Raytracing Renaissance: An elegant framework for modeling light at Multiple Scale

Sudhanshu Kumar Semwal
Department of Computer Science
University of Colorado
Colorado Springs, CO
USA 80906
ssemwal@uccs.edu

ABSTRACT

Ray tracing remains of interest to Computer Graphics community with its elegant framing of how light interacts with virtual 3D objects, being able to easily support multiple light sources during rendering and using sampling of estimates of intensity values at multiple surfaces in a recursive manner using light as ray. Ray tracing can also provide a simple framework of merging synthetic and real cameras. Recent trends to provide implementations at the chip-level means raytracing's constant quest of realism would propel its usage in real-time applications. AR/VR, Animations, 3DGames Industry, 3D-large scale simulations, and future social computing platforms are just a few examples of possible major impact. Raytracing is also appealing to HCI community because raytracing extends well along the 3D-space and time, seamlessly blending both synthetic and real cameras at multiple scales to support storytelling. This presentation will include a few milestones from my work such as the Slicing Extent technique and Directed Safe Zones. Our recent applications of applying Scan&Track with machine learning techniques creating novel synthetic views, which could also provide a future doorway to handle dynamic scenes with more compute power as needed, will also be presented. It is once again renaissance for ray tracing which for last 50+ years has remained the most elegant technique for modeling light phenomena in virtual worlds at whatever scale compute power could support.

Keywords

Ray tracing, Slicing Extent Technique, Directed Safe Zones, Active Space Indexing Method, AR.

1. INTRODUCTION

In Augmented Reality applications as the synthetic camera images are merged with the reality all around us, the merging is usually not smooth due to lighting conditions and mismatch of conditions as two disparate events are joined together with spatial mismatch. The main thinking of this paper is that raytracing with Active Space Index Method could propel a renaissance of using raytracing techniques towards an effective solution of merging of such disparate events by merging real and synthetic scenes into one physical space captured in front of a set of cameras. We call this 3D-space an active-space as it allows projection of a point onto a set of cameras to be seen. Also, an active-space indexing method is developed so that given the projection of the same 3D point in active-space in the set of cameras can be used to estimate the 3D point's coordinates. This paper presents basic ideas in this paper so that real and imaginary objects could be part of raytracing techniques. Summary of our previous work [Dau90, Kva97, Sem92, Sem93, Sem98a, Sem98b, Sem01] is first presented. Using some of these ideas we make a case towards using raytracing as a unifying concept towards merging synthetically generated scenes with camera-based sequences in the hopes of creating a process towards resolving subtle light mismatch seen in recent work in AR applications [Li20, Har23] and movies where synthetic and natural objects are merges to create a rendered image.

2. Spatial subdivision algorithms for Raytracing

A ray starting at some point C and passing through a point on the image-plane (IP) intersects with object A and generates two new rays. R1 and T1. These two rays recursively traverse the scene. For example, ray R1 is shown to intersect with object B generating, in turn, R2 and T2. Bot the intersection points on objects A and B are in line-of-sight of light source as shown in the Figure 1, which contributes light intensity as L1 and L2 at points A and B respectively. When the rays start from point C the process is called *backward* ray tracing as opposed to when the rays start at the light source and then are tracked in forward raytracing. Since the idea is to generate the

scene for the image-plane, backward raytracing is considered much efficient as only those rays are tracked which originate from C and pass through every pixel on the image plane and are needed to create the scene-render. For example, if we are generating a 60 by 40 image, there are 2400 initial rays which are tracked through the scene starting at point C through 2400 pixels on the image plane. The intensity of each such ray starting at point C and passing through some pixel-point IP is estimated using a tree which keeps track of the secondary rays as shown in Figure 2.

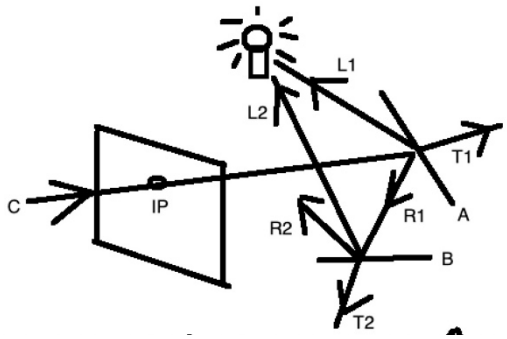

Figure1: Basic ray tracing starting from C.

To estimate the intensity at point IP on the image plane using the concept of recursive raytracing, we follow the path of the rays which are generated as reflective rays, R1 and R2, and transmitted rays, T1, and T2 (Figure 2) are followed generating their own intersection points with other objects in the scene in turn providing sample intensities which can then be combined as these intensities are summed upward through the tree from leave nodes. Effect of lights for intensity values being returned from all visible intersection points can also be added as shown in Figure 3. Ambient intensities approximate the intensity returned by a ray when a ray travels out of the scene. Usually this happens when a ray travel outward away from all objects as it intersects the bounding box containing the scene [Woo90]. The direct line of sight from light sources to the intersection points means that the light source effects can also be incorporated into the intensities, as shown in Figure 3. Aggregate of all these effects can be summed incorporating the distance of light source, reflectivity and transmissivity of the objects mathematically. All these effects can be combine to return the estimated intensity for the pixel IP as shown in Figures 1 and 3. Subpixel samples can be incorporated when multiple stochastic rays generate effects based on bi-directional reflectance distribution function (BRDF) based on material properties of the objects in the scene. This leads to the idea of path tracing which have been used to create stunning realist images in many movies and animation sequences. Path tracing has also been implemented in several industry leading special effects and movies recently as well.

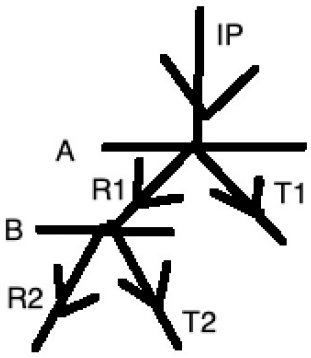

Figure 2: Estimating Intensity IP for a primary ray at point IP using secondary rays T1, R1, T2, R2 etc.

Path tracing has become the industry standard creating photo-realistic images by judiciously spawning several rays stochastically and applying BRDF functions and its variations judiciously. Our methods and new directions proposed can also be extended to those industry leading implementations. This includes path tracing renderers in Maya [Geo18], Sony's Arnold [Kul18], Weta's Manuka [Fas18], Disney's Hyperion [Bur18] and Pixar's Ruderman [Chr18]. Cloud implementations of raytracing are also working towards a goal of real time ray tracing [Xie2021] with 8 frames per second being reported, and NVidea is reporting GPU enhanced ray tracers for some years now. In recent work [Har21], the graphics pipeline is improved as deep learning techniques generate frames in between two graphics pipeline rendered frames. If the scenes are not changing, then generated rendered frames can be used as examples of images based on camera angle when scene is invariant. In game playing, as large number of frames are rendered for invariant scenes and light sources, they can be used for training and synthetic frame rendering and can sometime be used to generate acceptable frames as explained in [Har21]. The trained network is used to create an in between to offload the rendering pipeline and works for the case reported in [Har21]. The idea is to offload graphics pipeline and use deep learning, and supposedly faster frames in-between the rendered frames [Har21]. Ofcourse, when the scene or light source change then offloading can be suspended in favor of graphics rendering again. When the scene stabilizes then we can revert back the load to using deep learning learned images. As we will discuss later, most of the spatial data structures do not handle change in scenes due to explosions or when objects intersects the data structure updates are necessary (i.e. preprocessing) needs to occur adding delays in rendering. Once idea which we propose

later in this paper is to let the user know that the scene is under construction, especially when massively multiplayer games interactions are so critical to happen in real-time.

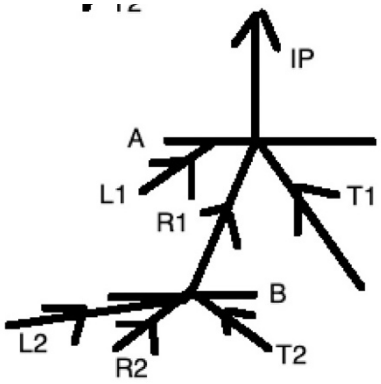

Figure 3: Merging of samples of intensities using a raytracing tree which us generated and thein intensities combined.

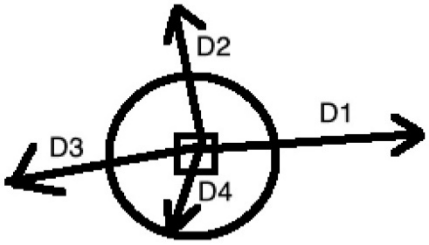

Figure 4: Proximity Cloud (PC) vs Directed Safe Zones (DSZ).

3. SET and DSZ ray tracing techniques.

The Slicing Extent Technique (SET) [Sem87, Sem92] uses projection of the objects in the scene of 2-D planes surrounding the object. In 1990, is was implemented using a 3D-grid interpretation, and was called the modified slicing extent technique (MSET) [Sem93]. The benefit was to allow the fast grid-traversal to be incorporated while ray traverses through the scene mimicking the SEADS implementation [Fuj86]. In addition, we also incorporated an octree [Gla84] to isolated isolate dense and sparse area in the scene so that a finer grid could be used for dense areas, and sparse areas can be skipped quickly in comparison to octree [Gla84], thus allowing multiple hierarchies to be managed using MSET. Later, work on the proximity cloud [Coh94a, 94b] further improved the grid-implementations for ray tracing. A method was developed during preprocessing so that a safe-distance value (D) was determined for every grid-voxel so that a ray passing through a voxel with distance D can skip distance D without missing an intersection as during pre-processing it was determined that there we no objects withing a D distance from this voxel, hence the name safe-distance. This D distance allowed isolated areas to be bypassed in much more efficient way in cormarison to distance D=1 which will mimic the grid traversal itself. It is much faster to skip areas of the scene with no objects in it. During preprocessing, the value of D was determined using transformation [Bor86] to isolate areas of no-object efficiently by using a 3 by 3 by 3 filter on 3D-grid voxels. This is illustrated in Figure 4 where a 2D-grid voxel, or cell, is shown and nearest object for any ray through the edge of the 2D-voxel (cell) are D1, D2, D3, and D4 away. So minimum radius is D4 which is the safe distance a ray could travel in any direction from this 2D-cell without finding an object to intersect. Every cell thus could have its own 2D-circle, or extending this idea to 3D, its own 3D-sphere. Each voxel of the 3D-grid could have such sphere. One can now imagine every voxel to have different (yet close) values creating many spheres with no objects in them. One could imagine spheres of different radii, hence the term, proximity clouds (PC) used in [Coh94b] to describe a Proximity Cloud. Proximity Cloud was a major improvement very as it was more efficient way where ray tracing image generation times were shown to improve over the previously known grid implementations. Normal grid-traversal, moving from one voxel to next voxel could be suspended in favor of jumping D distance away without missing any intersections. Figure 4 shows this concept in 2D with four safe values in four directions. In PC, we choose the minimum of these 4 values, and conclude the safe distance of D4 for that voxel.

Usually city-block distance transformations are used to implement Proximity Clouds so instead of radius we could imagine city-block distances [Coh94b]. This faster method was further improved by a variation of the slicing extent technique called Directed Safe Zones (DSZ) in [Kva97] where the six safe-distances in six direction are calculated as the ray emerges out of a voxel through one of the six-faces of a 3D-voxel (grid-cell). All such distances are calculated during preprocessing by modifying PC's distance transformations filter to suit the DSZ implementation [Kva97]. After preprocessing in DSZ method each of the six faces of 3D-voxel of would have a distance associated with it moving outward from the cell towards left, right, bottom, top, up and down directions. In 2D, this is shown as D1, D2, D3, D4 values in Figure 4. In DSZ, the ray has the capacity to skip six different distances based on its direction of traversal as the ray passes through any of these six faces. This meant that Directed Safe Zones, which extends the Slicing Extent techniques, is more efficient. As shown in Figure 4, DZS has the

flexibility to choose either of D1, D2, D3 or D4 as safe-distances guaranteeing that image generation time will always be better or same in DSZ in comparison to PC implementations. DSZ uses larger distance based on the traversal direction of the ray showing in theory and is an improvement over PC. Using scenes from [Hai98] called random, lanes, snowflake, and cars, a comparative analysis in [Kva97] showed that DSZ outperformed the SEADS/grid and PC implementations for all four scenes. As expected, because of the potential of more empty areas in random and snowflakes scenes major improvements in rendering times were obtained for random and snowflake scenes using DSZ in comparison to Proximity Clouds and SEADS/Grid implementations. Typical performance speedup of 2 for DSZ were seen when compared with grid implementation. For the PC, speedup was 1.5 with respect to grid implementation. In all cases, as expected, DSZ outperformed the PC method and grid (SEADS) [Fuj86] method.

Additional benefits of DSZ method is that, in addition to the outgoing rays emanating from a voxel, DSZ method can treat incoming rays because each face of the voxel can maintain two distances, as was also explained in [Sem87,Sem93]. A ray passing through the left face of a 3D-voxel to the right, or from right to left, upwards-to-downwards or downwards-to-upwards, and front-to-back and back-to-front can be recognized, allowing two directions per face so that 12 different classifications instead of six are possible as a ray passed through a face of a voxel. This is a useful benefit allowing us to manage 3D scenes better, as we plan to embed synthetic scenes in active spaces as explained in next sections.

4. Active Space Indexing Method Review Setup and Data Capture

Active-Space Indexing Method [Sem01] uses ideas of triangulation, including closest distance between two rays to find 3 D (x,y,z) position of a point P. Given image-imprint Im1, Im2, and Im3 on the camera-images for a point P, Active Space Indexing method preprocesses projections of several 3D grid points on each camera images to determine the 2D-indices for each camera images. If we assume that 2D grid points can be indexed between 1 to n and 1 to m then Im1, Im2, and Im3 must fall on some index (x,y) using the projections of grid-patterns in p such planes. When three such indices on for each Im1, Im2 and Im3 points identified in all three camera images as corresponding to the point P, then these indices define an area, and that area will decrease first and then increase as we process p of these plane from front to back. This allows us to find a voxel which contains point P. Active-space indexing method connects the projection-space to the real 3D space. More details of how Im1, Im2, Im3, called imprint-set can be used to determine position P is further explained in [Sem01] in more detail.

In summary, Active Space Indexing Method is a study of 2D-projections of a set of 3D-points as seen by three cameras. These set of 3D-points are arranged in real 3D-grid in physical space in front of the three cameras. This space in front of the camera is called an active space [Sem98]. The active space indexing method is created by projecting set of n by m planer points inside a rectangle R which contains n by m in an equal distance grid pattern. We used a whiteboard for this purpose. The points on the whiteboard can be shifted some distance away from the previous placement of the whiteboard. In this war, the whiteboard it has an effect of moving same points inside the rectangle R will also move parallel to previous plane positions. We repeat this process several times to obtain images for a set of p whiteboard positions. As the whiteboard moves, it has an effect that 3D grid points inside the active space are projected and preprocesses during preprocessing to create an active space indexing method. During preprocess, the exact pixel locations of all grid points for all p whiteboard positions are determined and stored during preprocessing. This information is sufficient to estimate point P's x,y,z location in n by m by p space using point P's imprint-set as explained in [Sem01]. Assuming the process repeats p times, using same distance. This will have an effect of creating a set of 3D-grid points in the real physical space which we call active-space. For example, n=m=4 and p=8 in Figure 5 so that a 4 by 4 by 8 grid of points are in the active space. Each time image-plane moves we record the projections of set of 4 by 4 image-plane points on 3 cameras which are called Left (L), Center (C) and Right (R) points as shown in Figure 5 and then the as shown in Figure 6. The three cameras are related in a way that all p planes are visible from all the three camera and their projections are therefore highly correlated. We also have shown the projected shape of the rectangles in Figure 5. As explained earlier, rectangle R was created on a whiteboard which was available in the lab [Sem98], and already had a 2D-grid-pattern and the intersection of this points were highlighted with a blue/black pen which were then easily picked up during pre-processing. In this way, we were able to correlated n by m by p points in active space to their projections by preprocessing p of these projections for every camera. All of these points were clearly visible in the three cameras because we had planned the arrangement. Figure 6 shows approximately the front-plane, marked F, and back-plane, marked B, and corresponding projections of rectangular B and F planes and their projected shapes in all three cameras. As the white-board moves from Back to Front p planes create p projections of the grid-pattern on the

on three cameras. Sets of these projections are created and processed during preprocessing and are the basis of active space indexing method. Vertical and horizontal lines and their projected lines on image planes are used to first find the index in 2D planes in all three cameras, as explained earlier, and a triangulation algorithm is used to find the voxel which contained the given 3D-point P using the image print (Im1, Im2, and Im3) of the point P. The image-print, i.e. Im1, Im2, Im3, is identified manually in our implementation. Image-imrpint could be detected automatically as well. Finding image prints is called correspondence problem and there are a variety of techniques including finding significant points first and then correlating these points as image prints using the projections of these points on three cameras. We used significant points extraction and correlation of these points as mentioned in [Sem98a, Sem98b, Sem01].

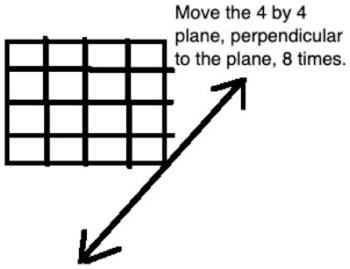

Figure 5: Scan&Track data collection – 4 by 4 planer grid moves 8 times perpendicularly to create a 4 buy 4 by 8 active space, 128 grid points in 3D active-space. A set of voxels in 3D-real space is created.

Image imprint and corresponding 3D points in practice

Basic idea of Scan&Track [Sem98] system was implemented project by a 10 by 10 by 10 set of 3D grid points onto three highly corelated cameras [Sem98a,, Sem98b, Sem01]. Highly corelated camera would mostly maintain the geometric relationship between any two grid points in the same plane. Data collection step uses a planer whiteboard with, say fixed 10 by 10 points clearly marked and visible from each of the camera. Next whiteboard is moved by fixed distance perpendicular to the present location of the whiteboard 10 times to capture projection of 100 points spaced as 10 by 10 by 10 grid of 1000 points all visible in the three cameras. Here the idea was to create active space, a10 by 10 by 10 grid in physical area) with over constrained systems of 1000 points in 3D space. For example, a 3D point P and their associated projections image-imprint (Im1, Im2, and Im3) are known by processing each of the projections. Image imprint Im1, Im2, and Im3 are the pixel locations in the images. Now if the person is inside the active-space and we identify same point, e.g. the tip of the nose in all three camera images as Im1, Im2, and Im3, then active space can be used to find P the location of the tip of the nose of the participant. Scan&Track system can now be used to identify set of image imprints. As explained, Active-space indexing method uses a triangulation process to find the 3D point given I1, I2, and I3 pixel location.

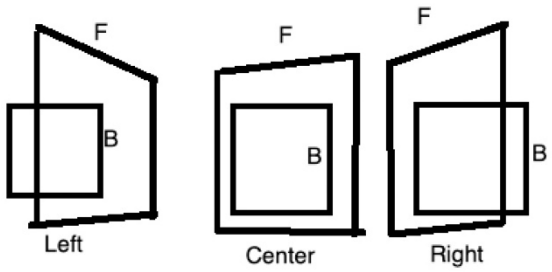

Figure 6: Scan and Track projections on left, center and right. Distortions of the planes are exaggerated to show the effect of projections of front and back planes of active space.

Generating Image imprints using Deep Learning

One of the tasks in the Scan&Track implementation was to generate image imprints (Im1, Im2, Im3). Identification of the imprint was done by a simple filter in our [Sem01] implementation. Today we can use deep learning algorithms to find distinct image imprints specifically for the human participant. The end points for hands and feet could be identified across the three cameras, creating image imprints I1, I2, I3 for hands and feet. This will identify 3D positions in the active-spaces for hands and feet and many such points, as explained in [Sem98b].

5. Future Proposed Work

Active-Spaces for both real and virtual worlds

Both Scan&Track and DSZ methods are based on 3D grid data structures. In DSZ, a voxel's six faces helped us define 12 different directional distance measures which we can safely skip along the direction of the ray. In the Scan&Track system active spaces are the 3D-grids in real-space which we can effectively use to embed both real and synthetically generate worlds.

Our main idea is that virtual worlds consisting of C, IP, A, B and L which can be used to generate intensity values at point IP. Real world consisting of human participant seen by three cameras. Active-space allows us to place A, B, and L relative to human participant while also isolating human participants in the real-work inside an active space. Now virtual objects (such as A, B, and L) can be

placed in the active space in front of the human participant by using active-space Indexing method to find the approximate location of the human participant and then placing A, B and L relative to H. Environment E could also be wrapped around all of these, as shown in Figure 7. E could be another projected image of some other active space, or another active space could be defined in that place behind the human participant as E. The idea is that world of many active spaces can be populated by synthetic scenes and synthetic objects as active spaces provides us one way to define virtual and realobject in the same 3D space. Once these objects are placed, raytracing can be used to combined overall scene. In Figure 7, we have tried to show the mixed-reality scene in active-space merging both synthetic and real objects together so that idea of raytracing can be applied to render images with real participant enclosed inside their active space, and E, represented by active-space of its own in turn. Extending the idea of multiple active spaces we can now expand the 3D space to larger areas as well as define the 3D-space recursively, e.g. one active space can contain several other active spaces. The faces of active space containing human participant H could be samples by multiple cameras in outside-in manner so that approximate intensities on the surface on the active-space can be used during ray tracing as was done in ASET [Dau90].

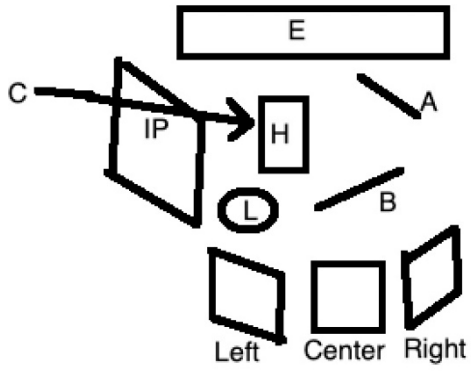

Figure 7: Mixed Reality Setup. Light source L has been placed in front of human participant H. Ray tracing camera is shown with C and IP (See also Figures 1 and 2). A and B are synthetic objects. Note E is environment which could be another Active space Index Mixed Reality Setup or can be green/blue screeded real camera-captured natural scene.

Proposal to merge moving Active-Spaces and multiple scales of grids

Both Scan&Track and DSZ methods are based on 3D grid data structures. In DSZ, a voxel's six faces helped us define 12 different directional distance measures which we can safely skip along the direction of the ray. In the Scan&Track system active spaces are the 3D-grids or real world where we can embed other active-spaces or even synthetically generate virtual world objects. As the objects move the grid-spaces can be managed such that light-effects can be collected in the voxels as an approximation method as we have explored this idea in [Dau90] where primary rays, rays starting from point C in Figure 1 and secondary rays at first level called T1 and R1 or even second level T2 and R2 are checked for intersection with real objects to create a some level or correctness. However, at some level, say level 3 onwards, secondary rays may start to consider that the object occupies the whole voxel which it passes through, thus discarding any intersection checks for level 3 or more secondary rays. Our motivation in [Dau90] was to avoid the actual intersection where such approximation could suffice. Here we are proposing that external facing active spaces faces can be approximated by camera images which capture the human participants actions, and the image's r,g,b values can suffice as intensity value to combine with other effects during ray tracing process. This needs to be further investigation in future.

Scene changes and updates

One of the challenges for ray tracing has been that any movement of objects or light source, even if minor, can severely impact the final image. Spatial data structures need to be completely updated if such events occur. Also, any changes in the shape or the objects such as explosions can severely affect the whole data structure during ray tracing forcing us ton first manage those updates to objects before rendering can occur. We think that such updates in the grid method can be managed by hierarchical embedding of one active-space in another or extending the active-spaces of multiple voxel size. As the objects move, some voxels may be vacated by the object and other will be occupied based on the movement of the object itself. By mapping both synthetic object and real-objects in active-spaces and their grids allows us one way to manage as both the real and imaginary worlds can be combines using number of active-spaces. Active spaces with high activity, handling major explosions, can be isolated and can be label as "under construction" where the walls of that active space would be considered "approximate" while under construction thus giving time to synchronize itself to "available again" while that active space completes whatever it was "under construction" for.

6. Summary

Main idea here is that both virtual worlds and real worlds are in appropriate scale merged using a variety of active-space grids holding both synthetic and real object at an appropriate scale.

Labelling a particular active space under construction is similar to real-life, for example when we see "under construction" sign—we know that experience will usually improve in future. More processor resources could be allocated to fix the restructuring of the active space to manage fragments of data in case of explosion, or movement due to scaling. translation and rotation which object may go through. Rendered frames which are labeled "under-construction" may also appear to notify the user that their experience will improve later.

7. CONCLUSION AND FUTURE RESEARCH

Using distance transformation, we developed a faster method for ray tracing called the directed safe zones (DSZ) where the direction of incoming and outgoing ray from a 3D voxel can be classified based on which face of the voxel the ray emerges out of or goes into. We also used rays to develop a triangulation method to first find the voxel which contains a 3D point P when point P's image imprint (Im1, Im2, Im3) is known. This led to calculating the location of point P. In this paper, we proposed that active spaces could be distributed in real-world of human participants and these worlds can be places with synthetic objects by adding them to the active spaces so that synthetic objects and active-spaces can raytraced, hopefully avoiding the mismatch of scale when synthetic and real-worlds are combined in Mixed Reality 3D applications. In our case, we are proposing that such scenes can be raytraced.

8. ACKNOWLEDGMENTS

My deepest thanks to my colleagues Dr. Jun Ohya, Department Head, and Dr. Ryohei Nakatsu, Director of ATR Media Integration and Communication Lab. hosting my Summer Research visits at ATR, Kyoto, Japan during 1997-99 where Scan&Track systems was developed. My deepest thanks to Dr. Vaclav Skala for providing me this opportunity to present this keynote at one of the finest graphics conference in the world – WSCG 2023! Thank you.

9. REFERENCES

[Bor86] Borgefors G. Distance transformation in digital images, *Computer Vision, Graphics and Image Processing,* 34, pp. 344-371 (1986).

[Bur18] Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. ACM Trans. Graph. 37, 3, Article 33 (July 2018), 22 pages. https://doi.org/10.1145/3182159

[Chr18] Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Ban- nister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. ACM Trans. Graph. 37, 3, Article 30 (Aug. 2018), 21 pages. https://doi.org/10.1145/3182162.

[Cle88] Cleary J.G. and Wyvill G. Analysis of an algorithm for fast ray tracing using uniform space subdivision, *The Visual Computer* 4, pp. 65-83 (1988).

[Coh94a] Cohen D. Voxel Traversal along a 3D Line, *Graphics Gems, IV,* pp. 366-368 (1994)..

[Coh94b] Cohen D. and Sheffer Z. Proximity clouds - an acceleration technique for 3D grid traversal, *The Visual Computer,* 11, pp. 27-38 (1994).

[Dau90] David Dauenhauer and Sudhanshu Kumar Semwal, Approximate Raytracing, Graphics Interface, Halifax, Nova Scotia, Canada, pp. 75-82. Canadian Information Processing Society, International Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques (ACM SIGGRAPH) (ACM SIGGRAPH), and Canadian Man-Computer Communication Society (1990).

[Fas18] Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch- Shading Architecture for Spectral Path Tracing in Movie Production. ACM Trans. Graph. 37, 3, Article 31 (Aug. 2018), 18 pages. https://doi.org/10.1145/3182161.

[Fuj86] Fujimoto A, Tanaka T. and Iwata K. ARTS: Accelerated ray tracing system, *IEEE Computer Graphics And Applications,* 6(4), pp. 16 26 (1986).

[Geo18] Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. ACM Trans. Graph. 37, 3, Article 32 (Aug. 2018), 12 pages. https://doi.org/10.1145/3182160

[Gla84] Glassner A.S. Space subdivision for fast ray tracing, *IEEE Computer Graphics And Applications,* 4(10), pp. 15-22 (1984).

[Gla89] Glassner, A.S. An Introduction to Ray Tracing edited by A.S. Glassner, Academic Press (1989).

[Gla21] Andrew Glassner, Deep Learning: A visual approach, Penguin Random House, 776 pages, 2021.

[Hai87] Haines E.A., A Proposal for Standard Graphics Environments, *IEEE CG&A,* 7(11), pp. 3-5 (Nov 1987).

[Har21] Mark W Harris and Sudhanshu Kumar Semwal, A multi-stage advanced deep learning Graphics Pipeline, SA'21 Technical Communications: SigGraph Asia 2021 technical communications, Article No.: 7, pp. 1-4. https://doi.org/10.1145/3478512.3488609 (2021).

[Har23] Hartholt, Arno; Mozgai, Sharon Creating Virtual Worlds with the Virtual Human Toolkit and the Rapid Integration & Development Environment**,** In: 2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG), pp. 1–6, 2023.

[Kul18] Christopher Kulla, Alejandro Conty, Clifford Stein, and Larry Gritz. 2018. Sony Pictures Imageworks Arnold. ACM Trans. Graph. 37, 3, Article 29 (Aug. 2018), 18 pages. https://doi.org/10.1145/3180495

[Kva97] Kvanstrom H. The Dual extent and Directed Safe Zones techniques for ray tracing, Graphics Interface, 1-72 (1997).

[Li20] Li, Jiaman; Kuang, Zhengfei; Zhao, Yajie; He, Mingming; Bladin, Karl; Li, Hao, Dynamic Facial Asset and Rig Generation from a Single Scan, In: ACM Transactions on Graphics, vol. 39, no. 6, 2020.

[Sem87] Sudhanshu Kumar Semwal, The Slicing Extent Technique for Ray Tracing, Ph.D. dissertation supervised by Dr. Mike Moshell, Department of Computer Science, University of Central Florida, Orlando, Summer 1987, pp. 1-227 (1987). https://stars.library.ucf.edu/rtd/5062/

[Sem92] Sudhanshu Kumar Semwal, Ray Tracing using the Slicing Extent Technique, Institute of Electronics, Information and Communication Engineering (IEICE) Spring Conference, Tokyo, Japan, pp. 7-367 (1992).

[Sem93] Semwal S.K., Kearney C.K., and Moshell J.M. The Slicing Extent Technique for Ray Tracing: Isolating Sparse and Dense Areas, IFIP Trans actions, vol. B-9, pp. 115-122 (1993).

[Sem98a] Semwal SK, Ohya J. The scan&track virtual environment, Virtual Worlds 98, LNAI 1434, pp.63-80, 1998.

[Sem98b] Sudhanshu Kumar Semwal and Jun Ohya, Geometric-Imprints: A Significant Points Extraction Method for the Scan&Track Virtual Environment, Proceedings of the IEEE Third International Conference on Automatic Face and Gesture Recognition (F&G98) Conference, April 14-16, 1998, Nara, Japan, pp. 480-485, IEEE Computer Society.

[Sem01] Sudhanshu Kumar Semwal and Jun Ohya. Spatial Filtering using the Active-Space Indexing Method, in the Graphical Models and Image Processing, Academic Press journal, vol 63, pp 135-150 (2001).

[Wes17] West Geoffrey, Scale: The universal Laws of life and death in organisms, cities and companies, Weidenfeld & Nicolson, Great Briton, pp. 1-455 (2017)

[Whi80] Whitted T., An improved illumination model for shaded display, *CACM,* 23(6), 343-349 (June 1980).

[Woo90] Woo A. Fast Ray-Box Intersection, *Graphics Gems, I,* pp. 395-396 (1990).

[Xie21] Fen Xie, P. Mishchuk, W. Hunt, Real-time cluster path tracing, SA'21 Technical Communications: SigGraph Asia 2021 technical communications, Article No.: 17, pp. 1-4.