



# ReViVD: Exploration and Filtering of Trajectories in an Immersive Environment using 3D Shapes

François Homps, Yohan Beugin, Romain Vuillemot

## ► To cite this version:

François Homps, Yohan Beugin, Romain Vuillemot. ReViVD: Exploration and Filtering of Trajectories in an Immersive Environment using 3D Shapes. IEEE VR, Mar 2020, Atlanta, United States. 10.1109/VR46266.2020.1581269207852 . hal-02482695

**HAL Id: hal-02482695**

**<https://hal.science/hal-02482695>**

Submitted on 18 Feb 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# VRT: Exploration and Filtering of Trajectories in an Immersive Environment using 3D Shapes

Category: Research

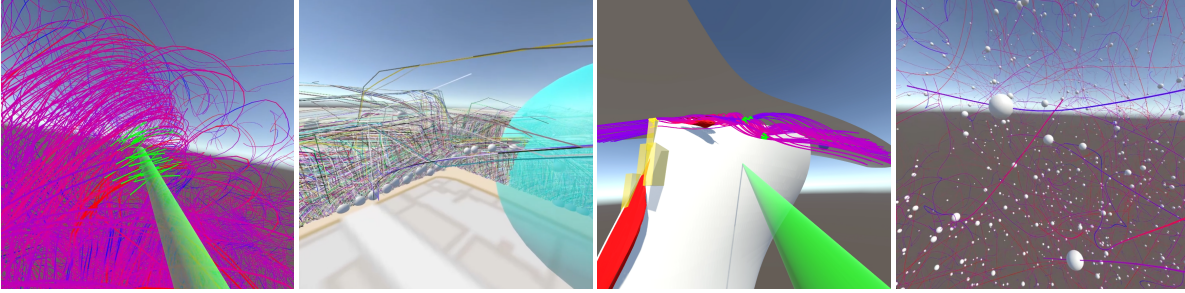


Figure 1: VRT enables dynamic filtering of large trajectory datasets using 3D shapes. From left to right: (a) A *cylinder* used as a saber to select and filter particles from a numerical simulation dataset, (b) a *sphere* used to select and animate vehicles at a road junction in a car traffic dataset. Such shapes can become (c) *persistent* to refine queries and extract complex vortex, and trajectories can be (d) *animated* to re-create turbulent flows of particles.

## ABSTRACT

We present VRT, a tool for exploring and filtering large trajectory-based datasets using virtual reality. VRT’s novelty lies in using simple 3D shapes—such as cuboids, spheres and cylinders—as queries for users to select and filter groups of trajectories. Building on this simple paradigm, more complex queries can be created by combining previously made selection groups through a system of user-created Boolean operations. We demonstrate the use of VRT in different application domains, from GPS position tracking to simulated data (e. g., turbulent particle flows and traffic simulation). Our results show the ease of use and expressiveness of the 3D geometric shapes in a broad range of exploratory tasks. VRT was found to be particularly useful for progressively refining selections to isolate outlying behaviors. It also acts as a powerful communication tool for conveying the structure of normally abstract datasets to an audience.

**Index Terms:** Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual Reality; Human-centered computing—Visualization—Visualization systems and tools—Visualization toolkits

## 1 INTRODUCTION

Many fields generate large trajectory datasets: GPS trackers on aircrafts are used for real-time monitoring of thousands of moving objects; road traffic simulations give insights on how to prevent traffic jams in a city; particle simulations of turbulent flows make it possible to identify phenomenon normally invisible due to their small scale in space and time. Those domains usually involve domain-expert users who need to explore their dataset to assess the structure and quality of collected or generated data [25].

Interaction is key during the exploration process, in particular to isolate elements of interest. For instance, when interacting with aircraft datasets, a frequent exploratory task is to filter by take offs and landings, for a given specific airport area. Assuming the planes’ positions are displayed in 3D with coordinates (x, y, z), and their trajectories result from their tracking over time (t), selecting planes is achieved by having constraints on both 1) the 2D position on the ground (a circle centered on the airport), and 2) maximal altitude. Performing such a selection is difficult with standard 2D dynamic queries [22] as they only operate on a plane.

Immersive environments demonstrate great promise to explore and interact with 3D datasets [5]. Recent work includes Fiberclay [15], which provides a tilted 3D view, direct brushing and selection interactions. However, those selections are not localized and also select fly-overs; the process requires deselecting trajectories from another point of view.

This paper seeks to introduce a simple yet general-purpose interaction technique that operates simultaneously on the three spatial dimensions. The technique uses 3D geometric shapes (spheres, cuboids and cylinders) that encode a 3D query which the user manipulates to probe a highly cluttered environment [12]. Our research hypothesis is that such 3D geometric queries are not only expressive enough for open-ended exploration tasks, but also to single out specific elements. To validate this hypothesis, we present VRT [name changed for peer review], a VR trajectory data visualizer in which we implemented the technique and tested it against different application domains and datasets. Our paper has three main contributions:

- The first contribution is to present a visualization toolkit that allows non-experts in Virtual Reality to intuitively create and use simple 3D geometric shapes as selection tools, and to filter the dataset through the use of custom Boolean operations.
- The second contribution of this paper is the implementation of persistent selectors that allow the reuse of specific queries.
- The third contribution is the mapping choice of all the interactions possible in VRT on the limited inputs available with the VR controllers.

## 2 RELATED WORK

### 2.1 Visualization in Immersive Environments

Immersive environments (e. g., VR, augmented reality, interactive walls) have demonstrated great promise for the exploration of abstract datasets [5]. In particular, recent years have seen a fast democratization of Virtual Reality (VR) technology: newer headsets with much more resolution and precise user’s movements, VR support in consumer-grade graphics card and the emergence of high-level development software such as Unity<sup>1</sup>, along with standardized libraries

<sup>1</sup><https://unity.com/>

like Valve’s OpenVR<sup>2</sup>, have made standardized VR technology accessible.

In the meantime, the task of exploring large datasets has become pervasive in all research and engineering domains. We found that while extremely powerful visualization tools, such as Kitware’s ParaView<sup>3</sup> and its plugins like the Topology Toolkit [24] exist, they lack the ease of use and intuitiveness that VR shines in.

This is why we started work on VRT, a lightweight open-source tool aimed at science and engineering domain-experts specialized in quickly visualizing and filtering trajectory datasets in VR.

Indeed, when compared to traditional 3D visualizers, VR shows high potential. Users can immerse themselves in the dataset with many more degrees of freedom than through a traditional mouse and keyboard interface, both for the control of their point of view and of their interactions with the data. However, these benefits come at a cost; depth perception and comfortable viewing are only possible on dual high resolution screens, which greatly impact the rendering performance. Meanwhile, a high frame rate (traditionally, a minimum of 30 fps) is essential in VR to reduce risks of inducing nausea.

## 2.2 Exploration in Immersive Environments

Exploration techniques in those settings often rely on displaying large amounts of data for the user to navigate within, with no previous assumptions (e. g., filtering, aggregation) made. Fiberclay [15] is a flagship example that displays thousands of trajectories and lets user build queries using beams. ImAxes [10] and IATK [9] permit the creation of axes to map data attributes to spatial locations and create standard charts (e. g., parallel coordinates, scatterplots) based on logical rules. Those techniques have in common that they require no user interface to switch between interaction modes: only the users’ VR controllers are visible in the virtual environment. VRT is heavily inspired by this approach and augments the controllers to display current possible queries as 3D shapes called selectors, as well as operation menus used to filter the dataset by combining multiple selections.

## 2.3 Dynamic Queries and Brushing

Dynamic queries [22] set the foundation for interactive exploration of large datasets. They enable the selection of value ranges in an iterative and safe way and are informative as they are coordinated to visual feedback. Such queries enable users to grasp the structure and distribution of the datasets through transient selections, which are key for exploratory data analysis [25]. Our work is built on this paradigm, and in particular *brushing*, which complies with the direct manipulation paradigm to let users manipulate regions of interest on the screen in a natural way. Brushing has been extended beyond 2D environments such as in 3D [21] and immersive environments, e. g., for Cave interactions [23], in Virtual Reality [15, 20], or with tangible interactions [4, 17]. However, brushing specific areas or volumes remains challenging especially with large datasets due to occlusions.

## 2.4 3D Occlusions and Selection Tools

3D environments require occlusion management [12], as objects in a scene can mask each other, especially when their density is important. To facilitate large selections, different techniques have been used in VR such as hand avatars, bimanual selection, 3D cursors and raycasting [2]. While ray casting allow ease of interaction and selection in Virtual Reality and seems to have become a default technique [2], it does not allow for distinguishing between objects that are both on the infinite ray. In this case, other techniques such as shapes (e. g., squares, circles) can be used to select all items

within them [7, 29]. In a 2D space, rectangle shapes are standard as they can easily be built using mouse selection [22], but if more attributes are being used, complex shapes can emerge [13], e. g., by using a non-visual attribute. In 3D worlds, 3D shapes (e. g., cuboid, sphere) enable the manipulation of spatial regions in an efficient way to select groups of objects and so progressively refine the selection [16, 18]. BalloonProbe [11] uses spherical shapes to let users separate objects based on their attributes in a 3D scene. Bounding boxes of groups of points in a 3D space [14] can become interactive to select items. Balloon Selection [3] also lets users create and translate spheres in a 3D space for target selections, with a combination of 2 DOF interactions. Flat 2D planes can be used for spatial structure projection, e. g., in brain imagery [1, 6, 8]. To manage this problem, already lessened by the small footprint of the trajectories, our work capitalizes on using a variety of probes both in shapes and sizes, as well as by providing an ease of navigation to make better points of view quickly available to the user.

## 3 FEATURES DESIGN

The design approach of VRT is based on three core features illustrated Fig. 2:

1. Visualization: VRT allows the user to see a 3D rendering of the raw dataset and easily change their point of view through intuitive game-like movement controls.
2. Selection: intuitive arm motions allow the user to select the parts of the dataset they want to filter by touching them with configurable 3D shapes called selectors.
3. Filtering: the user can create custom Boolean operations to control the visibility of the trajectories they selected.

This section details the design of those three features.

### 3.1 Immersive Visualization of Trajectory Data

At the core of VRT is its ability to display trajectory data in VR. The VRT engine renders a visualization as an ensemble of independent *paths*, each made of many small 2D segments called *ribbons*. Each ribbon serves to link two adjacent points of the path. We name each of these points *atoms*, as they can hold more attributes than spatial coordinates (e. g., a time value, the current kinetic energy of a particle, etc.). Examples of visualizations can be seen Fig. 1.

Additionally, VRT enables time-based animation of trajectories using small spheres that follow each path from its start to its end (optionally using its atoms’ time values for reference if applicable). The position, radius, and animation speed of these spheres can be configured in real time, making them a good tool to expose invisible implicit attributes such as speed.

Trajectories can also be animated from a shared starting position, i. e. a given spatial selection from which a playback starts, ignoring the context of the actual starting time of each path. We call this technique “dropping” spheres on a selection.

### 3.2 Selection using 3D Geometric Queries: Selectors

A selector is a tool for selecting trajectories; it takes the form of a colored 3D shape attached to the hand of the user. It functions as a 3D brushing tool, allowing the user to make 3D geometric queries on the dataset. There are six colors available for selectors (red, green, blue, yellow, cyan and magenta). Thus, the user has six stored selectors that they can use by switching from one color to the other.

When the user presses a physical selection trigger (Fig. 3), the selector becomes active, and all visible ribbons touching the selector takes the color of the selector. This notifies the user that the paths those ribbons belong to have been selected and put in that selector’s color group (used afterwards through operations). The selector then

<sup>2</sup><https://github.com/ValveSoftware/openvr>

<sup>3</sup><https://www.paraview.org/>

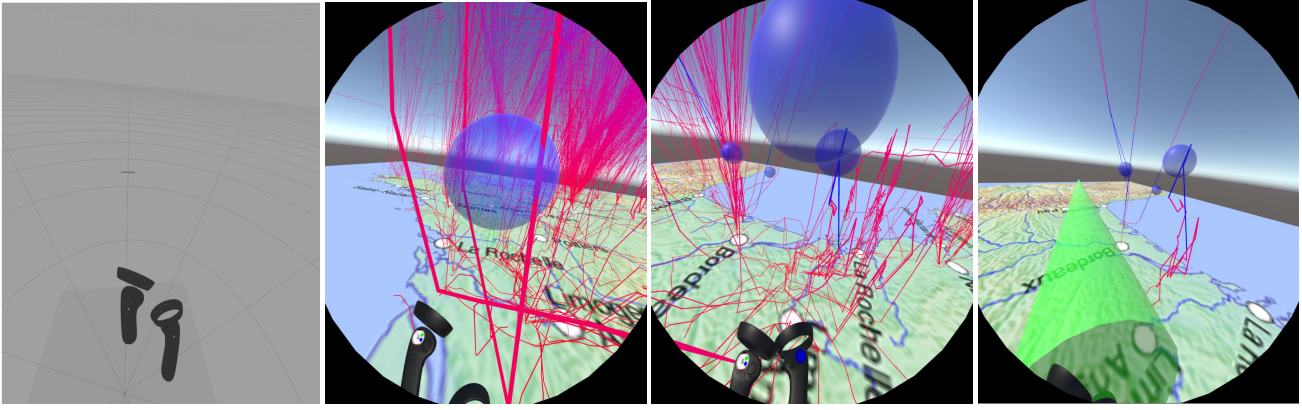


Figure 2: Overview of a standard session of exploration using VRT. From left to right: (a) SteamVR environment seen while configuring and loading the session; (b) Hand-held selection using a sphere; (c) Persistent selectors used for repeated selections; (d) Filtering trajectories based on previous selections.

stays active until the user lets the trigger go, allowing for selection through painting-like strokes. It is important to note that while only a part of the path is colored, the entire path is considered selected in that color. This partial coloration does not hide too much of the original color of the path, which can be configured to relay information.

We include three selector shapes in VRT: *cuboids*, *spheres* and *cylinders* (see Table 1). These shapes can be manipulated through *parameters* which define their size as well as their position and rotation relative to the hand of the user. Controllable parameters, listed Table 1, are intentionally restrictive. This is necessary in order to not confuse the user with too many possible operations on the restricted number of buttons available on the VR controllers. Cuboids have the additional limitation of being rotation-locked on both horizontal axes; this way, their bottom side is always parallel to the ground of the visualization, making axis-aligned queries easier to perform.

The three selector shapes were chosen for their utility in different use cases. Spheres are mostly useful for selecting all trajectories passing through a specific point of interest, or used as a detached pointer to select specific trajectories at a medium distance from the user. Cylinders are often used as a kind of sword, slicing through a bundle of similarly-oriented trajectories to select them all. Cuboids, thanks to their rotation lock, are more useful as walls, ceilings or floors delimiting certain regions of space, e.g., for selecting all trajectories going above a certain height.

### 3.2.1 Persistent selectors

It can often be interesting to perform the same query multiple times - for example, when redoing a selection after having hidden a subset of ribbons. In this case, using hand-held shapes to try to make the exact same query is cumbersome. To solve this problem, we introduce *persistent* selectors: by pressing a button, the user can clone the currently hand-held selector in the 3D space as a persistent selector, which won't move with the user and can be activated at a distance when necessary.

Persistent selectors also give the user the opportunity to seed multiple points of interest ahead of time, to use later in combined queries - this is especially interesting in fields focused on such points of interest, e.g., with traffic data.

### 3.3 Using Selections to Filter Data: Operations

Once selections have been done, the user can utilize the color groups they populated to filter the currently displayed trajectories. This process of filtering is done through the use of operations that the


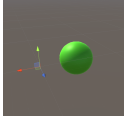
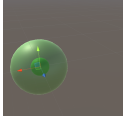

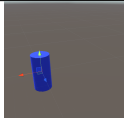
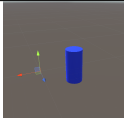
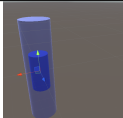
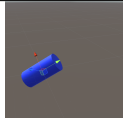

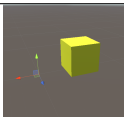
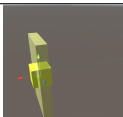
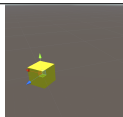
Shape	Parameters relative to the hand of the user		
	Offset	Size	Rotation
 Sphere	 Offset (Vector3)	 Radius (float)	 RotXYZ (Vector3)
 Cylinder	 Offset (Vector3)	 Radius(float) Width (float)	 RotXYZ (Vector3)
 Cuboid	 Offset (Vector3)	 SizeXYZ (Vector3)	 RotZ (float)

Table 1: Selector shapes implemented in VRT and their parameters

user can create on the fly. Operations are characterized by their *type* (OR or AND) and the color groups they hold (which can be any combination of the six selector colors).

An OR-type operation will, out of all currently displayed paths, hide all but those that are selected in at least one of the operation's color groups; as an example, the "Blue, Green, OR" operation will only keep displayed the trajectories that are either selected in blue or green.

An AND-type operation will only keep previously-displayed trajectories that are selected in all of the operation's color groups. Thus, the "Blue, Green, AND" operation would this time only keep displayed the trajectories that are selected in blue and green.

Of importance is the fact that operations and selections only affect currently displayed trajectories. This adds another level of flexibility to VRT's filtering system, as consecutive filtering operations will be able to complement each other.



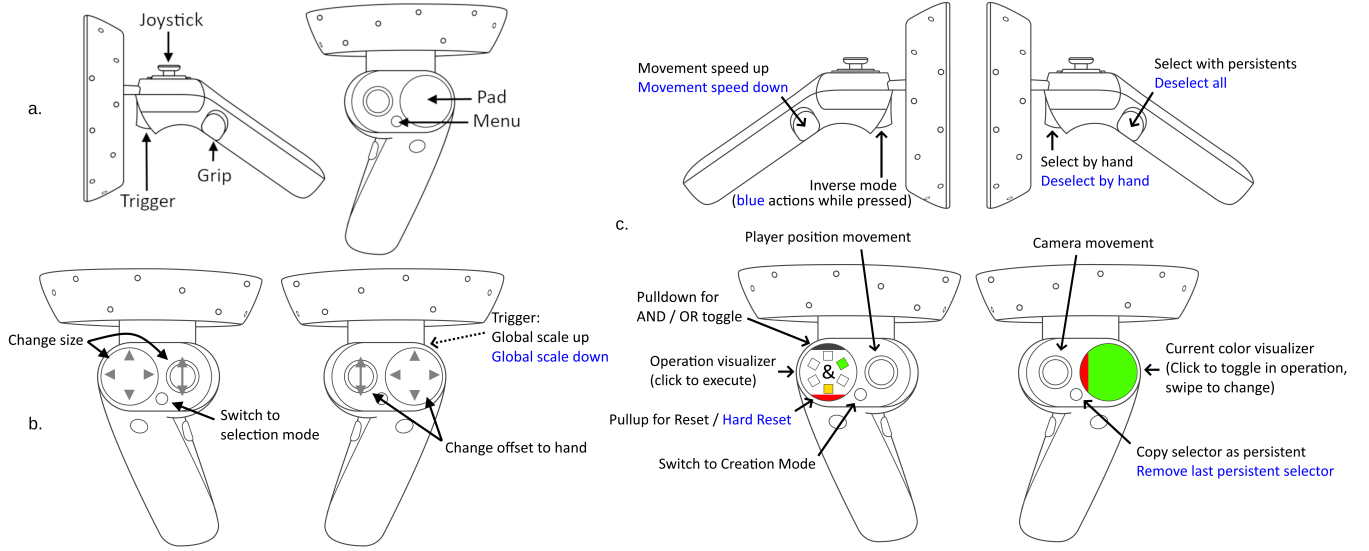


Figure 3: Description of the input of the controllers. Left: (a) Input buttons names, (b) Input mapping of "creation" mode. The controls for "creation" mode slightly differ for different selector shapes. Right (c): Input mapping of "selection" mode.

#### 4 CONTROLLER MAPPING AND USER EXPERIENCE

Apart from animation controls, all previously introduced features of VRT are accessible directly on the controllers. We did not implement any large menu on the screen, to let the user explore the visualization with their vision unobstructed.

The control scheme in VRT was designed to give intuitive access to the previously described features, despite being restrained by the limited set of controls offered by VR controllers. Since standardized controls for VR devices are not yet available, this section is largely specific to the Samsung HMD Odyssey<sup>4</sup> controllers we used during the development of the project, however, most if not all of these controls could be adapted to different devices. The nomenclature of the different inputs and all of VRT's controls are respectively illustrated in Fig. 3 (a), (b) and (c).

##### 4.1 Control Modes

In order to limit the complexity of the controls, two separate control layouts are accessible and toggled at the press of a button: selection mode and creation mode. In selection mode, the user can select trajectories using the selectors, switch between different selector colors, place and remove persistent selectors, and create and execute filtering operations. Creation mode is exclusive to selector modification, allowing the user to position the current selector relative to their hand and to change its dimensions; configurable parameters for each type of selector are summarized in Table 1.

To broaden the range of possible inputs, an "inverse" trigger is used: while the left controller's trigger is pressed, some controls are modified (usually to have the opposite of their normal effect).

##### 4.2 Operation Controls

Displayed on the left controller's pad are six initially blank squares, representing the six colors groups previously mentioned (Fig. 3, c). Those squares encircle a symbol which shows the operation's type: either "&" (AND-type) or "||" (OR-type).

By clicking the right controller's pad, which displays the current selector's color (Fig. 3, c), this color's presence in the operation will be toggled. When a color is to be used in the operation, its

corresponding square will be filled. This way, the left pad provides a complete overview of what the operation will do. A pull-down menu on the left controller's pad can be clicked to toggle the operation's type. All changes to the operation are also signaled through vibrations in the left controller, such that an experienced user will not have to look at it to know the operation's state.

Finally, the created operation can be applied to the visualization by pressing the left controller's pad (Fig. 3, c). Resetting the effects of operations requires pulling up from the bottom part of the left pad to bring up the "RESET" button, then pressing it. Finally, while the "Inverse" trigger is pressed, the left pad changes to display the word "Invert"; pressing the left pad at this moment will execute the "Invert" selection, toggling the visibility of all the paths in the dataset. This special operation is not influenced by the color groups or type of the standard operation.

#### 5 IMPLEMENTATION NOTES AND OPTIMIZATIONS

VRT is implemented in C# using Unity<sup>5</sup>, a popular 3D real-time rendering engine traditionally used for video game development. We used Samsung's HMD Odyssey as VR equipment, with Alienware laptops to help achieve real-time rendering. Our implementation uses Valve's SteamVR library to interface with our VR equipment. VRT's algorithms could however be re-implemented using other software and hardware solutions.

VRT is released as an open source project with permissive licence. Code and documentation are available on GitHub [Left blank for peer review].

While we relied on a well established stack of software and devices, an important development effort was required to achieve solid rendering performance and fast collision detection. The following sections provide a high-level overview of those algorithms and discuss the impact they had on the design of VRT.

##### 5.1 Trajectories Rendering

A crucial constraint in exploratory data analysis is to build techniques with as little latency as possible; and in VR software, maintaining a solid framerate is critical. Indeed, bad performance will cause poor synchronisation of the video feedback to the user's movements,

<sup>4</sup><https://www.samsung.com/us/computing/hmd/windows-mixed-reality/>

<sup>5</sup><https://unity.com/>

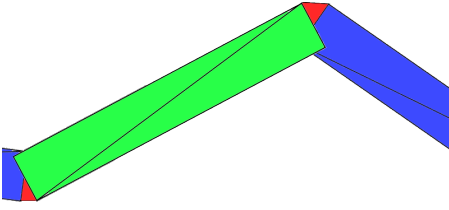


Figure 4: Structure of the procedural mesh used to render trajectories. In green and blue, ribbons; in red, junction triangles at the intersection of ribbons.

quickly inducing nausea; thus, VRT has to be able to render the trajectories of a dataset at least 30 times per second.

To alleviate the load on the graphics card used, we chose to render paths with a minimal amount of triangles through procedural mesh creation. Each ribbon (individual segment of a path) is made of two triangles forming a rectangle; an additional triangle is rendered between each two consecutive ribbons to soften the junction between the two rectangles. Individual meshes are procedurally created for each path by linking all of the path’s ribbons. This vertex configuration is shown Fig. 4.

Compared to an approach where actual 3D cylinders are rendered, this method produces cleaner results and uses fewer triangles and memory, however, it requires the ribbons to be rotated towards the camera at all time. This was done through the use of a custom HLSL (High Level Shading Language) shader.

This system allows us, on our computers equipped with Nvidia’s 1080Ti (Max-Q design), to sustain an acceptable framerate up to around 20 million on-screen atoms (constitutive points of the trajectories).

## 5.2 Collision Detection

Whenever the user presses the selection trigger, the selector in their right hand becomes active, selecting and painting all the ribbons it touches in its color. Unity provides efficient methods for checking if two individual 3D shapes collide. However, a naive implementation would need to apply this check to all the ribbons in the scene and repeat this intensive operation each frame that the selector is active, making for unacceptable performance.

To alleviate this problem, space in VRT is divided in small cuboid sections we call *districts*. Each district keeps track of all the ribbons inside it. This partitioning allows the program to quickly get a gross approximation of which ribbons are in a specific spatial zone and to restrain the computationally expensive accurate collision detection algorithm to only those ribbons.

In order to know which districts to check for collisions, we implemented a shell-filling algorithm, particularly efficient for selections with large selectors.

1. First, the center of the selector is used to determine a district necessarily touched by the selector, which we call the “center district”.
2. A line of districts stemming from the center district in an arbitrary direction is progressively checked until a district at the edge of the selector is found (which we call a “border district”).
3. A flooding algorithm starting from the found border district finds all the border districts around the selector, creating an hermetic “shell” of border districts.
4. Starting again from the center district, a flooding algorithm finds all the districts enclosed by the shell; those districts, fully comprised within the selector, are called “inside districts”.

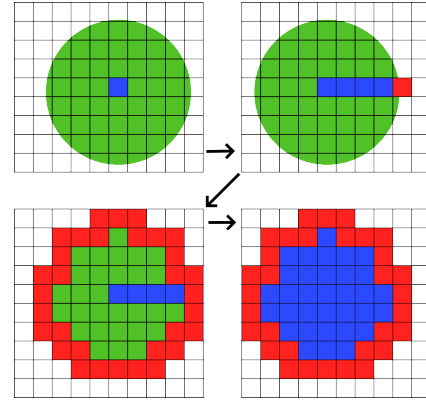


Figure 5: Four phases of the collision detection algorithm. In green, the selector; in blue, “inside”-type districts; in red, “border”-type districts.

Once this has been done, the algorithm can safely assume that all ribbons pertaining to “inside” districts are touched by the selector without need for further calculations. Afterwards, only ribbons exclusive to “border” districts will be fully tested to determine if they touch the selector or not.

This algorithm, summarized Fig. 5, essentially reduced the complexity of collision detection from  $O(N^3)$  to  $O(N^2)$  on very large selectors, as only the surface of the selector gets “deeply” checked.

## 6 USER EVALUATION

During the design of VRT, we followed the Munzner [19] nested model design methodology by constantly engaging with many potential end-users from various domains, involving both human behavior trajectory recordings as well as simulations. This phase spanned over 9 months in total and helped us to validate the *problem*, *data abstraction* and *encoding* steps of the model. This process remained highly informal, with multiple exchanges of sample datasets, questions, and screenshots of the prototype. We validated the *algorithm part* of the nested model by achieving real-time frame-rate for the datasets we used in our experiments.

We conducted a user evaluation for the *interactions encoding* part of the nested model during formal sessions with domain-expert participants who were not involved in the design process. We sought for experts who needed to visualize large amounts of data, but with a lack of expertise in building interactive systems. In total we contacted 6 experts, either from our local university or from professional networks who submitted a dataset to load into VRT to us (as a technical procedure was needed to clean, format and scale the dataset). We also added static 3D models and background elements useful for context (e. g., rugby pitch, map of a city/country) (see details Table 2). We invited them to one or two evaluation sessions (depending on their availability or for follow-ups related to adding more or different datasets). Due to limited space, we report only on 4 case studies we selected based on the diversity of datasets and tasks.

Each session started with a training procedure. After briefly presenting the tool, the HMD, and the controllers, the participants used the HMD for 30 minutes in a guided tutorial visualization of aircraft trajectories over a country over a day. This dataset allowed participants to use VRT with a realistic exploration task in a reasonably-sized dataset with low trajectory density. The training ended with an evaluation process requiring the use of selectors and operations to solve a specific task with this dataset: selecting trajectories between two given airports. Following the tutorial, participants were able to explore their visualization and use all the capabilities of VRT to filter their datasets. Each session lasted on average 60 minutes; the

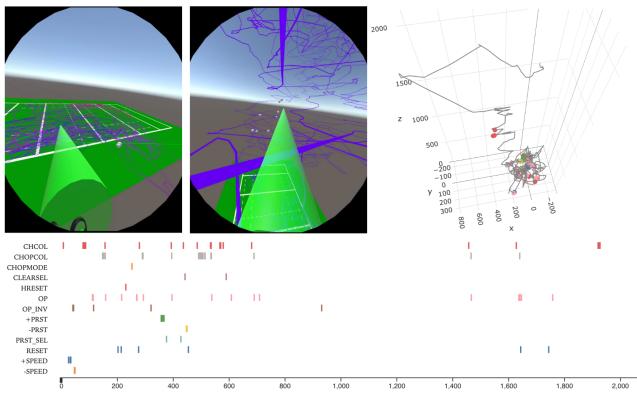


Figure 6: Case study RUGBY: the expert primarily used VRT to visualize data from a familiar point of view and used queries to single out interesting players. He also used the animation feature to playback players position over space (x, y), and their effort over time (z).

participants' behavior was recorded through a webcam, a screen-cast from the headset, and an automated log of their positions and interactions within the 3D environment. Those logs are illustrated in Fig. 6, 7, 8 and 9) with the position of the user (top-left 3D chart) and a timeline of its interactions (from top to bottom: choice of selector, creation of a persistent selector, operation on selectors, reset, change of speed and selectors edition mode activated/deactivated). Such logs enable to grasp the user activity over space and time. It also enabled us to efficiently browse video recordings to sequences of interest, e. g., when multiple persistent selectors have been created.

## 6.1 Case study RUGBY: Players Performance Analysis

Sports tracking data is becoming increasingly available due to advances in wearable GPS sensor technologies. Such data provides fine-grained position of players on a pitch. We collaborated with a Rugby Team (consistently ranked in the top 5 teams of one of the major European Leagues) which constantly collects GPS data of its players and substitutes during training and games (warm-ups included). We contacted a researcher expert in rugby, with experiences as a player and trainer. He was also familiar with the tactics of the team we collaborated with and mainly used dashboards, datasheets, and video streaming to identify weaknesses in the team, which zone of the pitch was occupied the most, as well as the team behavior during an action. Individual behaviors are also important to identify exhausted players needing to be substituted. We loaded the GPS dataset of a single game from the team in VRT. As the dataset was very accurate, we only loaded 1/10 of the dataset with uniform spatial sampling. We added 3D models of goalposts and a rugby pitch map to give spatial context and references to the expert.

The first step of our domain expert was to grasp the distribution of the dataset as a standard 2D map (Fig. 6 left). While this map generates an important visual clutter (due to over-plotting), the expert positioned himself in a tilted position similar to one of a spectator or a TV camera. He animated players using the playback feature and configured the paths color to reflect the players’ speed. The cylinder interaction was used to explore the dataset and pick a particular player (# 14). This player’s behavior was isolated, though with no particular findings. After resetting the selection, he inspected the team’s collective behavior and was able to infer the positions of the forward players by looking at group placements during waves of attack or defence. He constantly changed his position around the game to adjust his point of view. For this scenario, he did not find useful to select a spatial region in particular. The second step was to encode player load (which is an attribute related to effort) as the vertical axis (z) and to animate all the players. The resulting vertical

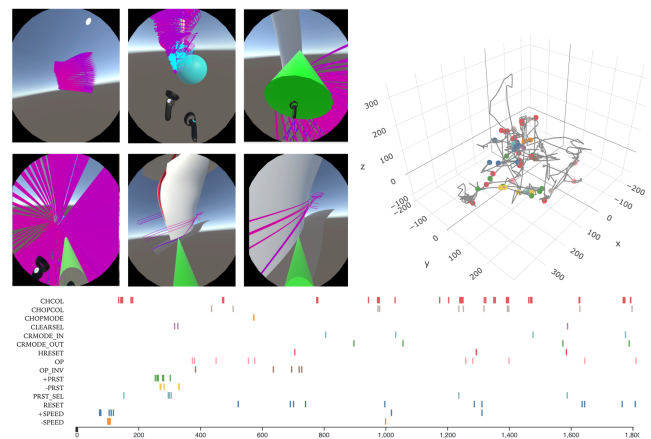


Figure 7: Case study PROPELLER: the expert managed to quickly select streamlines of interest around a propeller. He refined his selection to single out streamlines involved in the creation of the vortex and separated flow areas.

visualization allowed him to quickly identify the top 3 most active players on the field. Here, cylindrical selectors were often used as pointing rods to explain parts of the game or point to particular players without using their selecting ability.

## 6.2 Case study PROPELLER: Flow Simulation Exploration

Our colleagues from the Engineering department of our local university provided us with a numerical simulation of turbulence phenomena around a propeller over time. After discussions, we selected a data subset from a numerical simulation of 9,440 streamlines over a maximum of 150 instants, resulting in a total of 738,096 atoms. Those streamlines were localized around a propeller currently studied by these experts. Such a structure generates regions of interest such as a vortex and separated flow regions. They typically wanted to know if the streamlines involved in the vortex were from the same departure areas or not, and to study their time evolution. They also sought to identify supersonic regions in the flow. Experts knew interesting areas in advance and were seeking for a quick way to explore and visualize their dataset, as well as the possibility of communicating their research to either students or the general public in intuitive video format. They currently use Mathwork's Matlab<sup>6</sup> and Kitware's ParaView to run their simulation and visualize the results.

As streamlines are already by nature 3D data, their representation is straightforward. We tested two color strategies: coloring by Mach number to highlight the supersonic area, or using random colors to study the streamlines separately. 3D models of the propeller and of the supersonic layers completed the visualization.

The expert started his exploration with an overview of the trajectories and then selected streamlines from departure locations he knew would hold streamlines involved in vortex. Then, he changed his point of view to refine the selection using the cylinder shape (due also to a preference for what he called "a lightsaber"). By using another color, he could see the newly selected trajectories as a subset of previous selections. Fig. 7 illustrates this step-by-step process up to the final selection. The expert then spent some time explaining the physics foundations behind this phenomena using the cylinder to point to and describe the behavior of the selected streamlines.

<sup>6</sup><https://www.mathworks.com/products/matlab.html>

	Number of			Attributes used for				Context elements	
Case study	Trajectories	Atoms	Instants	X	Y	Z	Color	3D models	2D background
RUGBY	21	1,118,992	72,201	x	y	Speed Acceleration <b>Player load</b> Odometer Time	<b>Speed</b> Acceleration Player load Odometer Random	Goalposts	Rugby pitch
PROPELLER	9,440	738,096	150	x	y	z	<b>Mach number</b> Random	Propeller Supersonic layers	-
PARTICLES	10 <sup>6</sup>	2.5 · 10 <sup>9</sup>	2,500	x	y	z	Random Speed Acceleration <b>Power</b>	-	-
TRAFFIC	6,121	4,641,944	3,600	x	y	<b>Speed</b> Distance Time	Speed <b>Random</b>	-	City map

Table 2: Summary of the case studies by datasets properties, by count of unique trajectories, atoms (records) and instants (temporal time points) used for playback. Attributes are the ones available in the dataset for spatial mapping, for which dataset either already have a 3D structure (PROPELLER, PARTICLES) or mapping of a non-spatial attribute was used (RUGBY, TRAFFIC). We highlighted in **bold** the most interesting attributes during the case studies. Finally, some case studies required to add a 2D background (often a map) and a 3D model for context.

### 6.3 Case study PARTICLES: 3D Particles in a Turbulent Flow

Other colleagues from the Engineering department provided us with a simulation of 3D particles in an isotropic and turbulent flow. Due to the large number of particles, they face visualization and exploration problems with this dataset. Surprisingly, they had never been able to simultaneously visualize a correct amount of particles even though it is an important aspect of their research work. Indeed, they wanted to explore their turbulent flow to develop an intuition on data and results and confirm predictions on the evolution of attributes according to the shape of the trajectory. They currently use Mathwork's Matlab to visualize these particle trajectories, but this tool only allows them to extract and visualize some parts of the dataset. We met these colleagues to discuss their dataset and visualization problems. This allowed us to identify an interesting subset of data that we enriched by computing the speed, acceleration, and power attributes for all the particles. We ended up with a dataset describing the evolution of a million particles in a turbulent flow over 2,500 instants, which means 2,500,000,000 records in total.

The expert started exploring from within (Fig. 8, left) the particles' trajectories. The speed, acceleration, and power attributes were used to color the trajectories along with random colors for each particle depending on the focus of the expert. As VRT could not display all the points available in the dataset, we had to sample the dataset. A standard sampling example, used several times, was a visualization of 20,000 particles chosen randomly in the dataset between the instants 500 and 1,500. No context elements such as 3D models or 2D background were added to this visualization. The expert was primarily interested in identifying changes in the power received by the particle from the surrounding flow.

### 6.4 Case study TRAFFIC: Traffic Flow Simulation

Understanding flows and movements of vehicles in large cities is important to understand the mobility of the residents. This data can lead to reductions in contamination and improvements to the traffic and living conditions. However, such data is usually difficult to collect as not all cars share their coordinates in real time. We collaborated with a Research laboratory specialized in simulation traffic data, in particular with one of their latest dataset over a European city. It contained 6,121 trajectories of vehicles simulated over 3,600 instants (1 hour of traffic simulation), making a total of 6,641,944 records. Such a dataset enables following individual cars with a very high (simulated) spatial resolution. The experts were interested in identifying congestion areas and sources of traffic that

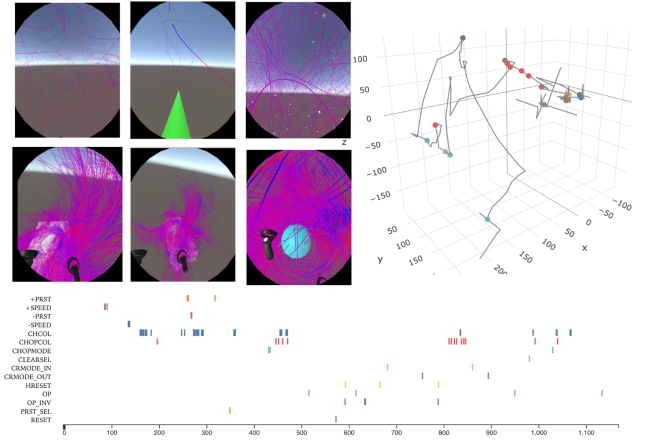


Figure 8: Case study PARTICLES: the expert studied the changes in power of the particles relative to the shape of the trajectories.

can create it. They also needed a visualization tool for their results to quickly validate the parameters of the simulation. They currently use Ifsttar's Symuvia<sup>7</sup>, a specialized simulation tool for traffic with 2D visualizations of the results to generate this dataset, as well as Mathworks' Matlab for all the post-processing and analysis of data.

The expert started exploring the dataset and first used a cylindrical, saber-like selector to extract the vehicles passing through a critical junction of the city and study their global trajectory (Fig. 9). The second step was to reset the selected trajectories and use persistent spheres, strategically placed on main junctions of a district of the city, to extract only these trajectories. He then used the sphere-drop functionality to only animate the vehicles on this selection and study the congested junctions. This simulation traffic dataset was intrinsically made of 2D trajectories on a plane, so we chose the speed of the vehicles as a third dimension. During the evaluation session, the coloration of the trajectories was either unique for each trajectory or representative of the speed of the vehicle. Despite the density of points, VRT was able to handle the full visualization and all the vehicle trajectories were displayed. We added the map of the city as a background to give spatial references to the expert.

<sup>7</sup><https://www.licit.ifsttar.fr/linstitut/cosys/laboratoires/licit-ifsttar/plateformes/symuvia/>



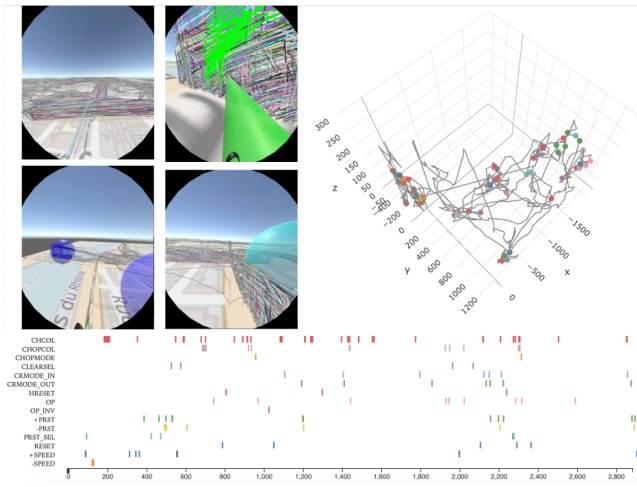


Figure 9: Case study TRAFFIC: the expert studied the vehicle trajectories likely to create congestion through the main critical junctions of the city.

## 7 DISCUSSION

We collected post-study feedback from users regarding the intuitiveness of the tool. Participants filled a questionnaire with a Likert scale ranging from 1 to 5 (with 1 corresponding to the less intuitive command mappings). The most intuitive commands were the movement in space using the Joystick, change of selector / color, and creation of Boolean operands. Change of speed, use of persistent selectors, and manual selection/deselection of queries were found quite intuitive. The reset command and creation mode were also found quite useful by participants, but one was not able either to use nor remember those commands. Performance was rated very good, but it could be improved to load larger datasets, mainly by refactoring the code to use a Data-Oriented Programming (DOP) approach to facilitate multi-threading, as well as moving collision detection computation to the graphics card.

The expert of the case study RUGBY raised some concerns about the additional value of VR in the exploration of the dataset. Indeed, most of the operations that he had done, could have been done in a desktop environment. Playback of the players could be achieved through video streaming of the game, and the identification of the most exhausted players through a dashboard or the use of datasheets. Nevertheless, the expert told us that VRT and VR might bring something new to the field if other attributes (e. g., heart rate) or automatic recognition of similar trajectory patterns were available. This would for instance allow for comparison of data between similar players, as well as a visualization of the players' effort during specific actions (e. g., rucks).

Further exchange with the experts allowed us to find two other flaws of VRT in its current state. First, even though VRT supports very early exploratory analysis, preliminary work is still needed to prepare the datasets. During our user evaluation, we pre-processed datasets to either filter or extract simulation samples as the entire dataset was too detailed. We also added domain-knowledge (e. g., 3D models, 2D background) to contextualize the exploration process. Those steps were purely technical and were achieved after multiple exchanges with domain experts. This was, however, partially addressed in later development stages, as the latest version of VRT facilitates this process by letting experts load their own trajectory datasets through a .json configuration file.

Secondly, exploration of 3D structures requires easy access to hidden, additional attributes, which VRT can for now only show after a reload of the entire visualization. During the case studies, an

operator assisted the user to switch z-axis mapping and get details on the datasets information (e. g., names of moving entities) as those features are inaccessible in real-time in VRT. Adding those features has a design implication that requires further research. Such features are needed to fully support analytical processes [5], along with additional standard interactions such as sorting, deriving, or aggregating values.

One of our main research perspectives is to investigate more diverse shapes for selectors, beyond the three basic shapes we implemented. According to [27], two directions could be explored: *custom* shapes and *data-dependent* shapes. Custom shapes operate on the geometry of the shape, independently from the data. CAD (Computer Aided Design) tools (e. g., Autocad<sup>8</sup>) already support building such shapes, along with operations such as extrusion (subtraction of a shape by another). Extrusion could be a promising operation to build custom shapes using the 3D models of the scenes, as we have seen those greatly influence selections. For instance, experts of the PROPELLER case study could create a contour of the propeller, by subtracting a cube from the propeller shape instead of adding small spheres together to re-construct this contour. Gestures could be used to create such shapes, and an extensive body of work is already investigated this area (e. g., [26]).

Data-dependent shapes are driven by the data distribution, so that the shape selects groups of data points optimally using the density of the surrounding region [28] or non-spatial attributes [16]. Combining shapes with data beyond selections has a very high potential. For example, remote selection of trajectories could be facilitated by encoding the density as a visual property of the shape. Similarly, density can be encoded in haptic feedback [20]) to access cluttered trajectories.

## 8 CONCLUSION

We presented VRT, a tool for exploring and filtering large trajectory-based datasets using virtual reality. It extends the design space of volumetric probes by introducing a set of configurable 3D shapes and logical operations for use in intuitive selection. We evaluated its use case with domain-experts who sought to quickly grasp an overview of their dataset, as well as to extract particular regions or trajectories of interest. Our results show the ease of use and expressiveness of the 3D geometric shapes implemented in VRT as selectors. VRT was found to be particularly useful for progressively refining selections from extensive trajectory datasets to small subsets of outlying behavior. It also demonstrated its use as a powerful tool for conveying the structure of normally abstract datasets for communication purposes.

## REFERENCES

- [1] D. Akers. CINCH: a cooperatively designed marking interface for 3d pathway selection. In *Proceedings of the 19th annual ACM symposium on User interface software and technology - UIST '06*, p. 33. ACM Press, Montreux, Switzerland, 2006. doi: 10.1145/1166253.1166260
- [2] F. Argelaguet and C. Andujar. A survey of 3d object selection techniques for virtual environments. *Computers & Graphics*, 37(3):121–136, May 2013. doi: 10.1016/j.cag.2012.12.003
- [3] H. Benko and S. Feiner. Balloon Selection: A Multi-Finger Technique for Accurate Low-Fatigue 3d Selection. In *2007 IEEE Symposium on 3D User Interfaces*, Mar. 2007. ISSN: null. doi: 10.1109/3DUI.2007.340778
- [4] L. Besançon, M. Sereno, L. Yu, M. Ammi, and T. Isenberg. Hybrid Touch/Tangible Spatial 3d Data Selection. *Computer Graphics Forum*, 38(3):553–567, June 2019. doi: 10.1111/cgf.13710
- [5] T. Chandler, M. Cordeil, T. Czauderna, T. Dwyer, J. Glowacki, C. Goncu, M. Klapperstueck, K. Klein, K. Marriott, F. Schreiber, and E. Wilson. Immersive Analytics. In *2015 Big Data Visual Analytics (BDVA)*, pp. 1–8, Sept. 2015. doi: 10.1109/BDVA.2015.7314296

<sup>8</sup><https://www.autodesk.com/products/autocad/overview>

- [6] B. Chen, J. Moreland, and J. Zhang. Human Brain Functional MRI and DTI Visualization With Virtual Reality. vol. ASME 2011 World Conference on Innovative Virtual Reality of *World Conference on Innovative Virtual Reality*, pp. 343–349. American Society of Mechanical Engineers Digital Collection, June 2011. doi: 10.1115/WINVR2011-5565
- [7] Z. Chen, W. Zeng, Z. Yang, L. Yu, C.-W. Fu, and H. Qu. LassoNet: Deep Lasso-Selection of 3d Point Clouds. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019. arXiv: 1907.13538. doi: 10.1109/TVCG.2019.2934332
- [8] D. Coffey, N. Malbraaten, T. B. Le, I. Borazjani, F. Sotiropoulos, A. G. Erdman, and D. F. Keefe. Interactive Slice WIM: Navigating and Interrogating Volume Data Sets Using a Multisurface, Multitouch VR Interface. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1614–1626, Oct. 2012. doi: 10.1109/TVCG.2011.283
- [9] M. Cordeil, A. Cunningham, B. Bach, C. Hurter, B. H. Thomas, K. Marriott, and T. Dwyer. IATK: An Immersive Analytics Toolkit. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 200–209, Mar. 2019. ISSN: 2642-5246. doi: 10.1109/VR.2019.8797978
- [10] M. Cordeil, A. Cunningham, T. Dwyer, B. H. Thomas, and K. Marriott. ImAxes: Immersive Axes As Embodied Affordances for Interactive Multivariate Data Visualisation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pp. 71–83. ACM, New York, NY, USA, 2017. event-place: Québec City, QC, Canada. doi: 10.1145/3126594.3126613
- [11] N. Elmqvist. BalloonProbe: Reducing occlusion in 3d using interactive space distortion. *VRST '05*, pp. 134–137. ACM, 2005. doi: 10.1145/1101616.1101643
- [12] N. Elmqvist and P. Tsigas. A Taxonomy of 3d Occlusion Management for Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095–1109, Sept. 2008. doi: 10.1109/TVCG.2008.59
- [13] J. Heer, M. Agrawala, and W. Willett. Generalized Selection via Interactive Query Relaxation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pp. 959–968. ACM, New York, NY, USA, 2008. event-place: Florence, Italy. doi: 10.1145/1357054.1357203
- [14] B. Hentschel, M. Wolter, and T. Kuhlen. Virtual Reality-Based Multi-View Visualization of Time-Dependent Simulation Data. In *2009 IEEE Virtual Reality Conference*, pp. 253–254. IEEE, Mar. 2009. doi: 10.1109/VR.2009.4811041
- [15] C. Hurter, N. H. Riche, S. M. Drucker, M. Cordeil, R. Alligier, and R. Vuillemot. FiberClay: Sculpting Three Dimensional Trajectories to Reveal Structural Insights. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):704–714, Jan. 2019. doi: 10.1109/TVCG.2018.2865191
- [16] B. Jackson, D. Coffey, and D. F. Keefe. *Force Brushes: Progressive Data-Driven Haptic Selection and Filtering for Multi-Variate Flow Visualizations*. The Eurographics Association, 2012. doi: 10.2312/PE/EuroVisShort/EuroVisShort2012/007-011
- [17] B. Jackson, T. Y. Lau, D. Schroeder, K. C. Toussaint, and D. F. Keefe. A Lightweight Tangible 3d Interface for Interactive Visualization of Thin Fiber Structures. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2802–2809, Dec. 2013. doi: 10.1109/TVCG.2013.121
- [18] R. Kopper, F. Bacim, and D. A. Bowman. Rapid and accurate 3d selection by progressive refinement. In *2011 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 67–74, Mar. 2011. ISSN: null. doi: 10.1109/3DUI.2011.5759219
- [19] T. Munzner. A nested model for visualization design and validation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):921–928, 2009.
- [20] A. Prouzeau, M. Cordeil, C. Robin, B. Ens, B. H. Thomas, and T. Dwyer. Scaptics and Highlight-Planes: Immersive Interaction Techniques for Finding Occluded Features in 3d Scatterplots. In *Conference on Human Factors in Computing Systems*, CHI '19, pp. 325:1–325:12. ACM, Glasgow, United Kingdom, May 2019. doi: 10.1145/3290605.3300555
- [21] A. Sherbondy, D. Akers, R. Mackenzie, R. Dougherty, and B. Wandell. Exploring connectivity of the brain’s white matter with dynamic queries. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):419–430, July 2005. doi: 10.1109/TVCG.2005.59
- [22] B. Shneiderman. Dynamic Queries for Visual Information seeking. *IEEE Softw.*, 11(6):70–77, Nov. 1994. doi: 10.1109/52.329404
- [23] J. Symanzik, D. Cook, B. D. Kohlmeier, and C. Cruz-Neira. Dynamic Statistical Graphics in the CAVE Virtual Reality Environment. In *Proc. Dynamic Statistical Graphics Workshop*, pp. 41–47, 1996.
- [24] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, Jan. 2018. doi: 10.1109/TVCG.2017.2743938
- [25] J. W. Tukey. Exploratory data analysis. *Reading, Ma*, 231:32, 1977.
- [26] Vinayak, S. Murugappan, H. Liu, and K. Ramani. Shape-It-Up: Hand gesture based creative expression of 3d shapes using intelligent generalized cylinders. *Computer-Aided Design*, 45(2):277–287, Feb. 2013. doi: 10.1016/j.cad.2012.10.011
- [27] G. J. Wills. Selection: 524,288 ways to say ”this is interesting”. In *Proceedings IEEE Symposium on Information Visualization '96*, pp. 54–60. IEEE, Oct. 1996. doi: 10.1109/INFVIS.1996.559216
- [28] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. Efficient Structure-Aware Selection Techniques for 3d Point Cloud Visualizations with 2dof Input. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2245–2254, Dec. 2012. doi: 10.1109/TVCG.2012.217
- [29] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg. CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3d Particle Clouds. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):886–895, Jan. 2016. doi: 10.1109/TVCG.2015.2467202