information Technologies Institute, Centre for Research & Technology, Greece
[2]Department of Informatics, Athens University of Economics & Business, Greece
[3]Electrical and Computer Engineering Department, University of Patras, Greece

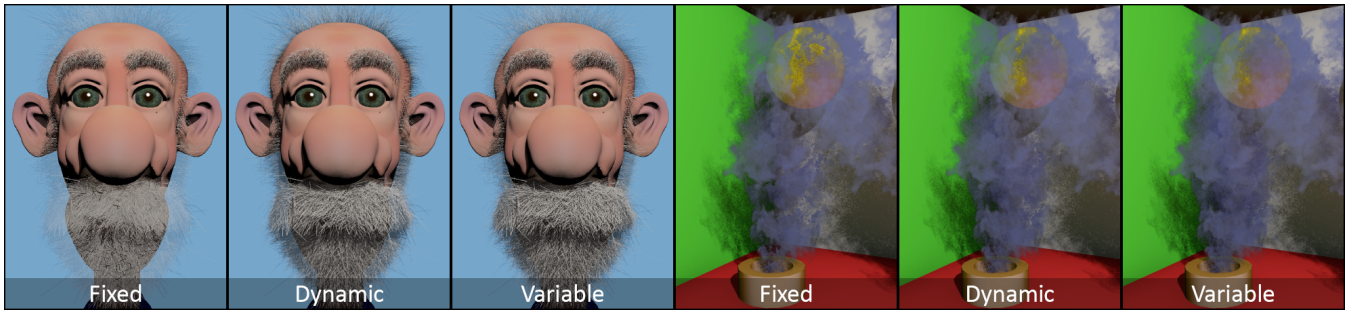

Figure 1: *Our approach, in contrast to current k-buffer approaches (fixed and dynamic), adaptively adjusts k on a per-pixel level, with respect to regions of the image that are deemed to be important, thus allowing for better utilization of the allocated storage space in the same memory budget and reduction of view-dependent artifacts.*

Abstract

Successfully predicting visual attention can significantly improve many aspects of computer graphics and games. Despite the thorough investigation in this area, selective rendering has not addressed so far fragment visibility determination problems. To this end, we present the first "selective multi-fragment rendering" solution that alters the classic k-buffer construction procedure from a fixed-k to a variable-k per-pixel fragment allocation guided by an importance-driven model. Given a fixed memory budget, the idea is to allocate more fragment layers in parts of the image that need them most or contribute more significantly to the visual result. An importance map, dynamically estimated per frame based on several criteria, is used for the distribution of the fragment layers across the image. We illustrate the effectiveness and quality superiority of our approach in comparison to previous methods when performing order-independent transparency rendering in various, high depth-complexity, scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Visible surface algorithms I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures

1. Introduction

Multi-fragment rendering (MFR) is increasingly becoming an important aspect of real-time image generation for supporting complex effects in games such as order-independent transparency (OIT) [MCTB11] and image-based global illumination [VVP16]. However, practical implementations must severely constrain both the memory budget and the computation time for the depth-sorted fragment determination, leading to the adoption of bounded memory MFR configurations such as the *k-buffer* [BCL*07].

The *k*-buffer assumes a pre-assigned, and global, value of *k* fragment layers across the entire image. Unfortunately, this standard practice of employing a fixed number of *k* for all pixels can lead to various quality and memory issues. On one hand, setting *k* to a small number, can result in view-dependent artifacts as more than *k* fragments might be required for some pixels at a particular viewing configuration to correctly simulate the desired effect. On the other

* abasilak@iti.gr
† These authors contributed equally to this work

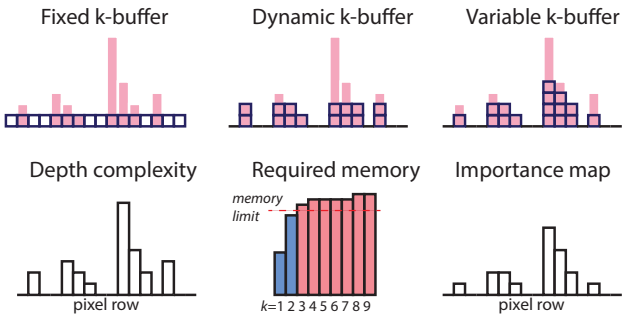

Figure 2: *A representative example demonstrating the three compared k-buffer allocation strategies (top row). In the bottom row, we show the respective actual depth complexity, the required memory for a fixed k across the image and an importance map.*

hand, employing a large value of k can result in excessive memory demands, as a large and potentially unused storage is pre-allocated for pixels that contain less than k fragments. Clearly, regardless of the choice of k, all approaches that use a single value for k across the entire image result in bad utilization of the allocated space.

The key idea in this work is that, given a fixed memory budget for MFR, the value of k can be dynamically assigned on a *per-pixel* level, according to a *pixel importance distribution function*, which determines the depth complexity for each pixel based on its estimated importance and without exceeding the pre-allocated storage space. This importance distribution is calculated in image-space, stored in an *importance map*, and passed on to the k-buffer before the generation of the fragment data. The importance map can be statically determined once (e.g., for FOV-based importance in VR/AR applications and games) or calculated using any number of metrics, perceptual or otherwise, at each frame. To the best of our knowledge, this is the first k-buffer implementation that dynamically adjusts and distributes k across the image according to the available storage space.

2. Related Work

When low graphics memory requirements are of the utmost importance, the **fixed k-buffer** [BCL*07] with its more recent variations [Sal13, MCTB13] and extensions [HBT14, VP15] can objectively be considered a viable alternative to the full A-buffer-style MFR, which at least ensures the correct depth order on the closest subset of all generated fragments [MCTB11]. Relying on an iterative trial-and-error procedure, however, where the user manually configures the value of k, can inevitably result in (i) bad memory utilization and (ii) view-dependent artifacts.

The k^+-buffer [VPF15], a **dynamic k-buffer**, solved both issues by performing, first, a dynamic and precise memory allocation strategy (inspired by the S-buffer [VF12]), tailoring storage utilization to the depth complexity of individual pixels, and secondly, an intuitive and automatic method for the optimal estimation of k value under constrained memory budget via on-the-fly depth complexity histogram analysis.

Table 1: *Overview of the most well-known k-buffer alternatives.*

Method	[MCTB13, Sal13]	[VPF15]	Ours
k (Selection)	Fixed (Manual)	Fixed (Histogr.)	Variable
Distribution	Uniform	Uniform	Non-uniform
Memory	Fixed	Adaptive	Adaptive
Performance	High	Average	Average

In contrast to previous approaches, where the manually or automatically chosen k is considered to be *the same for all pixels*, (see Figure 2), our **variable k-buffer** dynamically distributes the available MFR storage across the image in a non-uniform manner. Therefore, our importance-based approach is aiming at optimizing the fragment distribution in the allocated space, rather than minimizing it, like the k^+-buffer approach. Table 1 presents a comparative summary of the most significant k-buffer variations with respect to memory requirements, rendering complexity and fragment extraction distribution.

3. Method Overview

3.1. Variable k-buffer Pipeline

Our algorithm operates in three main steps, as illustrated in Figure 3. Note that we consider the memory fixed, pre-allocated (contrary to dynamic k^+-buffer which is reallocated in every frame) and linearly organized into variable contiguous regions of length $k(\mathbf{p})$ for each pixel $\mathbf{p}$. Similar to the S-buffer [VF12], an additional, fast, geometry rendering pass is responsible for the per pixel fragment accumulation process via atomic counters or blending operations (**Per-Pixel Count pass**). Note that this stage is specific to the importance metric we opted to employ (see Sec. 3.2, Eq. 3). Then, the importance map values $I(\mathbf{p})$ are dynamically determined in a full-screen quad pass (**Importance Estimation pass**). We leave the derivation and discussion about $I(\mathbf{p})$ for subsection 3.2. Since we only know $I(\mathbf{p})$ for each pixel in isolation, we also accumulate the unnormalized values in an atomic value I_{tot} to derive an importance-based probability $P(\mathbf{p}) = I(\mathbf{p})/I_{tot}$. If M is the desired k-buffer memory size, the $k(\mathbf{p})$ of our variable k-buffer is finally estimated as (**Per-pixel k Calculation pass**):

$$k(\mathbf{p}) = \lfloor P(\mathbf{p}) \cdot M \rfloor. \tag{1}$$

It is often more convenient and intuitive for the sake of comparison to consider $k(\mathbf{p})$ with respect to an average image layer depth k_{ave} instead of the total requested memory storage M of the k-buffer, in which case Eq. 1 becomes:

$$k(\mathbf{p}) = \lfloor P(\mathbf{p}) \cdot k_{ave} \cdot w \cdot h \rfloor, \tag{2}$$

where w, h are the dimensions of the frame buffer. Finally, any k-buffer alternative that exploits pixel synchronization can be employed, e.g., the PixelSync [Sal13] or the k^+-buffer [VPF15], for dynamic depth-sorted fragment determination (**Synchronized k-buffer pipeline**).

3.2. Determining Importance

In our implementation, we estimate the unnormalized importance $I(\mathbf{p})$ of each pixel $\mathbf{p}$ based on three general, but highly sensitive to

older people, factors: (i) I_d, the expected depth complexity, (ii) I_{per}, the distance of pixel from the fovea region, and (iii) I_f, the Fresnel term of the nearest fragment, as registered in the the previous frame.

The **depth-complexity heuristic** is the ratio of the number of visible layers N_{layers} that should be drawn at pixel $\mathbf{p}$ against a rough anticipated maximum layer count N_{limit}:

$$I_d = \alpha + (1-\alpha)\min\{1, N_{layers}/N_{limit}\} \quad (3)$$

where α is a range bias that assists pixels with very low depth complexity and also dampens the prevalence of pixels with very high layer count in the importance distribution. We use an α value of $[0.25-0.5]$ in our experiments. I_d is linear with respect to depth complexity and expresses the raw fragment count demand. The exact value of N_{limit} is not crucial, since it is the same for all pixels and we typically set it to a scene-dependent large enough value ($20-30$ in the example scenes) so that the fraction is not saturated.

The **periphery heuristic** gradually lowers the significance of pixels towards the edges of the frame buffer (periphery vision) [GFD*12], by assuming that the information near the center of the image $\mathbf{o}$ is perceptually more important. In a more elaborate setup, the point of interest $\mathbf{o}$ could be derived from other mechanisms, such as an eye tracking input.

$$I_{per} = 1 - \left(\frac{2||\mathbf{p}-\mathbf{o}||}{1.1\sqrt{w^2+h^2}}\right)^d \quad (4)$$

In the above metric, d controls the rate of importance drop towards the periphery. We set d to 3 for all test cases. For steeper falloff near the edges of the image, d should be set to 10 or more.

Finally, the **Fresnel heuristic** is based on the observation that near the grazing viewing angle of a transparent surface, the reflected radiance prevails over the transmitted one and no matter how many transparent layers show through the respective fragments, the transmitted energy in a physically-based renderer will be significantly dimmer than the (more important) reflected light. The heuristic depends on the eye direction $\mathbf{e}$ and the normal $\mathbf{n}_{prev}$ at the closest visible MFR layer fragment at $\mathbf{p}$ from the previous frame, and is given by the following formula:

$$I_f = 1 - \frac{(1-\mathbf{n}_{prev}\cdot\mathbf{e})^5}{2} \quad (5)$$

The importance factor is the product of the above heuristics, since if at least one of the factors is sufficiently low, the transparency precision can be reduced. Additionally, a noise bias is added to mask potentially visible abrupt transition zones across otherwise smooth image regions,

$$I(\mathbf{p}) = \max\{0,\ \ I_{per}\cdot I_f \cdot I_d + \lambda\cdot(\xi-0.5)\}, \quad (6)$$

where $\xi \in [0,1]$ is a uniformly distributed scalar and λ the noise level, set to 0.1 across all our experiments. An example of the three heuristics and the combined importance map, scaled to maximum 1 for clarity, is shown in Figure 4.

It is important to note that the above selection of heuristics was based on the available information in our current rendering pipeline. Other attributes, such as low- or high-level saliency predictors [MMKI14], can potentially contribute to the above equation or even replace the factors involved.

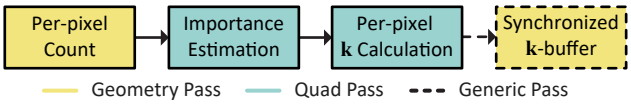

Figure 3: *The pipeline of the importance-based variable k-buffer.*

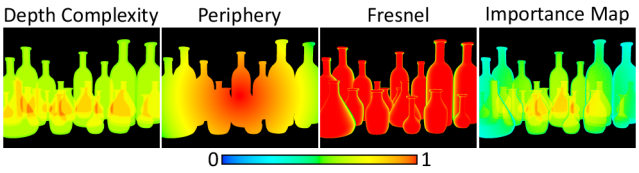

Figure 4: *Heatmaps of the three importance factors used in our pipeline as well as the final importance map for determining k.*

4. Experimental Study

We present an experimental analysis of our variable k-buffer approach when compared with both a fixed [MCTB13] and an adaptive one [VPF15] with respect to performance, image quality and memory budget under different testing configurations (see also a comparative overview in Table 1). All experiments were conducted on a 1024^2 viewport with varying memory demands M on an NVIDIA GTX980 Ti. We implemented all methods in modern OpenGL by integrating order-independent transparency effects into our deferred shading pipeline.

The visual impact of the three methods are shown in Figures 1 and 5. Table 2 presents performance measurements as well as fragment distribution capabilities of each approach under a fixed memory budget in all test scenes. The *Fragment Loss* value corresponds to the percentage of required fragments (for each scenario) that could not be allocated in the currently defined storage space.

Quality. The fixed k-buffer produced the lowest quality images (i.e., lowest k) due to its strategy to allocate memory for both empty as well as occupied fragments in the image, which also becomes evident by the large percentage of fragments that were not exploited within the budget M. The dynamic k-buffer was able to provide an increase in the number of layers, and therefore the quality of the image, due to its histogram based allocation strategy. The use of a global and fixed k value, however, constrain this method in scenes containing near-uniform depth complexity and/or low transparency objects (e.g., see Fig. 1 right). Finally, our variable k-buffer outperformed the aforementioned approaches due to its ability to assign a larger range of k values in the important regions of the image under the same memory budget (see k row in Table 2).

Performance. The fixed-k approach outperforms the other two methods due to its ability to exploit *atomicMin* hardware instructions during the k fragment allocation step, while the other two methods currently require the use of a spinlock mechanism, which is available either as an OpenGL extension for the NVIDIA Maxwell architecture or through software implementation [VPF15]. Furthermore, an (expected) additional overhead is imposed on the variable k-buffer due to the irregular fragmental complexity, compared to a fixed k value, that consequently increases the thread divergence during the fragment allocation stage.

Figure 5: *Quality evaluation between the three k-buffer approaches in a low (left) and a high depth complexity scenes (right) under different memory budget setups. Our method achieves comparable quality results when compared to the ground-truth images produced by an A-buffer implementation where all transparent fragments contribute to the final image synthesis [VF12].*

5. Conclusions

In this work, we have introduced the first k-buffer method that dynamically distributes the desired storage space instead of assuming the same value of k for all pixels. Our approach assigns k on a per-pixel basis according to importance-based distribution function, thus, allowing higher depth complexity in regions that are deemed important. In the future, we aim to explore further directions for enhancing various screen-space rendering problems, such as motion blur, depth-of-field and ray-tracing [VVP16], by investigating new importance maps (e.g., motion-based, eye tracking) [MMKI14].

Acknowledgements

The Beard Comic Character shown in Figure 1 was downloaded from www.cgtrader.com and is used under a royalty-free license. This research has received funding from the European Union's Horizon 2020 research and innovation programme "FrailSafe" under grant agreement No 690140 and the Athens University of Economics and Business Research Centre.

References

[BCL*07] BAVOIL L., CALLAHAN S. P., LEFOHN A., COMBA J. L. D., SILVA C. T.: Multi-fragment effects on the GPU using the k-buffer. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (NY, USA, 2007), I3D '07, ACM, pp. 97–104. 1, 2

[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: Foveated 3d graphics. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 1–10. 3

[HBT14] HILLESLAND K. E., BILODEAU B., THIBIEROZ N.: Deferred shading for order-independent transparency. In *Proceedings of Eurographics 2014 Short Papers* (France, 2014), EG '14, pp. 49–52. 2

[MCTB11] MAULE M., COMBA J. L., TORCHELSEN R. P., BASTOS R.: A survey of raster-based transparency techniques. *Computers & Graphics 35*, 6 (2011), 1023 – 1034. 1, 2

[MCTB13] MAULE M., COMBA J., TORCHELSEN R., BASTOS R.: Hybrid transparency. In *Proceedings of the 2013 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2013), I3D '13, ACM, pp. 103–118. 2, 3, 4

[MMKI14] MCNAMARA A., MANIA K., KOULIERIS G., ITTI L.: Attention-aware rendering, mobile graphics and games. In *ACM SIGGRAPH 2014 Courses* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 6:1–6:119. 3, 4

Table 2: *Measurements of the three tested k-buffer approaches in different testing scenarios under a fixed memory budget M. D denotes the scene's maximum depth complexity.*

Method	[MCTB13]	[VPF15]	Variable k-buffer
Beard scene, $D = 92$, $M \approx 20$MB, Fig. 1 (left)			
Time (ms)	1.4	7.3	11.8
k	1	1-4	1-7
Fragments (Loss)	232K (87%)	714K (60%)	840K (53%)
Smoke scene, $D = 16$, $M \approx 35$MB, Fig. 1 (right)			
Time (ms)	0.8	5.4	6.5
k	2	1-4	1-5
Fragments (Loss)	1M (62%)	1.4M (47%)	1.7M (39%)
Bottles scene, $D = 22$, $M \approx 35$MB, Fig. 5 (top)			
Time (ms)	1	5.7	6.8
k	2	1-3	1-5
Fragments (Loss)	1M (70%)	1.4M (55%)	1.8M (46%)
Greenhouse scene, $D = 106$, $M \approx 35$MB, Fig. 5 (bottom)			
Time (ms)	15	47	58
k	3	1-4	1-10
Fragments (Loss)	1.9M (80%)	2.5M (74%)	2.7M (72%)

[Sal13] SALVI M.: Pixel synchronization: Solving old graphics problems with new data structures. In *ACM SIGGRAPH 2013 Courses* (New York, NY, USA, 2013), SIGGRAPH, ACM. 2

[VF12] VASILAKIS A. A., FUDOS I.: S-buffer: Sparsity-aware multi-fragment rendering. In *Proceedings of Eurographics 2012 Short Papers* (Cagliari, Sardinia, Italy, 2012), EG '12, pp. 101–104. 2, 4

[VP15] VASILAKIS A. A., PAPAIOANNOU G.: Improving k-buffer methods via occupancy maps. In *Proceedings of Eurographics 2015 Short Papers* (Zurich, Switzerland, 2015), EG '15, pp. 69–72. 2

[VPF15] VASILAKIS A. A., PAPAIOANNOU G., FUDOS I.: k^+-buffer: An efficient, memory-friendly and dynamic k-buffer framework. *IEEE Transactions on Visualization and Computer Graphics 21*, 6 (June 2015), 688–700. 2, 3, 4

[VVP16] VARDIS K., VASILAKIS A. A., PAPAIOANNOU G.: A multi-view and multilayer approach for interactive ray tracing. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, USA, 2016), I3D '16, ACM, pp. 171–178. 1, 4