Hybrid MBlur: A Systematic Approach to Augment Rasterization with Ray Tracing for Rendering Motion Blur in Games

Yu Wei Tan[a], Xiaohan Cui[b] and Anand Bhojan[c]
School of Computing, National University of Singapore, Singapore

Keywords: Real-time, Motion Blur, Ray Tracing, Post-processing, Hybrid Rendering, Games.

Abstract: Motion blur is commonly used in game cinematics to achieve photorealism by modelling the behaviour of the camera shutter and simulating its effect associated with the relative motion of scene objects. A common real-time post-process approach is spatial sampling, where the directional blur of a moving object is rendered by integrating its colour based on velocity information within a single frame. However, such screen space approaches typically cannot produce accurate partial occlusion semi-transparencies. Our real-time hybrid rendering technique leverages hardware-accelerated ray tracing to correct post-process partial occlusion artifacts by advancing rays recursively into the scene to retrieve background information for motion-blurred regions, with reasonable additional performance cost for rendering game contents. We extend our previous work with details on the design, implementation, and future work of the technique as well as performance comparisons with post-processing.

1 INTRODUCTION

We showcase a novel real-time hybrid rendering technique for the Motion Blur (MBlur) effect that combines ray tracing and post-processing, exploiting ray-traced information within a selection mask to reduce partial occlusion artifacts in post-processed MBlur. We provide a thorough illustration and evaluation of hybrid MBlur, extending our published work (Tan et al., 2020b) with the following additions:

- Extensive discussion of the design and implementation details of hybrid MBlur.
- Visual quality and performance evaluations of our technique in comparison to post-processing.
- Explanation of presented extensions and future work to the approach.

1.1 Background Information

1.1.1 Hybrid Rendering

Ray tracing is a common approach to produce realistic effects like glossy reflections, depth of field, and motion blur. However, its high computation time has limited its use mostly to offline rendering. On the other hand, many post-process techniques have been devised to approximate these effects on rasterized images to meet the constraint of interactive frame rates for games, albeit with limited visual quality.

Given recent developments in hardware acceleration, it is now the time to marry ray tracing with rasterization for games. Ray tracing can be employed to correct certain visual flaws originating from pure post-process approaches, while still adhering to the performance budgets of real-time rendering. This results in more realistic and convincing graphics while keeping the gaming experience interactive, enhancing the overall immersion of the player in the game.

1.1.2 MBlur

MBlur is the streaking or smearing effect of objects in the direction of relative motion with respect to the camera. It is employed in cinematics to emphasize the relative speed of moving objects, producing a blurring effect as shown in Figure 1. The MBlur effect has been used to express the speed of rapid movement, the disoriented state of mind of a character, or the dream-like quality of a scene.

In cameras, MBlur is produced with the choice of exposure time or shutter speed, which is the duration

[a] https://orcid.org/0000-0002-7972-2828
[b] https://orcid.org/0000-0002-6152-1665
[c] https://orcid.org/0000-0001-8105-1739

Tan, Y., Cui, X. and Bhojan, A.
Hybrid MBlur: A Systematic Approach to Augment Rasterization with Ray Tracing for Rendering Motion Blur in Games.
DOI: 10.5220/0010984700003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 1: GRAPP*, pages 115-125
ISBN: 978-989-758-555-5; ISSN: 2184-4321

Figure 1: MBlur effect. Image from authors.

for which the camera shutter is open per second. This duration directly correlates to the amount of time the camera sensor is exposed to light. Hence, when the exposure time is long, every resulting image does not represent an instantaneous moment but rather, an interval consisting of successive moments in time. An object moving in relation to the camera will therefore appear on multiple areas of the sensor over time, resulting in a directional blur effect of the object corresponding to its motion.

As described in Jimenez (2014), moving objects blur both outwards and inwards at their silhouettes, making the region around their silhouettes appear semi-transparent. Outer blur represents an object's blur into its neighbouring background while inner blur applies to the blur produced within the silhouette of the object itself. According to Cook et al. (1984), accurate motion blurring takes into account areas of background geometry occluded by any blurred foreground. Hence, realistic MBlur should have accurate partial occlusion by recovering true background colour in inner blur regions instead of approximating this colour like many post-process approaches. Background colour recovery in MBlur also helps to prevent inaccuracies between real and approximated backgrounds for sharp and blurred regions respectively. Hence, we employ ray tracing in our approach to retrieve the exact colour of occluded background.

2 RELATED WORK

2.1 Hybrid Rendering

The use of ray tracing to complement rasterization techniques to achieve high graphics quality while maintaining interactive frame rates has been attempted for several visual effects over the years. In Beck et al. (1981) and Hertel et al. (2009), shadow mapping is employed to mark shadow boundaries and reduce the actual number of shadow rays to be traced, of which only rays from regions deemed potentially inaccurate by shadow mapping are traced in Lauterbach and Manocha (2009) to generate precise and alias-free shadows. For realistic reflections, Macedo et al. (2018) invokes ray tracing only on pixels with reflections that cannot be solved with just screen space information. In Marrs et al. (2018), traditional temporal anti-aliasing (TAA) is also extended with ray-traced supersampling for regions with a high chance of TAA failure.

In Cabeleira (2010), diffuse illumination is computed via rasterization while reflections and refractions are handled by ray tracing. In more general pipeline approaches, Chen and Liu (2007) substitutes the generation of primary rays in recursive ray tracing (Whitted, 1979) with rasterization for improved performance. This approach is extended in Andrade et al. (2014) which adheres to a render time limit by tracing rays only for objects of the highest importance.

2.2 MBlur

Current implementations of MBlur include the use of an accumulation buffer. Within the exposure time when the camera shutter is set to be open, a series of discrete snapshots rendered at multiple time offsets from the actual time value of the image is integrated to produce MBlur such as in Korein and Badler (1983) and Haeberli and Akeley (1990). However, as discussed in McGuire et al. (2012), this approach is computationally expensive in terms of shading time. Furthermore, in the case of undersampling, this method can lead to a series of disjoint ghosting artifacts in the direction of motion instead of a continuous blur.

Stochastic sampling (Cook, 1986) for MBlur is a Monte Carlo technique based on Halton (1970) in which sampling is performed in a randomized and nonuniform manner along the time dimension. McGuire et al. (2010) incorporates stochastic sampling with ray tracing by conservatively estimating the total convex hull bound for the area of each moving triangle during the exposure time, leveraging hardware rasterization. Stochastically distributed rays are then shot into the scene within this bound for shading. This approach makes use of distributed ray tracing (Cook et al., 1984), where rays are distributed in time and the colour at each hit point is averaged to produce the final image.

Cook et al. (1987) translates micropolygons for each sample according to a jittered time. On the other hand, Fatahalian et al. (2009) and Sattlecker and Steinberger (2015) achieve efficiency in performance by extending the bounding box of each micropolygon to consider its pixels and their associated velocities in-

stead. Hou et al. (2010) is an approach that makes use of 4D hyper-trapezoids to perform micropolygon ray tracing. Methods of approximating the visibility function to sample have also been devised such as in Sung et al. (2002). However, as explained in Sattlecker and Steinberger (2015), at low sample rates, these methods will also exhibit ghosting artifacts whereas increasing the number of samples would lead to noise.

To produce the right amount of blur, some real-time approaches like Rosado (2007), Ritchie et al. (2010) and Sousa (2013) make use of per-pixel velocity information by accumulating samples along the magnitude and direction of velocities in the colour buffer. Other techniques in Korein and Badler (1983), Catmull (1984) and Choi and Oh (2017) accumulate the colours of visible passing geometry or pixels with respect to a particular screen space position while Gribel et al. (2011) makes use of screen space line samples instead. Potmesil and Chakravarty (1983) represents the relationship between objects and their corresponding image points as point-spread functions (PSFs), which are then used to convolve points in motion. Leimkühler et al. (2018) splats the PSF of every pixel in an accelerated fashion using sparse representations of their Laplacians instead. Time-dependent edge equations, as explained in Akenine-Möller et al. (2007) and Gribel et al. (2010), and 4D polyhedra primitives (Grant, 1985) have also been used for MBlur geometry processing. Recently, a shading rate-based approach involving content and motion-adaptive shading in Yang et al. (2019) has also been developed for the generation of MBlur.

In particular, attempts to simulate nonlinear MBlur include Gribel et al. (2013) and Woop et al. (2017). Our hybrid technique only considers linear inter-frame image space motion for now, but we intend to provide support for higher-order geometry motion in the future. We also assume mainstream ray tracing acceleration architecture widely available in modern gaming workstations. The GA10x RT Core of the newest NVIDIA Ampere architecture provides hardware acceleration for ray-traced motion blur (NVIDIA Corporation, 2021) but is only found in the premium GeForce RTX 30 Series graphics cards.

3 DESIGN

Our hybrid MBlur approach, as illustrated in Figure 2, compensates for missing information in post-processed MBlur with the revealed background produced by a ray trace-based technique.

A Geometry Buffer (G-Buffer) is first generated under a deferred shading set-up, rendering textures containing per-pixel information such as camera space depth, screen space velocity and rasterized colour. The same depth, velocity and colour information for background geometry is produced by our novel ray reveal pass within a ray mask for pixels in the inner blur of moving foreground objects. A tile-dilate pass is then applied to these 2 sets of buffers to determine the sampling range of our gathering filter in the subsequent post-process pass. Both the ray-revealed and rasterized output are blurred by this post-process pass and lastly composited together.

3.1 Post-process

The McGuire et al. (2012) post-process MBlur renders each pixel by gathering sample contributions from a heuristic range of nearby pixels. We adapt this approach to produce a motion-blurred effect separately for rasterized and ray-revealed information.

As suggested in Rosado (2007), motion vectors are given by first calculating per-pixel world space positional differences between every frame and its previous frame, followed by a translation to screen space. By using the motion vector between the last frame and the current frame, we simulate the exposure time of one frame. Then, we follow the approach of McGuire et al. (2012) in calculating the displacement of the pixel within the exposure time by scaling the inter-frame motion vector with the frame rate of the previous frame as well as the exposure time. Considering the full exposure as one unit of time, this displacement can be interpreted as a per-exposure velocity vector. This approach uses the assumption that the motion vector of each pixel between the previous frame and the current frame remains constant throughout the exposure time.

Jimenez (2014) is a technique that is based on McGuire et al. (2012). As described in Jimenez (2014), the major problems to be addressed when producing a post-processed MBlur are the range of sampling, the amount of contribution of each sample as well as the recovery of background geometry information for inner blur. With our method for calculating per-exposure velocities, we adopt McGuire et al. (2012)'s approach in determining the magnitude of the sampling range and representing different amounts of sample contribution, as illustrated in the Appendix for completeness. As shown in Figure 3, McGuire et al. (2012) centers the sampling area at the target pixel, creating a blur effect both outwards and inwards from the edge of the object. Although this produces a more uniform blur for thin objects and the specular highlights of curved surfaces, it poses the problem of having to smoothen the transition between

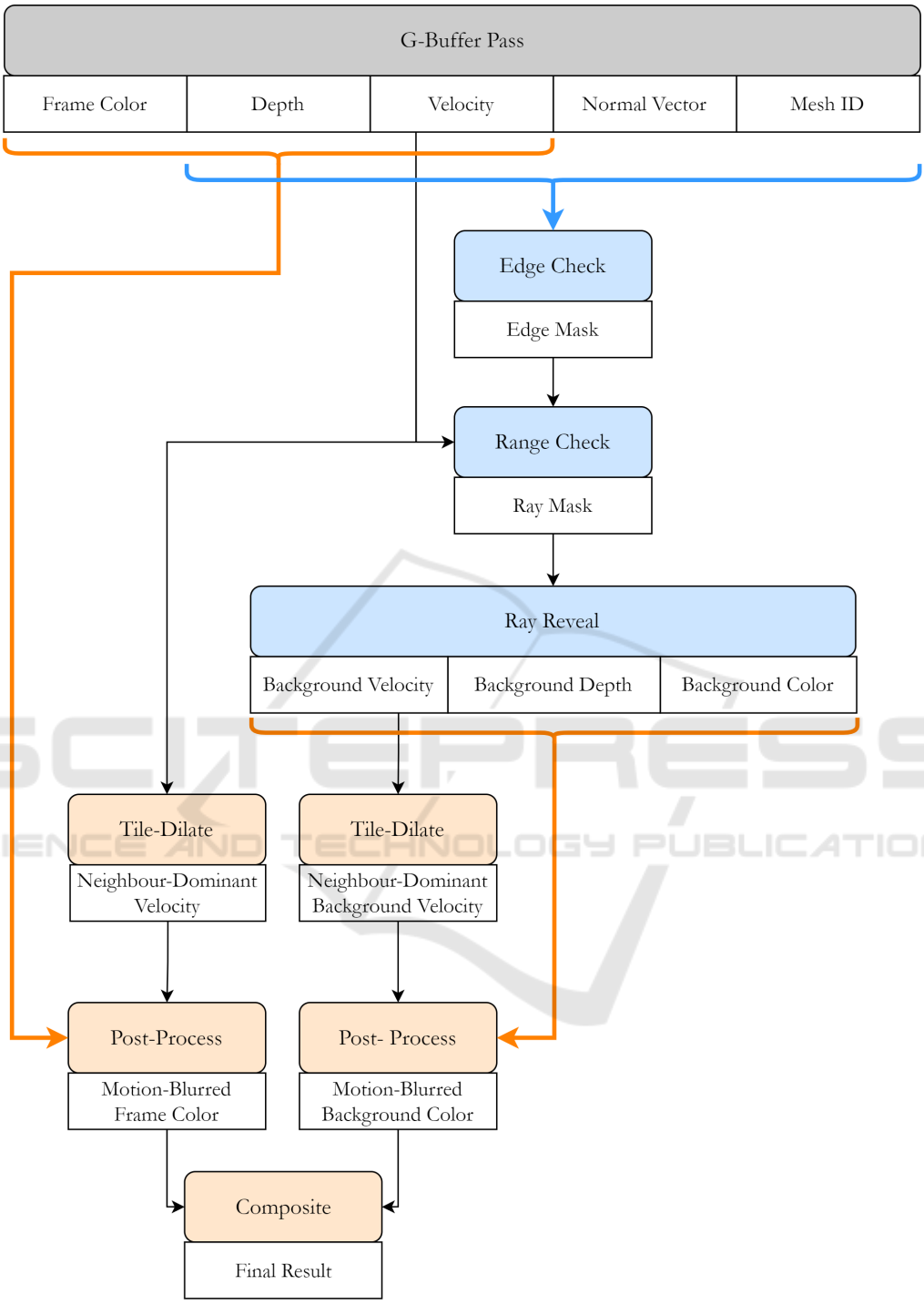


Figure 2: Hybrid MBlur rendering pipeline.

the inner and outer blur. Hence, we let the pixel be at the end of the sampling area instead, so inner and outer blur can be considered separately without much change to the pipeline. We also enhance the inner blur with ray-revealed background information.

3.2 Ray Reveal

McGuire et al. (2012) approximates the inner blur region of objects with colour information of nearby samples. This is based on the reasoning that approx-

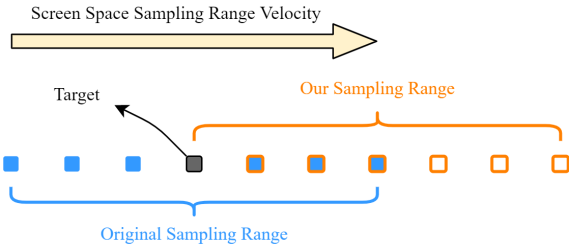


Figure 3: Range of samples for each pixel.

imate background information is still better than the absence of any background. With our ray reveal algorithm, we ray trace to obtain true background information for the inner blur.

3.2.1 Ray Mask

We adopt selective rendering (Chalmers et al., 2006) in generating a ray mask that determines regions of inner blur corresponding to geometry edges and only shoot rays within the mask for better performance like in adaptive frameless rendering (Dayal et al., 2005).

For every pixel with a nonzero speed, we first translate it by its velocity for one magnitude of its estimated displacement within the exposure time to predict its next screen space position at the end of the exposure duration and compare their mesh ID and depth. A target pixel is filtered out of the mask if the following conditions are not met:

$$m_n \neq m_t \tag{1}$$

$$z_n - z_t > \text{SOFT\_Z\_EXTENT} \tag{2}$$

m_n and m_t refer to the corresponding mesh ID of the pixel's next and current positions respectively. We assume that every object has uniform speed throughout its geometry, so we reject the case when the m_n and m_t are equal. z_n and z_t refer to the camera space depth of the pixel's next and current position respectively. For a target pixel to be in the inner blur region of a foreground object, it would have to be shallower than its next position. Hence, z_n has to be deeper than z_t by a value greater than SOFT_Z_EXTENT, a scene-dependent variable with a positive value as introduced in the McGuire et al. (2012) paper for calculating sample contribution.

For pixels that are not filtered out, they are passed through a 5×5 Sobel convolution kernel which estimates how extreme an edge is. The kernel is applied to the G-Buffer to obtain an approximate derivative of the gradient associated with each pixel, based on the depth and surface normal information of its vicinity readily accessible from the deferred shading stage. The depth derivatives identify the divide between overlapping objects where the colour of one object might be revealed through the other, while the normal derivatives can locate pronounced differences in the orientation of primitive faces within objects themselves near their silhouettes. The output of each pixel from this filter is hence computed as follows:

$$x = \delta_d + \delta_n \tag{3}$$

$$x_n = \texttt{saturate}\,(1 - \frac{1}{x+1}) \tag{4}$$

Here, δ_d and δ_n refer to the depth derivative and length of normal derivatives respectively from the Sobel operator. The normalized x, x_n, is subsequently evaluated against an edge threshold e, which is set to be high in order to effectively eliminate non-edges from the mask. Only pixels that pass the threshold test will be marked in the edge mask.

Lastly, a range check pass is applied. Each pixel with a nonzero speed then samples along the direction of its velocity. If any of its samples is marked in the edge mask, the pixel passes into the final ray mask.

3.2.2 Recursive Ray Tracing

To obtain more accurate information behind foreground objects for compositing inner blur, our ray tracing approach, as illustrated in Figure 4, adapts the idea of recursive ray tracing (Whitted, 1979) where we shoot rays and iteratively advance them deeper into the scene until a different object is found. At the end of the recursion, the occluded background is then revealed with the final hit point. Revealing more layers deeper into the scene with this method increases the accuracy of our motion-blurred background but at diminishing returns. Hence, we limit the ray reveal process to only one background layer and post-process (i.e. approximate using neighbour information) it instead.

For every pixel marked in the ray mask, we shoot a ray from the camera into its world space position and immediately spawn a second ray along the same direction after the first hit. Currently, we use a simple indicator, the difference in luminance, to identify different objects. Hence, we terminate the recursion when the latest hit point reads a luminance different from the first hit or the maximum number of recursions is reached.

4 IMPLEMENTATION

To implement our hybrid MBlur approach, we made use of the NVIDIA Falcor real-time rendering framework (Benty et al., 2020) on DirectX 12 for ray tracing acceleration. The scenes used for testing our approach are THE MODERN LIVING ROOM (commonly

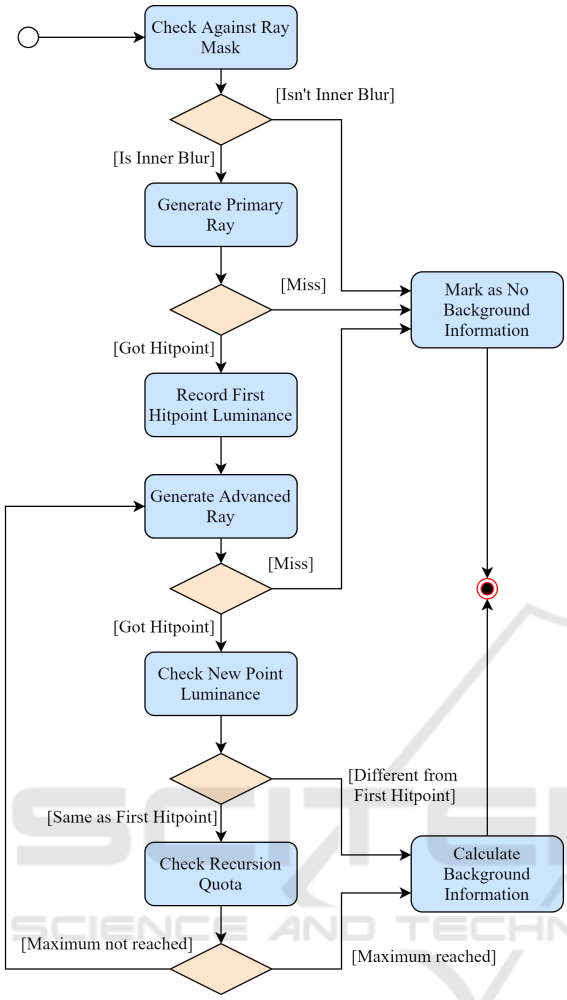


Figure 4: Ray tracing approach.

referred to as PINK ROOM) (Wig42, 2014), UE4 SUN TEMPLE (Epic Games, 2017) and the interior scene of AMAZON LUMBERYARD BISTRO (BISTRO INTERIOR) (Amazon Lumberyard, 2017).

For our post-process tile-dilate pass, we followed McGuire et al. (2012)'s approach of taking a neighbourhood size of $n = 3$ but found that a tile size of $m = 40$ generally worked better than $m = 20$ for us in our test scenes with fast-moving foreground objects. This is because the tile length scales the amount of blur we can observe in the final image, and we require a substantial amount of blur to expose and inspect the quality of semi-transparencies of foreground silhouettes through which background geometry is revealed. Hence, we also tested our scenes at long exposure times or low shutter speeds of $1/60$ s, against high frame rates of 100 to 300 fps so as to scale our per-exposure velocities effectively. As for our sample count, we took 15 samples along the magnitude of the dominant neighbourhood velocity as recommended in McGuire et al. (2012). Our SOFT_Z_EXTENT was 3 cm (PINK ROOM and SUN TEMPLE) and 5 mm (BISTRO INTERIOR), which are also within the suggested range of values from McGuire et al. (2012).

As for our ray mask, we employed a strict edge threshold e of 0.9 (PINK ROOM), 0.98 (SUN TEMPLE) and 0.96 (BISTRO INTERIOR) which were effective in detecting the intricate edges of scene objects while rejecting non-edges well. For the ray tracing pass, we limited the recursion level to 5. However, for complex scenes with concave objects that lead to a significant amount of self-occlusion, this level can be increased.

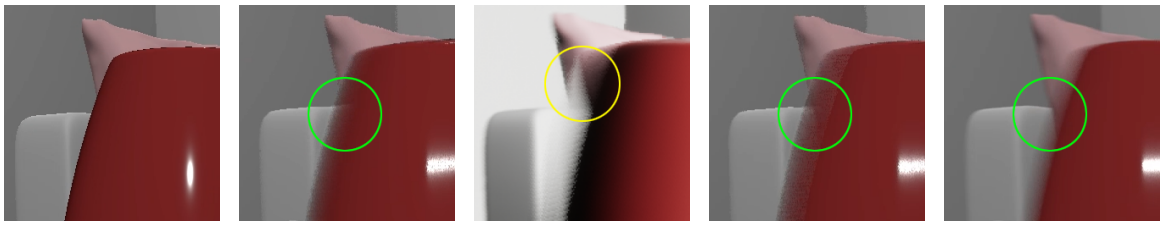
When the target pixel is deeper than its sample by at least SOFT_Z_EXTENT in the post-process pass, we artificially magnify w_f by 30 for a smooth colour transition from the object's edge to its outer blur. On the other hand, we deem the pixel to be in the inner blur region based on whether it is marked in the ray mask. If the pixel is in the mask, we also optionally magnify its overall background colour weight (w in background colour) and diminish its foreground colour weight (w in foreground colour) accordingly by the same magnitude of 3, so background geometry can be observed more clearly through the silhouette of foreground objects with partial occlusion.

5 RESULTS

Our hybrid MBlur approach (d) aims to improve the visual quality of the post-process technique in McGuire et al. (2012) with reasonable additional performance cost. Besides the original rasterized colour (a) and an adapted version of the post-process technique at the new sampling range as shown in Figure 3 (b), we also evaluate our approach against state-of-the-art post-process MBlur from Unreal Engine 4 (UE4) (c), which adopts a similar technique to McGuire et al. (2012). UE4 also produces MBlur by comparing the velocity of each pixel to that of its neighbours (Epic Games, 2020). The ground truth is given by the distributed ray tracing approach from Cook et al. (1984) (e). Our demo video (link) provides more detailed comparisons of the results.

5.1 Graphics Quality Comparison

Post-process MBlur inaccurately reconstructs the background by reusing neighbouring information available in the initial sharp rasterized image, which creates artifacts such as mismatched patterns when the background contributing to the inner blur is not of a uniform colour. In Figure 5, we choose a shot



(a) (b) (c) (d) (e)

Figure 5: Comparison of partial occlusion quality.

from PINK ROOM with a moving red vase in front of a white sofa with a pink cushion on it. For the adapted post-process MBlur from McGuire et al. (2012), the silhouette of the cushion incorrectly terminates at the edge of the vase. While the pink colour from the cushion should extend towards the bottom right direction into the geometry of the vase, it is blocked by a horizontal division created by motion vector-based sampling. This artifact is more noticeable for UE4, where the background colour within the inner blur is practically a linear translation from the left of the edge, as shown in the yellow circle. However, our hybrid method samples what is behind the inner blur of moving objects, hence we can obtain the desired background, which in this case is the extension of the cushion silhouette in the green circle corresponding to the ground truth.

However, it can be seen that hybrid MBlur does not achieve as much semi-transparency as compared to the ground truth, where the background colour is more visible within the silhouette of the vase. Nonetheless, the rendering time of the ground truth is 13 fps at 200 rays per pixel (rpp), which is a lot slower than 205 fps for hybrid MBlur. In practice, it is not common to use distributed ray tracing to render MBlur in real-time as it is difficult to meet interactive frame rates, while reducing the rpp to achieve ideal performance will introduce a significant amount of sampling noise.

Against the ground truth, our hybrid MBlur method performs well in objective visual metrics PSNR and SSIM when compared to other state-of-the-art post-process methods, as presented in Table 1. It is also interesting to note that the improved MBlur method implemented in the latest version of UE4 does not fare as well as the basic adapted post-process method from McGuire et al. (2012).

Table 1: Comparison of similarity to ground truth.

	PSNR	**SSIM**
Adapted Post-Process	24.20	0.91
UE4 Post-Process	12.32	0.68
Hybrid MBlur	**25.69**	**0.92**

5.2 Performance

Although the Falcor framework helped facilitate the implementation of our technique by taking care of scene loading, pipeline set-up and the creation of ray tracing acceleration structures, the abstraction of many low-level details from Direct3D makes it difficult to optimize rendering. However, even without substantial optimization, we have managed to achieve relatively interactive frame rates and pass durations, as shown in Table 2 based on shots of varying geometric complexity in Figure 6. Our measurements are taken with the Falcor profiling tool on an Intel Core i7-8700K CPU at 16GB RAM with an NVIDIA GeForce RTX 2080 Ti GPU.

Table 2: Pass durations (in ms) and frame rates.

Shot	Processor	PR	ST	BI
G-Buffer	CPU	0.26	0.54	1.74
	GPU	1.19	3.54	6.43
Ray Mask	CPU	0.05	0.05	0.06
	GPU	0.33	0.38	0.33
Ray Trace	**CPU**	**1.13**	**2.27**	**2.99**
	GPU	0.59	0.65	0.62
Tile-Dilate	CPU	0.05	0.06	0.08
	GPU	1.48	1.44	1.43
PP-Composite	CPU	0.03	0.04	0.04
	GPU	0.85	0.82	0.82
Others	CPU	3.14	4.67	3.48
	GPU	0.39	0.42	0.29
Total Duration	CPU	4.66	7.63	8.39
	GPU	4.83	7.25	9.92
Frame Rate		205	148	112

Hybrid MBlur, which contains an additional ray reveal process, is expectedly slower than the adapted post-process McGuire et al. (2012) approach that has a frame rate of 295 fps for PR. As seen in Table 2, the creation of the ray mask and the ray trace pass are the major components of the ray reveal algorithm. The ray trace pass increases in duration on the CPU with the geometric complexity of each shot. As such, it is expected that the G-Buffer pass aligns with this

(a) PINK ROOM (PR) (b) SUN TEMPLE (ST) (c) BISTRO INTERIOR (BI)

Figure 6: Shots used for profiling.

trend as it is where deferred shading is carried out. However, the durations of the other passes appear to be independent of this geometric complexity.

5.3 Limitations and Future Work

Since our work is focused on the main idea of real-time hybrid MBlur, some common spatial sampling constraints in MBlur are used for simplicity considerations. These constraints include linear inter-frame motion and stable lighting within the exposure time, as well as moderate screen space velocity for a reasonable tile size. However, it is possible to accommodate nonlinear MBlur at low cost with a curve-sampling scatter approach, as demonstrated in Guertin and Nowrouzezahrai (2015).

Additionally, the ideal indicator for termination in our ray reveal algorithm should be the difference in velocity and not luminance. However, it is inefficient to check the velocity of a hit point during the ray tracing process. Hence, as a future improvement to our approach, we hope to incorporate GeometryIndex from DirectX Raytracing Tier 1.1, which will enable the ray tracing shader to distinguish geometries. This will be more suitable than luminance in scenes with multiple overlapping moving objects of the same luminance value.

Moreover, it is possible to improve the efficiency of our pipeline by restricting the computation of velocities to a user-defined depth for the scene, if geometry movement is localized. We also intend to integrate hybrid MBlur with other real-time hybrid rendering effects, such as depth of field (Tan et al., 2020a) where we mitigate similar problems of partial occlusion from post-processing approaches.

6 CONCLUSION

We present a real-time hybrid motion blur rendering technique that produces more accurate partial occlusion semi-transparencies on the silhouettes of moving foreground geometry as compared to state-of-the-art post-processing techniques. By leveraging hardware-accelerated ray tracing within a ray mask, we have achieved relatively interactive frame rates for real-time rendering in games without extensive optimization. We aim to integrate and optimize multiple hybrid rendering techniques including depth of field into our hybrid rendering engine, which will be open-sourced for the benefit of the research community and industry.

ACKNOWLEDGEMENTS

We thank Wyman (2018) for the Falcor scene file of THE MODERN LIVING ROOM (CC BY) as well as the NVIDIA ORCA for that of UE4 SUN TEMPLE (CC BY-NC-SA) and AMAZON LUMBERYARD BISTRO (CC BY). This work is supported by the Singapore Ministry of Education Academic Research grant T1 251RES1812, “Dynamic Hybrid Real-time Rendering with Hardware Accelerated Ray-tracing and Rasterization for Interactive Applications”.

REFERENCES

Akenine-Möller, T., Munkberg, J., and Hasselgren, J. (2007). Stochastic rasterization using time-continuous triangles. In Segal, M. and Aila, T., editors, *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH ’07, pages 7–16, Goslar, DEU. The Eurographics Association.

Amazon Lumberyard (2017). Amazon lumberyard bistro, open research content archive (orca).

Andrade, P., Sabino, T. L., and Clua, E. (2014). Towards a heuristic based real time hybrid rendering - a strategy to improve real time rendering quality using heuristics and ray tracing. In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications - Volume 3: VISAPP, (VISIGRAPP 2014)*, volume 3, pages 12–21. The Institute for Systems and Technologies of Information, Control and Communication (INSTICC), SciTePress.

Beck, S., Bernstein, A. C., Danch, D., and Fröhlich, B. (1981). Cpu-gpu hybrid real time ray tracing frame-

work. volume 0, pages 1–8. The Eurographics Association and Blackwell Publishing Ltd.

Benty, N., Yao, K.-H., Clarberg, P., Chen, L., Kallweit, S., Foley, T., Oakes, M., Lavelle, C., and Wyman, C. (2020). The Falcor rendering framework.

Cabeleira, J. P. G. (2010). Combining rasterization and ray tracing techniques to approximate global illumination in real-time. Master's thesis, Portugal.

Catmull, E. (1984). An analytic visible surface algorithm for independent pixel processing. *SIGGRAPH Comput. Graph.*, 18(3):109–115.

Chalmers, A., Debattista, K., and dos Santos, L. P. (2006). Selective rendering: Computing only what you see. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 9–18, New York, NY, USA. ACM.

Chen, C.-C. and Liu, D. S.-M. (2007). Use of hardware z-buffered rasterization to accelerate ray tracing. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 1046–1050, New York, NY, USA. ACM.

Choi, J. and Oh, K. (2017). Real-time motion blur based on per pixel fragment list. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '17, pages 132–135, New York, NY, USA. Association for Computing Machinery.

Cook, R. L. (1986). Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72.

Cook, R. L., Carpenter, L., and Catmull, E. (1987). The reyes image rendering architecture. *SIGGRAPH Comput. Graph.*, 21(4):95–102.

Cook, R. L., Porter, T., and Carpenter, L. (1984). Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18(3):137–145.

Dayal, A., Woolley, C., Watson, B., and Luebke, D. (2005). Adaptive frameless rendering. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA. ACM.

Epic Games (2017). Unreal engine sun temple, open research content archive (orca).

Epic Games (2020). *Unreal Engine 4 Documentation: 1.12 - Motion Blur*.

Fatahalian, K., Luong, E., Boulos, S., Akeley, K., Mark, W. R., and Hanrahan, P. (2009). Data-parallel rasterization of micropolygons with defocus and motion blur. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 59–68, New York, NY, USA. Association for Computing Machinery.

Grant, C. W. (1985). Integrated analytic spatial and temporal anti-aliasing for polyhedra in 4-space. *SIGGRAPH Comput. Graph.*, 19(3):79–84.

Gribel, C. J., Barringer, R., and Akenine-Möller, T. (2011). High-quality spatio-temporal rendering using semi-analytical visibility. *ACM Trans. Graph.*, 30(4).

Gribel, C. J., Doggett, M., and Akenine-Möller, T. (2010). Analytical motion blur rasterization with compression. In Doggett, M., Laine, S., and Hunt, W., editors, *Proceedings of the Conference on High Performance Graphics*, HPG '10, pages 163–172, Goslar, DEU. The Eurographics Association.

Gribel, C. J., Munkberg, J., Hasselgren, J., and Akenine-Möller, T. (2013). Theory and analysis of higher-order motion blur rasterization. In *Proceedings of the 5th High-Performance Graphics Conference*, HPG '13, pages 7–15, New York, NY, USA. Association for Computing Machinery.

Guertin, J.-P. and Nowrouzezahrai, D. (2015). High Performance Non-linear Motion Blur. In Lehtinen, J. and Nowrouzezahrai, D., editors, *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. The Eurographics Association.

Haeberli, P. and Akeley, K. (1990). The accumulation buffer: Hardware support for high-quality rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 309–318, New York, NY, USA. Association for Computing Machinery.

Halton, J. H. (1970). A retrospective and prospective survey of the monte carlo method. *SIAM Review*, 12(1):1–63.

Hertel, S., Hormann, K., and Westermann, R. (2009). A hybrid gpu rendering pipeline for alias-free hard shadows. In Ebert, D. and Krüger, J., editors, *Eurographics 2009 Areas Papers*, pages 59–66, München, Germany. The Eurographics Association.

Hou, Q., Qin, H., Li, W., Guo, B., and Zhou, K. (2010). Micropolygon ray tracing with defocus and motion blur. *ACM Trans. Graph.*, 29(4).

Jimenez, J. (2014). Advances in real-time rendering in games, part i: Next generation post processing in call of duty: Advanced warfare.

Korein, J. and Badler, N. (1983). Temporal anti-aliasing in computer generated animation. *SIGGRAPH Comput. Graph.*, 17(3):377–388.

Lauterbach, C. and Manocha, D. (2009). Fast hard and soft shadow generation on complex models using selective ray tracing. Technical report, University of North Carolina at Chapel Hill.

Leimkühler, T., Seidel, H.-P., and Ritschel, T. (2018). Laplacian kernel splatting for efficient depth-of-field and motion blur synthesis or reconstruction. *ACM Trans. Graph.*, 37(4).

Macedo, D. V. D., Serpa, Y. R., and Rodrigues, M. A. F. (2018). Fast and realistic reflections using screen space and gpu ray tracing—a case study on rigid and deformable body simulations. *Comput. Entertain.*, 16(4).

Marrs, A., Spjut, J., Gruen, H., Sathe, R., and McGuire, M. (2018). Adaptive temporal antialiasing. In *Proceedings of the Conference on High-Performance Graphics*, HPG '18, pages 1:1–1:4, New York, NY, USA. ACM.

McGuire, M., Enderton, E., Shirley, P., and Luebke, D. (2010). Real-time stochastic rasterization on conventional gpu architectures. In *Proceedings of the Conference on High Performance Graphics*, HPG '10, pages 173–182, Goslar, DEU. The Eurographics Association.

McGuire, M., Hennessy, P., Bukowski, M., and Osman, B. (2012). A reconstruction filter for plausible motion blur. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 135–142, New York, NY, USA. ACM.

NVIDIA Corporation (2021). Nvidia ampere ga102 gpu architecture. White paper.

Potmesil, M. and Chakravarty, I. (1983). Modeling motion blur in computer-generated images. *SIGGRAPH Comput. Graph.*, 17(3):389–399.

Ritchie, M., Modern, G., and Mitchell, K. (2010). Split second motion blur. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, pages 17:1–17:1, New York, NY, USA. ACM.

Rosado, G. (2007). Chapter 27. motion blur as a post-processing effect. In Nguyen, H., editor, *GPU Gems 3*, chapter 27. Addison-Wesley Professional, 1st edition.

Sattlecker, M. and Steinberger, M. (2015). Reyes rendering on the gpu. In *Proceedings of the 31st Spring Conference on Computer Graphics*, SCCG '15, pages 31–38, New York, NY, USA. Association for Computing Machinery.

Sousa, T. A. (2013). Advances in real-time rendering in games, part ii: Graphics gems from cryengine 3.

Sung, K. H., Pearce, A., and Wang, C. (2002). Spatial-temporal antialiasing. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):144–153.

Tan, Y. W., Chua, N., Biette, N., and Bhojan, A. (2020a). Hybrid dof: Ray-traced and post-processed hybrid depth of field effect for real-time rendering. In *ACM SIGGRAPH 2020 Posters*, SIGGRAPH '20, New York, NY, USA. Association for Computing Machinery.

Tan, Y. W., Cui, X., and Bhojan, A. (2020b). Hybrid mblur: Using ray tracing to solve the partial occlusion artifacts in real-time rendering of motion blur effect. In *ACM SIGGRAPH 2020 Posters*, SIGGRAPH '20, New York, NY, USA. Association for Computing Machinery.

Whitted, T. (1979). An improved illumination model for shaded display. *SIGGRAPH Comput. Graph.*, 13(2):14–.

Wig42 (2014). The modern living room.

Woop, S., Áfra, A. T., and Benthin, C. (2017). Stbvh: A spatial-temporal bvh for efficient multi-segment motion blur. In *Proceedings of High Performance Graphics*, HPG '17, New York, NY, USA. Association for Computing Machinery.

Wyman, C. (2018). Introduction to directx raytracing. In *ACM SIGGRAPH 2018 Courses*, SIGGRAPH '18.

Yang, L., Zhdan, D., Kilgariff, E., Lum, E. B., Zhang, Y., Johnson, M., and Rydgård, H. (2019). Visually lossless content and motion adaptive shading in games. *Proc. ACM Comput. Graph. Interact. Tech.*, 2(1).

APPENDIX

Post-process Sampling Range

McGuire et al. (2012) uses the dominant velocity within a neighbourhood of pixels as a heuristic. It first produces a per-exposure velocity from an inter-frame motion vector, then applies a tile-dilate pass to select the dominant velocity in the neighbourhood with the largest magnitude. Here, the per-pixel velocities are clamped at the length of each tile for the tile-dilate pass to capture the maximum velocity from all possible contributions, preventing tiling artifacts.

For the tile-dilate pass, tile-dominant velocities are first selected for tiles of length m in pixel units, where m is a common factor of the width and height of the image buffer resolution. Then, neighbour-dominant velocities are selected for kernels of length n in tile units. For example, if we take m = 40, the size of each tile is hence $40 \times 40 = 1600$ pixels. If we then dilate each tile with n = 3 based on its 8 direct neighbours, our dilated tile will take the maximum pixel velocity from 9 tiles $\times$ 1600 pixels/tile = 14400 pixels in total.

Compared to approaches that use per-pixel velocities directly for gathering like Ritchie et al. (2010), this dominant neighbour velocity approach prevents issues such as sharp silhouettes as well as moving objects shrunk in size, as illustrated in McGuire et al. (2012). Following its idea, the direction of sampling at each pixel is along the dominant velocity of its neighbourhood, and the range of samples is within one magnitude of this velocity.

Post-process Sample Contribution

McGuire et al. (2012) performs a weighted average w of all corresponding samples for each target pixel.

$$w = w_f + w_b + w_s \tag{5}$$

$$w_f = \texttt{saturate}\,(1+z) \times \texttt{saturate}\,(1 - \frac{\Delta s}{v_s}) \tag{6}$$

$$w_b = \texttt{saturate}\,(1-z) \times \texttt{saturate}\,(1 - \frac{\Delta s}{v_t}) \tag{7}$$

$$w_s = \texttt{cylinder}\,(v_s, \Delta s) \times \texttt{cylinder}\,(v_t, \Delta s) \times 2 \tag{8}$$

where

$$z = \frac{z_t - z_s}{\text{SOFT\_Z\_EXTENT}} \tag{9}$$

$$\texttt{cylinder}\,(v, s) = 1 - \texttt{smoothstep}\,(0.95v, 1.05v, s) \tag{10}$$

In the above formulae, z represents the extent that the target pixel is deeper into the scene than its sample (negative when the pixel is shallower), while Δs

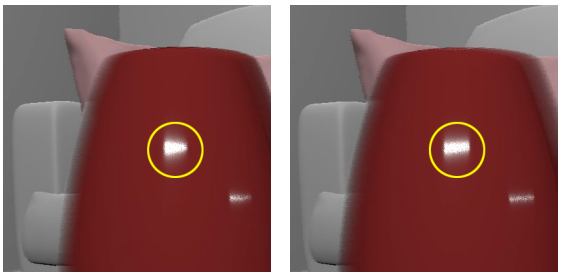
refers to the distance between the screen space positions of the target pixel and its sample. v_t and v_s are the screen space speeds of the target pixel and its sample respectively, while z_t and z_s are their camera space depths. SOFT_Z_EXTENT is a positive-valued scene-dependent variable as introduced in McGuire et al. (2012) used for the classification of samples into the foreground or background of the target pixel.

Case 1: Blurry Sample in Front of Any Target

w_f considers the first case where outer blur is present: a shallower sample of speed larger or equal to its offset from the target pixel in screen space can contribute to the pixel's colour as part of the foreground. The sample contributes as foreground by covering the target pixel within the exposure time. On the other hand, since a sample deeper than the target pixel stands a higher chance to be blocked during the exposure time, the first factor in Equation 6 smoothly filters out contribution from deeper samples. Given that faster pixels also contribute less for each unit (in our case it is the length of one pixel) in their trail, a comparison between the sample's speed and its distance to the target pixel is performed as the second factor of Equation 6.

Case 2: Any Sample behind Blurry Target

As for inner blur, the target pixel is moving and deeper samples within its velocity range are selected to fill the transparency left by it. Hence, for w_b, the depth comparison is done reversely to filter out shallower samples. Since Jimenez (2014) illustrates that the edge of a motion-blurred object should be a smooth gradient, a comparison between the target pixel's speed and its distance to the sample is also performed as part of the second factor of Equation 7 to assign a lower weight to further samples.



(a) w_s excluded (b) w_s included

Figure 7: Specular highlight in McGuire et al. (2012) post-processed MBlur (one-directional).

Case 3: Simultaneously Blurry Target and Sample

A sample at any depth can contribute to the target pixel's colour as part of the foreground or background when both the sample and the pixel are located in each other's velocity range. w_s blurs the target pixel and its sample into each other as a supplement to the cases not handled well by the first two terms in Equation 5. Since further samples have been assigned lower weights in w_f and w_b, the area of the resulting MBlur for regions of high variance appears to shrink as the target pixel gets further from these regions. For instance, in the case of specular highlights, as shown in Figure 7, the resulting blur appears to be triangular-shaped with the base at the exact specular highlight in the sharp rasterized image, while the desirable case should be rectangular-shaped. Since w_s will produce a positive value when the distance between the target pixel and its sample is not significantly higher than the speed of either of them, it creates a more reasonable blur effect by enlarging the shrinking region.