information

Article

Visualization of WiFi Signals Using Programmable Transfer Functions

Alexander Rowden [1,*], Eric Krokos [2], Kirsten Whitley [2] and Amitabh Varshney [1]

[1] Graphics and Visual Informatics Laboratory, Department of Computer Science, University of Maryland College Park, College Park, MD 20740, USA; varshney@umd.edu
[2] United States Department of Defense, 1400 Defense Pentagon, Washington, DC 20301-1400, USA; ericpkrokos@gmail.com (E.K.); visual.tycho@gmail.com (K.W.)
* Correspondence: alrowden@umd.edu

Abstract: In this paper, we show how volume rendering with a Programmable Transfer Function can be used for the effective and comprehensible visualization of WiFi signals. A traditional transfer function uses a low-dimensional lookup table to map the volumetric scalar field to color and opacity. In this paper, we present the concept of a Programmable Transfer Function. We then show how generalizing traditional lookup-based transfer functions to Programmable Transfer Functions enables us to leverage view-dependent and real-time attributes of a volumetric field to depict the data variations of WiFi surfaces with low and high-frequency components. Our Programmable Transfer Functions facilitate interactive knowledge discovery and produce meaningful visualizations.

Keywords: ransfer functions; volume rendering; WiFi visualization; data visualization; network rendering

Citation: Rowden, A.; Krokos, E.; Whitley, K.; Varshney, A. Visualization of WiFi Signals Using Programmable Transfer Functions. *Information* **2022**, *13*, 224. https://doi.org/10.3390/info13050224

Academic Editor: Willy Susilo

Received: 19 March 2022
Accepted: 22 April 2022
Published: 26 April 2022

1. Introduction

Visualizing WiFi signal strength can help us engineer superior buildings, academic campuses, and smart cities by ensuring constant, secure, and reliable coverage. Data connection reliability in the information age is an issue of utmost importance to ensure critical data are not lost. It is thus necessary to look into and update how we analyze the coverage and effectiveness of a WiFi signal space. Current design practices analyze aggregate data and heat maps but do not allow an analyst to make decisions about the environment, such as planning router locations that maximize coverage while minimizing cost and potential interference. In this paper, we propose a technique of WiFi visualization which utilizes the direct volume rendering of sampled WiFi data to provide a better situational awareness of the complex WiFi signal space.

Visualizing three-dimensional volumes using direct volume rendering involves the use of transfer functions. A transfer function maps the abstract voxel data to a human-interpretable color and opacity. Commonly, the volumes visualized through these means arise from scientific measurement or simulation and are thus complex, possibly involving a mixture of materials or interacting phenomena. Therefore, it is imperative to design a system to aid in creating simple and effective transfer functions. Good transfer functions reveal significant data regions, mask unimportant areas that may occlude regions of interest, and increase visual clutter. Designing transfer functions automatically, semi-automatically, and interactively is an open area of research. This paper aims to strengthen the visual analysis process by replacing the traditional lookup-based transfer function with a higher-level Programmable Transfer Function that facilitates flexible rendering of multiple scalar fields with their mutual interactions to enable flexible and easy-to-understand visualizations for exploring real-world multifield volumetric data.

Motivated by our driving application of WiFi visualization, we propose a new concept called the *Programmable Transfer Function*. Programmable Transfer Functions can be imple-

mented as shader functions that take multiple inputs, such as camera parameters, lighting parameters, and multiple scalar fields with their gradients, and generate a color and opacity. By replacing the traditional transfer function's single texture lookup, a Programmable Transfer Function enables:

- Superior design through user interaction by leveraging view-dependent attributes to enhance comprehensibility in real time;
- Compact representation as succinct and fast shader code that alleviates the need for large arrays of slow lookups from memory; and
- Operation on multiple scalar fields and therefore can highlight features such as their intersection curves and regions.

Programmable Transfer Functions can assist in visually analyzing large amounts of three-dimensional volumetric data. These functions are easier to plan, analyze and visualize than traditional transfer functions and enable the user to design multifield visualizations without prohibitive memory cost. In addition to our driving application of WiFi visualization, we believe that Programmable Transfer Functions could be helpful for scientists who run large-scale simulations and healthcare professionals studying medical images.

This paper presents how Programmable Transfer Functions are beneficial for our driving application of WiFi visualization. We first present an overview of the Programmable Transfer Function approach in Section 4, which includes the base rendering system's implementation in Section 4.1. We next present three use cases to validate the capabilities of the Programmable Transfer Function in Sections 4.2–4.4. We then present performance analysis in Section 4.5. Finally, we discuss other possible use cases in Section 5 and present our Conclusions in Section 6.

2. Related Works

2.1. WiFi Data Visualization

Current techniques for modeling WiFi signal strength over a large space are limited. An analyst could review the data in its raw form in a comma-separated values (CSV) format file, but the growing volume of data makes this intractable. Therefore, current approaches use statistical modeling and modern rendering techniques to gain an advantage. One such work by Kokkinos et al. [1] utilized upload and download speeds, throughput, and ping times as metrics of internet quality and collected their data via crowdsourcing. Therefore, they evaluated signal coverage over an area rather than at each sample location. Several authors have utilized two-dimensional heat maps to represent their signal data over indoor [2,3] and outdoor [4,5] areas. Another commonly used strategy uses a network graph to represent the wireless network. Although this works quite well for visualizing network security and infrastructure, it fails to provide practical information to answer questions of region coverage or interference [6,7]. This paper presents our system that uses volume rendering of the WiFi signal strength and real-world geometry to help review regional signal coverage, find areas of potential co-channel interference, assess possible security vulnerabilities, and more. We chose to use direct volume rendering rather than other three-dimensional scalar field rendering techniques as it allows for manipulation of the scalar field without substantial overhead, as would be necessary for marching cubes.

2.2. Direct Volume Rendering

Direct Volume Rendering is a method for visualizing three-dimensional scalar fields. These fields often arise in the medical, engineering, and scientific fields due to various data acquisition technologies. In addition, many computer simulations process and output data in n-dimensional grids. Direct Volume Rendering was first introduced by Levoy [8] and improved by Drebin [9]. Volume rendering techniques have improved over the years with the introduction of various acceleration data structures [10] and automated transfer function generation techniques [11].

Direct volume rendering is computationally intensive, and with an increase in the size of datasets, volume rendering performance quickly drops below interactive frame

rates. This field has seen many improvements. These include hardware acceleration techniques [12] that use per-fragment texture fetch operations, texture rendering targets, and per-fragment arithmetic to accelerate the rendering of volumetric data. Another source of frame rate improvements comes in the form of acceleration structures, such as the octree [12] to implement empty-space skipping, which steps over regions that do not contain renderable values. Multiresolution textures or mipmaps have also increased the interactivity of texture-based volume rendering [13] at the cost of memory. Roettger et al. [14] used an innovative technique known as *preintegration* to upsample only semantically significant areas in order to remove aliasing artifacts. The localized preintegration technique of Roettger et al. utilizes the second derivative to modify the step size adaptively and thus better enable step skipping. In order to avoid the cost of transfer function creation, several researchers have designed techniques that use a clustering algorithm to segment the volume into regions of interest and generate a transfer function to show these boundaries [15,16].

2.3. Interaction in Volume Rendering

In the field of volume visualization interaction, Sharma et al. [17] use a graph-based approach to identify material boundaries and create a transfer function. Their graph represents the different materials based on how deep they are in the volume and its density. They then allow user interaction by allowing the transfer function modification for each segment individually. This technique allows the user to change the color and opacity of different segments. After each edit, the transfer functions must be calculated and stored as a texture. Pflesser et al. [18] perform virtual cuts into volumes to simulate surgery to prepare surgeons in training, which acts as a way to view internal structures of a volume. Carpendale et al. [19] increased user interactivity by producing three-dimensional distortion tools for data analysis by modifying the camera to make certain areas of a volume appear larger or smaller. Ip et al. [20] use normalized cuts to create an interactive hierarchical structure for data exploration and transfer function creation. This technique allows users to automatically generate interesting data representations and interact with a fixed set of model variations.

Kniss et al. [21] present an elegant technique that uses multidimensional transfer functions to base the shading of the volume not just on the value at a specific location on the three-dimensional grid but also the gradient, or even further the Hessian, at that location. This paper proposes a set of controls to interact with the multidimensional transfer function to aid in creating these transfer functions. These controls help the user explore the data through different ways of looking at its volume, and they may edit the function by modifying the opacity for specific isovalues.

2.4. Non-Photorealism

Another area of interest for us is non-photorealistic rendering. Our Programmable Transfer Functions modify the rendering to aid in data analysis, but these effects are not a realistic simulation of light transport and are non-photorealistic. Non-photorealistic volume rendering has innovative use cases. For example, an importance-based method proposed by Viola et al. [22] assigns each sample a level of sparseness during an importance compositing step. Despite their intrinsic structure and opacity, they make significant regions more visible during their final render than unimportant regions. Treavett and Chen [23] use a pen-and-ink style to render a three-dimensional or even two-dimensional representation of volume, which they compare to an architect's sketch. They showed that this sketch-like visualization helped analysts in specific tasks. Csebfalvi et al. [24] use non-photorealism to render the contours along a surface, thus providing a more comprehensible view of the overall structure of the volume.

2.5. Multifield Data

Multifield data consist of multiple values at each point. An example of one such multifield dataset would be a standard scalar field paired with the gradient at each point,

or it could be representing another volume entirely. The visualization of multifield data is essential to modern researchers, as most scientific simulations and measurements yield multiple values at each point in three-dimensional space. One way to visualize high-dimensional data is to reproject it into three dimensions using clustering [25]. Another technique is to create a volume with a multidimensional transfer function [26]. While highly versatile, the dimensionality of the transfer function increases memory requirements. For instance, a one-dimensional transfer function over eight-bit characters would require 256 elements, whereas a four-dimensional transfer function would require 256^4 elements. This memory burden is the impetus of several performance improvements. One such technique is to use mixtures of analytical functions to represent the transfer functions, including Gaussians [27] or ellipsoids [28]. Multifield rendering can also visualize the mathematical properties of the inter-field relationships. For example, Multifield Graphs visualize the correlation between multiple scalar fields [29].

3. Data

To evaluate the effectiveness of our Programmable Transfer Functions, we visualize WiFi signal strength data as representing a varying volume over a large scale. The data used in our examples are a collection of WiFi signal data collected on the University of Maryland—College Park campus. Our dataset was collected using two handheld receivers moving across the campus over six one-hour data collection sessions. At each sample point, among the data collected was the router's Service Set Identifiers (SSIDs) and Basic Service Set Identifiers (BSSIDs)—(the names of the WLAN networks and the MAC addresses of the routers, respectively), WiFi signal strength in decibel-milliwatts (dBm), the GPS longitude and latitude of the sample, an estimate of GPS accuracy, the router's security capabilities, and the signal frequency (which contains channel information). We chose to analyze radio frequency (RF) signals as they presented a diverse set of volumes covering large areas whose propagation is affected by their environment in which analysts may be interested in seeing trends. Specifically, we are interested in studying campus coverage, signal propagation trends, and areas of potential co-channel interference, which create a source of signal loss due to the overlap of signals in the same frequency channel. The methods developed and explored here will lead to new tools to analyze how various channels may interact on a large scale, such as over a university campus or a smart city.

The data used in this paper have been interpolated after acquisition using Matlab's fit functions. We mapped the data to a two-dimensional uniform grid representing bins of latitude and longitude. We create a unique texture for each network, which is usually defined as a specific SSID or a specific SSID and frequency pair. The textures contain the RSSI (Received Signal Strength Indicator) value sampled at each latitude and longitude. We then used the Matlab biharmonic spline fit to model the router over the whole campus. The resulting function is then output to a binary file for our rendering.

In addition to our volume data, we also use both 3D building models and a campus vegetation inventory with GPS-accurate positioning to help our users orient themselves in the virtual world. Since these building models form a one-to-one mapping to the real world, users will be able to make actionable conclusions from the information, such as where to place additional routers or which routers to move to a new channel to improve the signal landscape.

4. Programmable Transfer Functions

Programmable transfer functions significantly improve traditional transfer functions due to their malleability. In order to edit a conventional transfer function, the user must perform a memory swap, replacing the transfer function array or texture with new data. Data transfer between the CPU and GPU is a significant bottleneck for rendering pipelines. One of our contributions in this paper is the idea of trading the transfer function lookup, which is memory intensive, for a function call at every sample location. Leveraging computation over memory fetches reduces lookup time and enables swift modifications

to the transfer function through parameter modification. We can give the user far more customization and interaction options with this function call. Examples of this increased functionality are shown in Sections 4.2–4.4.

The Programmable Transfer Function is well-suited for visualizing our complex WiFi signal space, as we have many interactions across the dataset to analyze. The interactivity that the Programmable Transfer Function gives significantly benefits a signal analyst. In real time, a user can update the isovalue to analyze stronger or weaker areas and efficiently recognize weak signal strength areas. Furthermore, utilizing the specialized Programmable Transfer Functions listed below, an analyst can easily recognize the shape of the WiFi isosurfaces and find their regions of intersection. These intersection regions are noteworthy as they indicate areas of potential frequency interference or areas where WiFi packet loss may occur due to sharing space on the RF spectrum. These regions of interest in the three-dimensional space would be harder to find using aggregate data, heat maps, or even volume rendering with a traditional transfer function.

4.1. Base Direct Volume Rendering

We perform the volume rendering in this paper in conjunction with traditional mesh rendering using rasterization. This rendering is a part of a multi-step rendering pipeline shown in Figure 1. First, we render a skybox as a background using the standard skybox shaders. We then render the buildings as one large mesh, the campus map as a single textured quad, and the vegetation inventory as multiple instances of a single tree object. Finally, we perform volume rendering with a GUI interface.

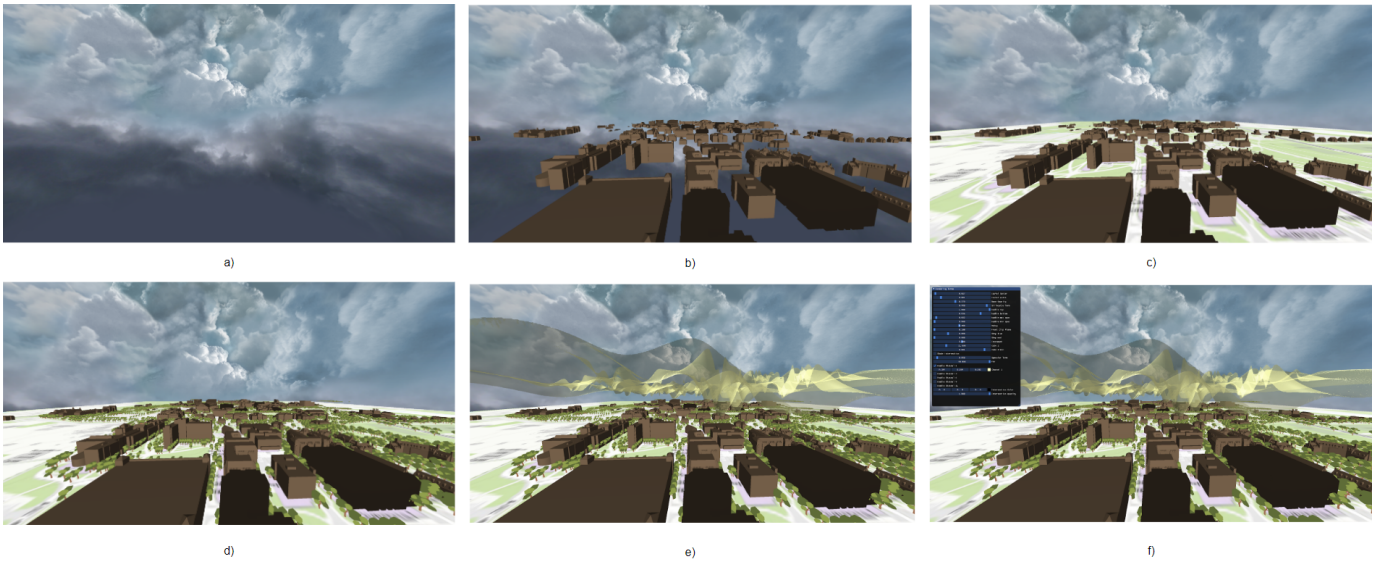

Figure 1. The base rendering pipeline with six steps: (**a**) skybox rendering, (**b**) model rendered with one mesh, (**c**) campus map modeled on a two-dimensional quad, (**d**) instance rendering of the vegetation directory, (**e**) volume rendering using ray marching, and (**f**) IMGUI window rendering.

We decided to use volume rendering to visualize the isosurfaces of the WiFi signals. We chose to visualize isosurfaces because they are easy for users to interpret. Due to the limited overlap region, they are also compact enough to allow the simultaneous rendering of multiple networks. This enables users to determine total coverage on campus and make decisions regarding co-channel interference. Please note that the decision to use isosurfaces was made for our use case of WiFi visualization and does not represent a fundamental limitation to Programmable Transfer Functions. Programmable Transfer Functions enable us to customize isosurfaces' appearance in real time fully. However, indirect volume rendering using traditional isosurface extraction techniques, such as Marching Cubes,

limits how flexible our visual depiction can be. It would also require us to re-extract the surfaces whenever our visualization parameters are updated.

Both the mesh rendering and the volume rendering are OpenGL implementations. We use a bounding cube as an acceleration structure and render the front and back hit points of the bounding cube to a framebuffer object. Then, we use the front- and back-hit points to create the ray representing the light path for that pixel. We step through the volume sampling at each point along the ray. We store the volume data as a flat two-dimensional array. When sampling, we indexed the array based on the latitude and longitude of the sample point and subtracted a value proportional to the z component of the sample position to represent the signal strength fall-off. A traditional volume renderer would take these samples and access a transfer function texture to get a color and opacity at each point on the ray. However, we instead call a function to receive the same information. This function is the Programmable Transfer Function. We then composite all the colors with opacities along the ray to create the final rendering. In order to terminate rays early and allow buildings to occlude the volume, we use the depth buffer from our building rendering; see Figure 2. Early ray termination improves frame rates and provides the occlusion necessary for proper depth perception in the environment. When appropriate, we process multiple volumes by sampling each volume in turn and calculating their respective contribution at each point along the ray. Figure 3 shows the results of this approach. We can implement additional features using various Programmable Transfer Functions from this volume rendering. See in the Figure 4.

When analyzing these renderings, the user should interpret the isosurfaces as indicating equal signal strength. The higher the isosurface in an area is, the higher the WiFi signal strength on the ground beneath it. Any area without a surface overhead is a dead zone with no measurable signal strength. Where volumes overlap, there is a potential for co-channel interference.

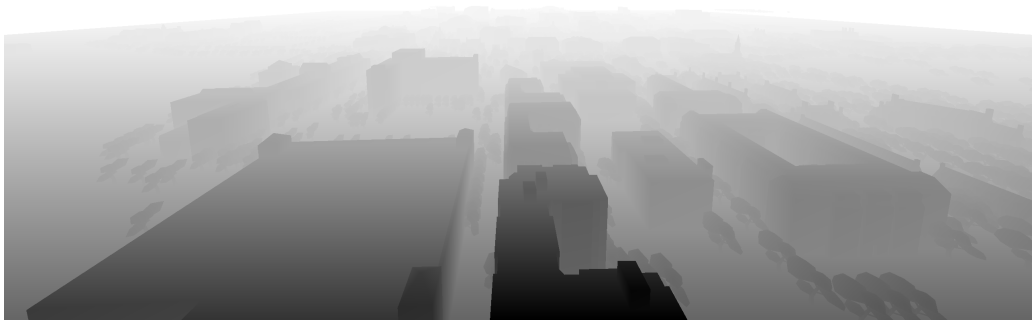

Figure 2. Depth mask used to terminate the volume rendering raycasts in order to enable occlusion and early ray termination. Occlusion will allow the user to maintain their sense of depth, while early ray termination will boost rendering performance.

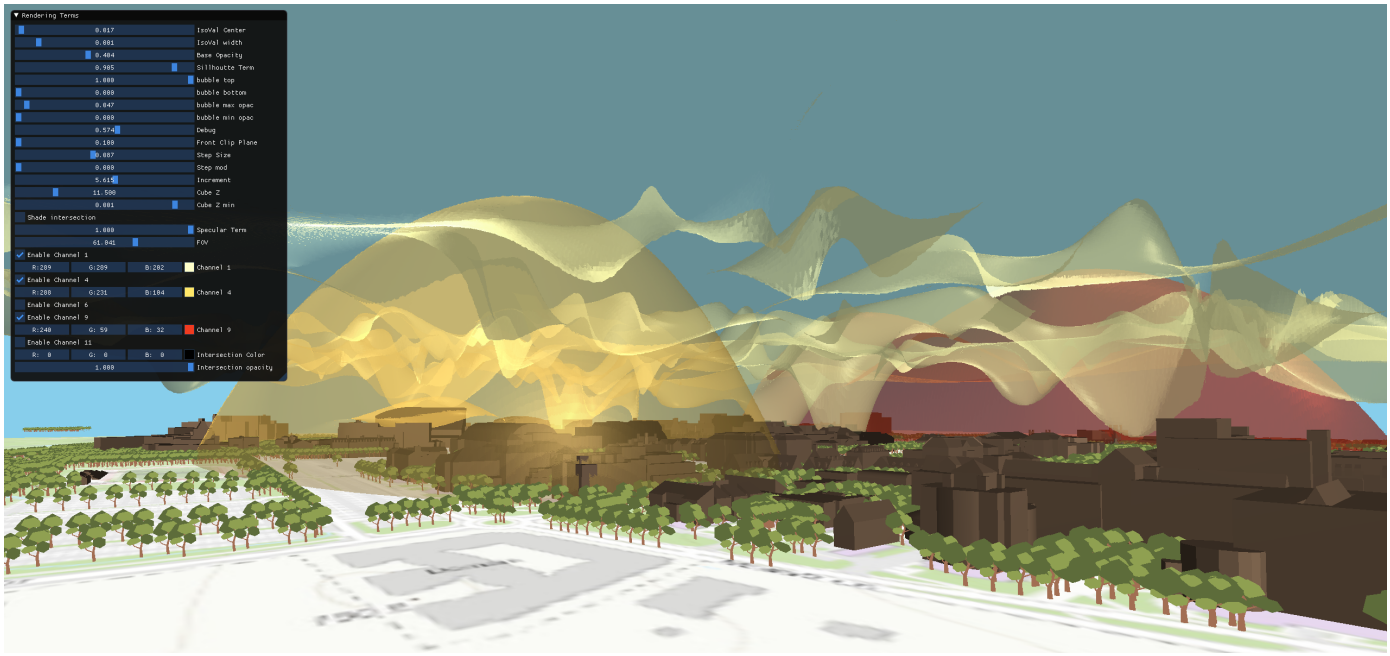

Figure 3. Screen capture of a rendering tool developed to utilize a Programmable Transfer Function which offers user interaction to enhance a data analysis task. Here, five networks are shown over the University of Maryland campus to allow the user to assess signal coverage and frequency interference potential.

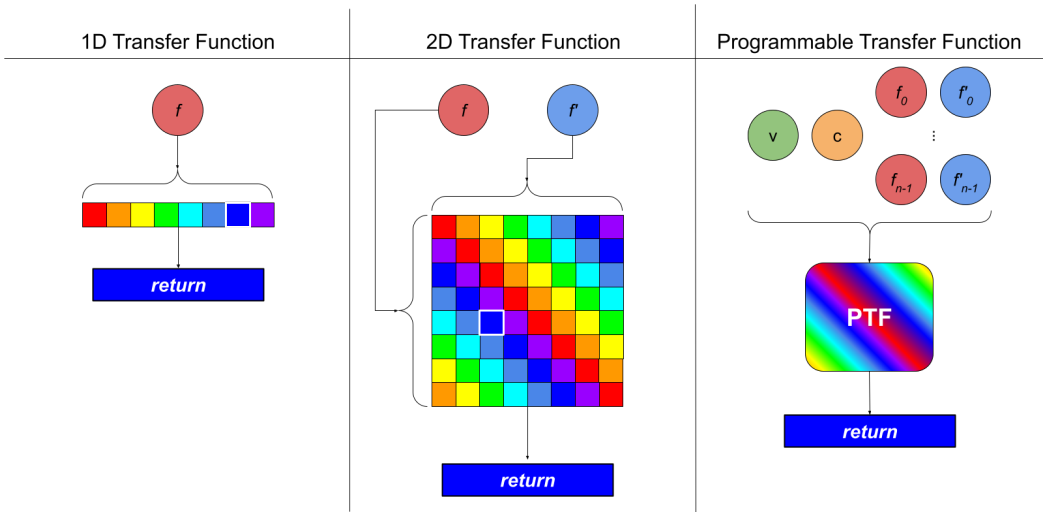

Figure 4. Comparison between (**left**) one-dimensional, (**middle**) two-dimensional, and (**right**) Programmable Transfer Functions. Going from left to right, we observe how the transfer functions generalize to accept a greater number of input parameters. These include view-dependent parameters (v), class labels (c), the vector fields f_i, and their gradients f_i', $0 \leq i \leq n-1$. Note how the increasing multi-dimensionality is succinctly handled by the Programmable Transfer Functions.

4.2. Silhouette Shading

A common problem in many volume renderings is unclear boundaries: specifically, a soft fall-off where a volume ends. This boundary is a concern, as it makes it difficult for an analyst to discern where features are in a volume. We introduce our first use case for a Programmable Transfer Function to address this issue. We use the idea that the silhouette of an isosurface will have normals perpendicular to the viewing angle, whereas the central parts of the isosurface will have normals aligned with the view direction. We can therefore use the dot-product of the isosurface normal vector and the view direction to modify the color and opacity to accentuate the silhouettes of the isosurface. The pseudocode

for this approach is in Algorithm 1. This method produces a bubble-like shading effect. This effect is described in Demir et al. [30]. Programmable Transfer Functions can not only implement the silhouette shading effect; they also enable the user to manipulate the silhouette parameters in real time. In traditional volume rendering, silhouette shading often requires a multi-dimensional transfer function, increasing storage requirements. However, with a Programmable Transfer Function, no additional cost is incurred.

Algorithm 1 Silhouette Shading

for `Each Pixel` **do**
  `RayDir` $\leftarrow$ `(FrontHit - BackHit)`
  for `Each sample along RayDir` **do**
    $\mu \leftarrow 1 -$ DOT(viewDir, normal)
    if $\mu \leq silh_{min}$ **then** $\mu \leftarrow silh_{min}$
    else if $\mu \geq silh_{max}$ **then** $\mu \leftarrow silh_{max}$
    $\mu' \leftarrow \frac{\mu - silh_{min}}{silh_{max} - silh_{min}}(\alpha_{max} - \alpha_{min}) + \alpha_{min}$
    $\alpha_{sample} \leftarrow \alpha_{base} + \mu'$
    $color_{sample} \leftarrow color_{base} + \mu * (1, 1, 1)$
    ...

Note that the silhouette coefficient described above is tuned based on two ranges. One range $[silh_{min}, silh_{max}]$ represents how thick the silhouette augmentation band should be. Typically, $silh_{max}$ is set to 1.0, as this defines the absolute edge. The other range $[\alpha_{min}, \alpha_{max}]$ defines how much the silhouette should be augmented. The results of this algorithm are shown in Figure 5. The silhouette shading would allow a network analyst to see the network's features more clearly, making it easier to draw conclusions from the data. For instance, in Figure 5, an analyst can not see much of the flat features without silhouette shading. With silhouette shading, the analyst can determine that the flat regions represent relatively high signal strength and thus are not worrisome.

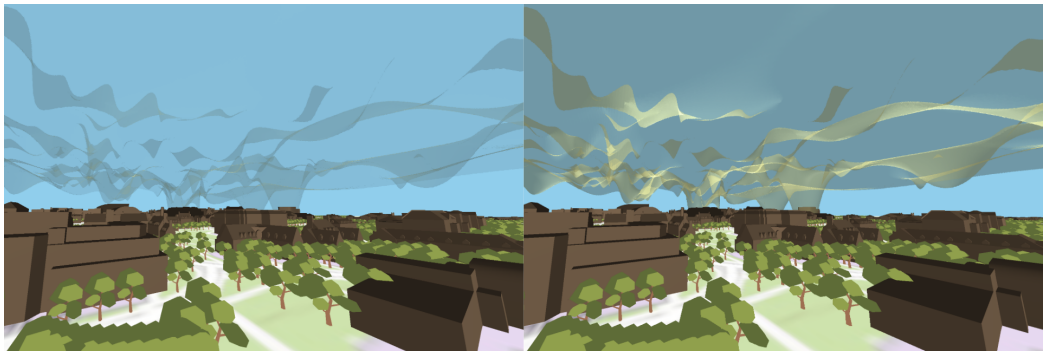

Figure 5. Volume rendering with (**Right**) and without (**Left**) silhouette shading. Notice how this makes it easier to distinguish different parts of the volume and accentuates details. The increased comprehensibility allows the user to make decisions regarding router coverage at a glance, as it is more clear where a volume, and thus a router's signal, ends.

4.3. Specular Highlight Augmentation

When illuminating thin semi-transparent surfaces, it is typical for the specular highlight to be lost. This loss is due to the lack of opacity, diluting the specular contribution. The Programmable Transfer Function can selectively boost the opacity in a region of high specular highlight to mitigate this. The added specular highlight will help users orient themselves in the virtual world and effectively discern the volume's features. Specifically, the specular highlight can help elucidate the curvature of the surface. The utilization of specular highlight in volume rendering is discussed in Fernando [31]. When implementing specular highlights using a Programmable Transfer Function, users can further enhance

comprehensibility by selectively increasing the opacity where a significant specular effect exists. This can emphasize surface shape by reducing the loss of visual appearance of highlights due to volumetric transparency. The specular highlight augmentation is tunable via the parameter μ_{spec} We have observed that the μ_{spec} value depends on the thickness of the surface, the base opacity, and the ray-stepping size. See in the Figure 6, Algorithm 2.

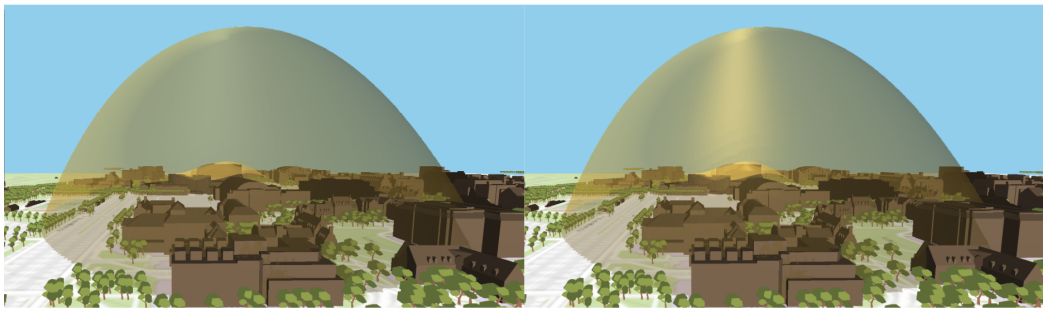

Figure 6. Volume rendering without (**Left**) and with (**Right**) specular highlight augmentation. This highlight will allow the user to understand the curvature of the surface better and make decisions about the environment more intuitively.

Algorithm 2 Specular Highlight Augmentation

```
for Each Pixel do
    RayDir ← (FrontHit - BackHit)
    for Each sample along RayDir do
        dotLightNorm ← DOT(lightDir, normal)
        spec ← POW(dotLightNorm, shininess)
        α_sample ← α_base + spec ∗ μ'_spec
        ...
```

With the aid of specular highlight, the curvature of the volume becomes much more easy to understand, and analysis of the volume becomes more intuitive.

4.4. Multi-Volume Interaction

It can be challenging to distinguish among independent volumes when rendering multifield data, notably when the volumes are semi-transparent. The Programmable Transfer Function can aid the user by highlighting their interaction as done by Jankowai and Hotz [32]. For instance, we can visualize where two volumes intersect and shade these regions a particular color, as in Figure 7. In this example, we are coloring the intersections of the two volumes black to highlight where they meet, but more elaborate interaction visualizations are possible. It is important to note that this style of visualization is best applicable for thin isosurfaces. A different shading model may be necessary for more complicated volumes. We suggest some of these possibilities in Section 5. This feature is valuable, especially in the case of our data where routers communicate on the same frequency band creating a higher probability of destructive interference and, thus, worse signal coverage and lower bandwidth in that area. In general, it also helps analysts determine the depth ordering of the volumes.

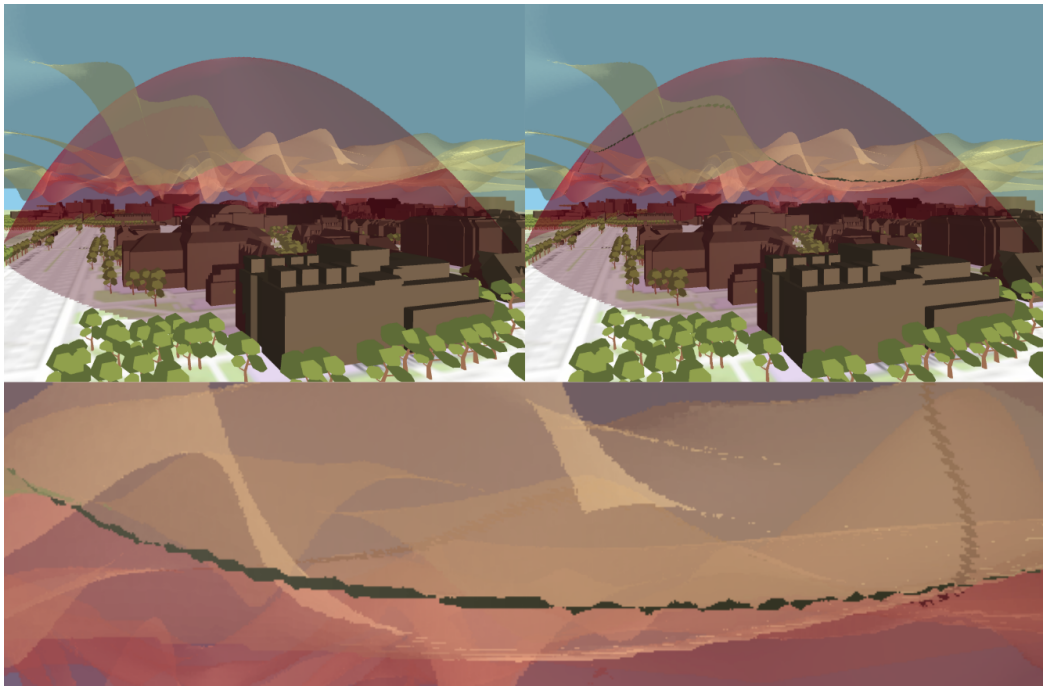

Figure 7. Volume rendering without (**Top Left**) and with (**Top Right**) intersection highlighting. Underneath is a zoomed-in picture of the area of intersection with the highlighting. Intersection highlighting allows a user to better determine the depth ordering of the volumes and understand the arrangement of the surfaces. In our use case, it also serves to highlight regions where co-channel interference is likely. In regions such as the one highlighted here, a network analyst could determine that due to the large region of overlap depicted in this figure, one of these networks should be configured to communicate on another frequency channel.

4.5. Performance

All rendering occurs on a single NVIDIA RTX 2080 with 48 GB of RAM and an Intel Core i7 CPU, and we report frame rates in Table 1. Interestingly, the addition of the specialized Programmable Transfer Functions does not result in any significant frame rate loss. The lack of a performance dip may be due to how OpenGL handles branching conditionals, since all functions are implemented in one shader and toggled through conditional statements. Notice, however, that even when rendering at its worst, the frame rates remain consistently well above our application's interactive threshold of 60 fps. We test each rendering configuration as shown in Figure 8.

Table 1. Performance measured in Frames Per Second (FPS). Each volume represents the signal strength of a single WiFi channel over the campus. Each test case in the single-volume case uses the same volume. For the 5 volume cases, we simultaneously render all channels from this dataset in the 2.4 GHz region. We captured interior and exterior frame rates to note the difference in performance from within and without the volume.

	Base	**Specular**	**Silhouette**	**Spec + Silh**	**Intersection**	**All 3**
1 Volume (interior)	105	105	105	105	NA	NA
1 Volume (exterior)	181	181	181	181	NA	NA
5 Volumes (interior)	85	85	85	85	85	85
5 Volumes (exterior)	116	116	116	116	116	116

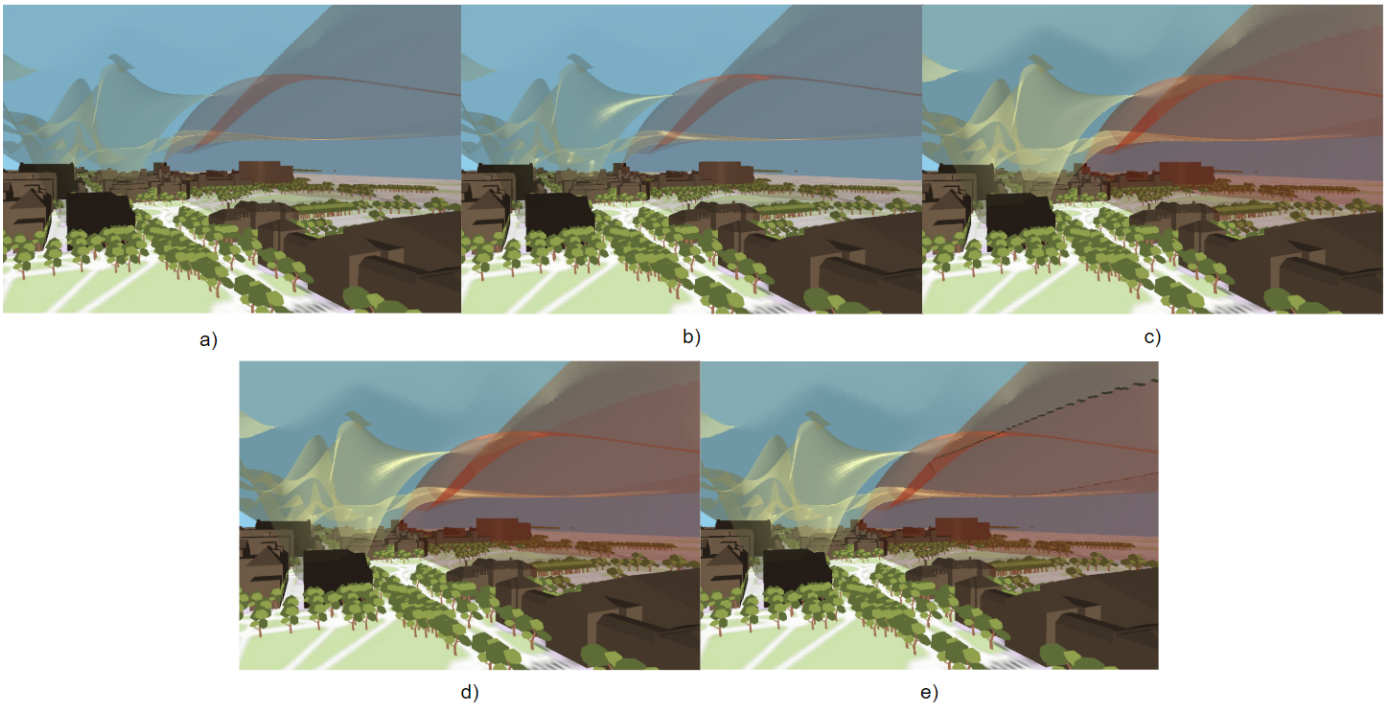

Figure 8. Representation of (**a**) the base volume rendering, (**b**) the rendering with specular highlight augmentation, (**c**) the rendering with silhouette highlighting, (**d**) the base rendering with both silhouette rendering and specular highlight augmentation, (**e**) the full suite with silhouette highlighting, specular augmentation, and intersection highlighting.

4.6. Interaction

We have shown the versatility of the Programmable Transfer Functions for our driving application of WiFi visualization. We have also implemented a GUI in our renderer so that variables in our Programmable Transfer Functions can be changed dynamically, and the user can tune them to produce the rendering they need. In our example, we created an IMGUI window that can modify many rendering aspects. From this window, the user can modify the volume rendering terms such as the isovalue used to render the surface, the color used to shade each surface, and the volume step size. The user can also modify all of the Programmable Transfer Function parameters such as the silhouette term, the silhouette coefficients, and the coefficient of specular highlight augmentation. In addition, the user can toggle the intersection shading on or off. We also control several acceleration techniques from this GUI.

In order to tune the rendering, one only needs to manipulate variables in the GUI with a simple widget, such as a slider or a checkbox, as shown in Figure 9. This ease of use contrasts with how interaction traditionally works, where a new transfer function would have to be computed and stored into a texture.

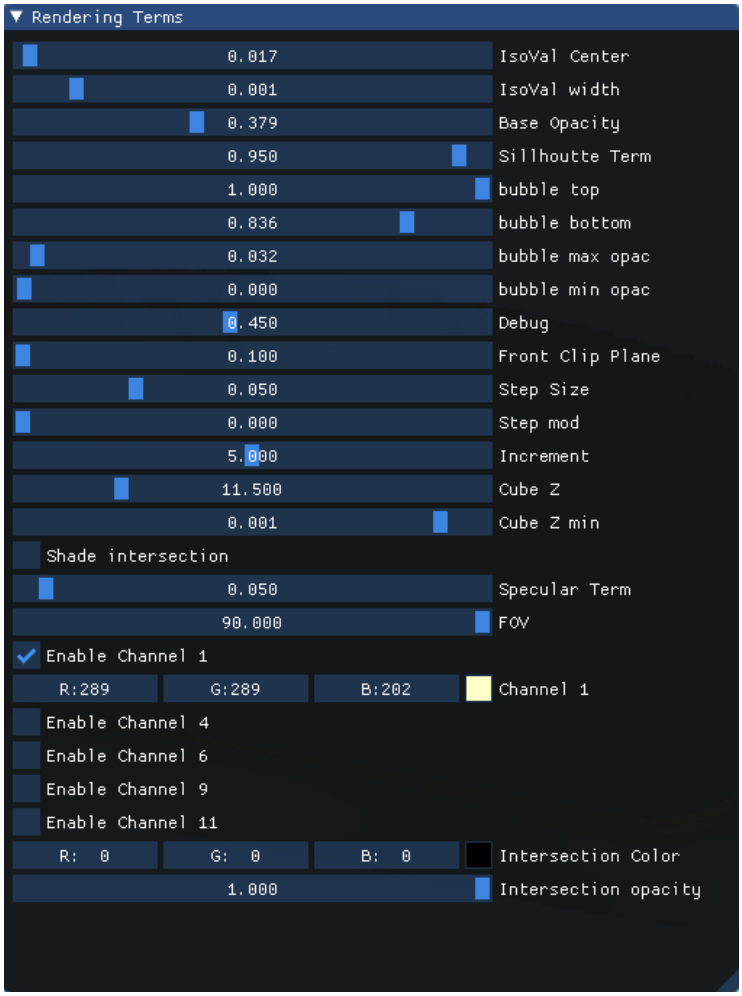

Figure 9. The IMGUI interface for user interaction. This GUI allows the user to take advantage of the flexibility afforded by Programmable Transfer Functions, by allowing them the ability to manipulate rendering parameters, and turn on and off various features such as intersection highlighting.

5. Limitations and Future Work

The application presented in this paper is just an example of what Programmable Transfer Functions can do, but their potential extends beyond what we have presented here. As with all WiFi visualization approaches, our approach is most effective at visualizing a limited number of networks. We have visualized up to six networks at a time. This limit, however, is due to the high-frequency details and scale of WiFi data. In general, Programmable Transfer Functions are scalable. In this paper, we have shown that we can visualize WiFi data using Programmable Transfer Functions. An exciting future work would be to examine the effectiveness of Programmable Transfer Functions for other application domains such as medical visualization or large-scale simulations.

There are many ways in which we are looking to leverage the potential of Programmable Transfer Functions. For ease of discussion, we have divided these into four categories; customized rendering, Section 5.1; data analytics features, Section 5.2; visual enhancements, Section 5.3; and multivolume tools, Section 5.4.

5.1. Customized Rendering

A promising direction for Programmable Transfer Functions is through fully customized rendering. In this method, an expert user would design their shader function during runtime, and the rendering would update in real time. This way, users could interact with the code in whichever way they saw fit. For example, one could imagine a system reminiscent of Unreal Engine material shaders, where users specify inputs and visually

program their desired rendering output. This system would allow arbitrary functionality and complete customization and be an exciting area for future work.

5.2. Data Analytics

The simplest form of data augmentation with a Programmable Transfer Function is data highlighting. For example, one could shade all values above 90% as red and everything else blue, thus highlighting the strongest signals. The Programmable Transfer Function can also leverage supplemental data. For example, in our WiFi signal data, we could shade the region based on which router had the strongest signal strength in that region, or we could mask the signal strengths in a specific region if we knew that data there are corrupted or proprietary. In addition, there are several interesting geometric properties that one could highlight using Programmable Transfer Functions, such as curvature and gradient.

5.3. Visual Enhancements

The silhouette shading and specular highlight augmentation from our implementation section fall into this category. Programmable Transfer Functions can aid in data analytics, but they also generally improve the visual component of the rendering. As an example, we could utilize additional textures to store class-based masks. For instance, a volume from a CT or MRI segmented into known tissues and organs could leverage Programmable Transfer Functions to hide organs and tissues that may be obscuring some feature of interest or highlight a region of interest. This functionality could be a valuable tool for visualizing multi-class data.

5.4. Multivolume Tools

Programmable Transfer Functions could be invaluable in analyzing how multiple volumes interact. This functionality is particularly useful in simulation and sensor data. The data are often in more than three dimensions, and analyzing two variables at once may help analysts identify previously unseen patterns. Algorithm 3 shows our implementation of this method. Instead of just viewing the intersection, one could consider any mathematical formulation of multiple fields, such as their difference or correlation.

Algorithm 3 Multivolume Intersection Shading

```
for Each Pixel do
    RayDir ← (FrontHit - BackHit)
    for Each sample along RayDir do
        hit_volume ← false
        for each volume do
            sample ← TEXTURE(vol_u, vol_v, vol_w)
            if ISINISORANGE(sample) then
                if hit_volume then
                    color_sample = (1, 1, 1)
                    break
                else
                    hit_volume ← true
                    ...                                  ▷ Shade Normally
```

Furthermore, we could design Programmable Transfer Functions to render any feature level-set as defined by Jankowai and Hotz [32]. Programmable Transfer Functions enable the user to choose features on the fly and unconstrained by the need to compute new scalar fields for rendering. For example, a Programmable Transfer Function could visualize the intersection depth for two volumetric scalar fields for any given point. An exciting possibility is to form a conditional operation based on multiple volumetric fields. For instance, one could view the signal strength of a particular SSID and only shade it red if it

is not the maximum signal over a set of SSIDs, therefore representing the set of all SSIDs with the maximum signal strength at each location.

6. Conclusions

Direct volume rendering has made many strides since its origins. With the advances in graphics processing hardware, we can now mathematically calculate the transfer function on the fly rather than storing it in a predefined lookup table. This method allows an analyst to modify the transfer function on the graphics hardware and interact with the volume more efficiently. This new freedom allows for the development of new kinds of transfer functions. No longer constrained by the dimensionality limits, transfer functions will be able to utilize other data sources. We have implemented three specific cases in which a Programmable Transfer Function can be used and suggested many others, but there are far more than we could mention here. We have shown the usefulness of the Programmable Transfer Function for the specific problem of WiFi signal analysis. In particular, we have used direct volume rendering to allow a user to assess the signal coverage at the University of Maryland Campus to conclude the interaction between the signals and their environment. Programmable Transfer Functions can aid in understanding and interpreting the WiFi volumes. Using the multi-volume intersection transfer function, we have also allowed analysts to evaluate the potential for co-channel interference. Programmable Transfer Functions have allowed us to get both of these benefits from one rendering technique. Programmable Transfer Functions also allow for a data-specific transfer function. For example, a function could use one scalar field to mask another, or the interactions between two fields can be visualized expressly in the transfer function. Programmable Transfer Functions offer a new way of thinking for designers of multifield volume visualizations. They enable data scientists to explore their data in a flexible and efficient way while still providing all the functionality of a traditional transfer function. We believe that the use of Programmable Transfer Functions is likely to benefit several other fields beyond WiFi signal analysis.

Author Contributions: Conceptualization, Investigation, Methodology, Visualization, and Writing: All authors; Software: A.R.; Data Acquisition: A.R. and E.K.; Supervision: K.W., E.K. and A.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the US Government contract H98230-19-D00030006. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors, nor does mention of trade names, commercial products, or organizations imply endorsement by the US Government.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The manuscript used the following abbreviations:

PTF	Programmable Transfer Functions
CSV	Comma-Separated File
WLAN	Wide Local Area Network
SSID	Service Set Identifier
BSSID	Basic Service Set Identifier
MAC	Media Access Control
GPS	Global Positioning Systems
RF	Radio Frequency
dBm	Decibel-milliwatts
CT	Computed Tomography

MRI	Magnetic Resonance Imaging
RAM	Random Access Memory
FPS	Frames Per Second
GUI	Graphical User Interface
CPU	Central Processing Unit

References

1. Kokkinos, V.; Stamos, K.; Kanakis, N.; Baumann, K.; Wilson, A.; Healy, J. Wireless crowdsourced performance monitoring and verification: WiFi performance measurement using end-user mobile device feedback. In Proceedings of the 2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Lisbon, Portugal, 18–20 October 2016; pp. 432–437. [CrossRef]
2. Sangkusolwong, W.; Apavatirut, A. Indoor WIFI Signal Prediction Using Modelized Heatmap Generator Tool. In Proceedings of the 2017 21st International Computer Science and Engineering Conference (ICSEC), Bangkok, Thailand, 15–18 November 2017; pp. 1–5. [CrossRef]
3. Radu, V.; Kriara, L.; Marina, M.K. Pazl: A mobile crowdsensing based indoor WiFi monitoring system. In Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), Zurich, Switzerland, 14–18 October 2013; pp. 75–83. [CrossRef]
4. Prentow, T.S.; Ruiz-Ruiz, A.J.; Blunck, H.; Stisen, A.; Kjærgaard, M.B. Spatio-temporal facility utilization analysis from exhaustive WiFi monitoring. *Pervasive Mob. Comput.* **2015**, *16*, 305–316. [CrossRef]
5. Tervonen, J.; Hartikainen, M.; Heikkila, M.; Koskela, M. Applying and Comparing Two Measurement Approaches for the Estimation of Indoor WiFi Coverage. In Proceedings of the 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Larnaca, Cyprus, 21–23 November 2016; pp. 1–4. [CrossRef]
6. Selamat, A.; Fujita, H.; Haron, H. *New Trends in Software Methodologies, Tools and Techniques: Proceedings of the Thirteenth SoMeT_14*; Google-Books-ID: oN3YBAAAQBAJ; IOS Press: Amsterdam, The Netherlands, 2014.
7. Hu, H.; Myers, S.; Colizza, V.; Vespignani, A. WiFi networks and malware epidemiology. *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 1318–1323. [CrossRef] [PubMed]
8. Levoy, M. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.* **1988**, *8*, 29–37. [CrossRef]
9. Drebin, R.A.; Carpenter, L.; Hanrahan, P. Volume rendering. *ACM Siggraph Comput. Graph.* **1988**, *22*, 65–74. [CrossRef]
10. Li, W.; Mueller, K.; Kaufman, A. Empty space skipping and occlusion clipping for texture-based volume rendering. In Proceedings of the IEEE Visualization, VIS 2003, Seattle, WA, USA, 19–24 October 2003; pp. 317–324. [CrossRef]
11. He, T.; Hong, L.; Kaufman, A.; Pfister, H. *Generation of Transfer Functions with Stochastic Search Techniques*; IEEE Computer Society Press: Los Alamitos, CA, USA, 1996. [CrossRef]
12. Kruger, J.; Westermann, R. Acceleration Techniques for GPU-based Volume Rendering. In Proceedings of the IEEE Visualization, VIS 2003, Seattle, WA, USA, 19–24 October 2003.
13. LaMar, E.; Hamann, B.; Joy, K. Multiresolution techniques for interactive texture-based volume visualization. In Proceedings of the Visualization'99 (Cat. No.99CB37067), San Francisco, CA, USA, 24–29 October 1999; pp. 355–543. [CrossRef]
14. Roettger, S.; Guthe, S.; Weiskopf, D.; Ertl, T.; Strasser, W. Smart Hardware-Accelerated Volume Rendering. In Proceedings of the VisSym03 Joint Eurographics-EEE TCVG Symposium on Visualization, Grenoble, France, 26–28 May 2003 ; p. 8. [CrossRef]
15. Bista, S.; Zhuo, J.; Gullapalli, R.P.; Varshney, A. Visual Knowledge Discovery for Diffusion Kurtosis Datasets of the Human Brain. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*; Hotz, I., Schultz, T., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 213–234. [CrossRef]
16. Cheng, H.C.; Cardone, A.; Jain, S.; Krokos, E.; Narayan, K.; Subramaniam, S.; Varshney, A. Deep-Learning-Assisted Volume Visualization. *IEEE Trans. Vis. Comput. Graph.* **2019**, *25*, 1378–1391. [CrossRef] [PubMed]
17. Sharma, O.; Arora, T.; Khattar, A. Graph-Based Transfer Function for Volume Rendering. *Comput. Graph. Forum* **2020**, *39*, 76–88. [CrossRef]
18. Pflesser, B.; Tiede, U.; Höhne, K.H. Towards Realistic Visualization for Surgery Rehearsal. In *Computer Vision, Virtual Reality and Robotics in Medicine*; Ayache, N., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1995; pp. 487–491. [CrossRef]
19. Carpendale, M.S.T.; Cowperthwaite, D.; Fracchia, F. Distortion viewing techniques for 3-dimensional data. In Proceedings of the IEEE Symposium on Information Visualization'96, San Francisco, CA, USA, 28–29 October 1996; pp. 46–53. [CrossRef]
20. Ip, C.Y.; Varshney, A.; JaJa, J. Hierarchical Exploration of Volumes Using Multilevel Segmentation of the Intensity-Gradient Histograms. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 2355–2363. [CrossRef] [PubMed]
21. Kniss, J.; Kindlmann, G.; Hansen, C. Multidimensional transfer functions for interactive volume rendering. *IEEE Trans. Vis. Comput. Graph.* **2002**, *8*, 270–285. [CrossRef]
22. Viola, I.; Kanitsar, A.; Groller, M. Importance-driven volume rendering. *IEEE Vis.* **2004**, *2004*, 139–145. [CrossRef]
23. Treavett, S.; Chen, M. Pen-and-ink rendering in volume visualisation. In Proceedings of the Visualization, VIS 2000, (Cat. No.00CH37145), Salt Lake City, UT, USA, 8–13 October 2000; pp. 203–210. [CrossRef]
24. Csébfalvi, B.; Mroz, L.; Hauser, H.; König, A.; Gröller, E. Fast Visualization of Object Contours by Non-Photorealistic Volume Rendering. *Comput. Graph. Forum* **2001**, *20*, 452–460. [CrossRef]

25. Linsen, L.; Van Long, T.; Rosenthal, P.; Rosswog, S. Surface Extraction from Multi-field Particle Volume Data Using Multi-dimensional Cluster Visualization. *IEEE Trans. Vis. Comput. Graph.* **2008**, *14*, 1483–1490. [CrossRef] [PubMed]
26. Kniss, J.; Hansen, C. *Volume Rendering Multivariate Data to Visualize Meteorological Simulations: A Case Study*; The Eurographics Association: Munich, Germany, 2002.
27. Kniss, J.; premoze, S.; Ikits, M.; Lefohn, A.; Hansen, C.; Praun, E. Gaussian transfer functions for multi-field volume visualization. In Proceedings of the IEEE Visualization, VIS 2003, Seattle, WA, USA, 19–24 October 2003; pp. 497–504. [CrossRef]
28. Jang, Y.; Botchen, R.P.; Lauser, A.; Ebert, D.S.; Gaither, K.P.; Ertl, T. Enhancing the Interactive Visualization of Procedurally Encoded Multifield Data with Ellipsoidal Basis Functions. *Comput. Graph. Forum* **2006**, *25*, 587–596. [CrossRef]
29. Sauber, N.; Theisel, H.; Seidel, H.P. Multifield-Graphs: An Approach to Visualizing Correlations in Multifield Scalar Data. *IEEE Trans. Vis. Comput. Graph.* **2006**, *12*, 917–924. [CrossRef] [PubMed]
30. Demir, I.; Kehrer, J.; Westermann, R. Screen-space silhouettes for visualizing ensembles of 3D isosurfaces. In Proceedings of the 2016 IEEE Pacific Visualization Symposium (PacificVis), Taipei, Taiwan, 19–22 April 2016; pp. 204–208. [CrossRef]
31. Fernando, R. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*; Pearson Higher Education: San Francisco, CA, USA, 2004; Chapter 39.
32. Jankowai, J.; Hotz, I. Feature Level-Sets: Generalizing Iso-Surfaces to Multi-Variate Data. *IEEE Trans. Vis. Comput. Graph.* **2020**, *26*, 1308–1319. [CrossRef] [PubMed]