CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3D Particle Clouds

Lingyun Yu, Konstantinos Efstathiou, Petra Isenberg, and Tobias Isenberg, *Senior Member, IEEE*

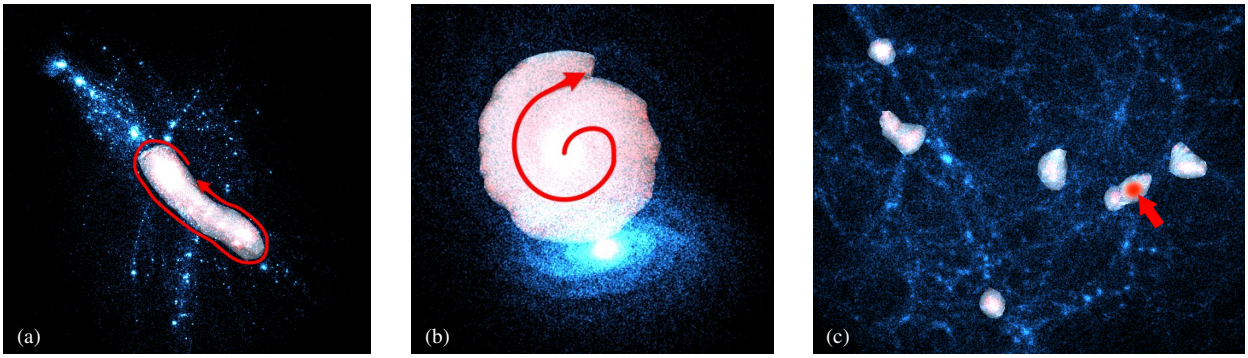

Fig. 1. (a) SpaceCast selects particle clusters by enclosing them with a lasso, based on the lasso shape; (b) TraceCast does not require an accurate lasso; and (c) with PointCast users can select tiny clusters from a noisy environment with only a single click or touch.

Abstract—We present a family of three interactive Context-Aware Selection Techniques (CAST) for the analysis of large 3D particle datasets. For these datasets, spatial selection is an essential prerequisite to many other analysis tasks. Traditionally, such interactive target selection has been particularly challenging when the data subsets of interest were implicitly defined in the form of complicated structures of thousands of particles. Our new techniques SpaceCast, TraceCast, and PointCast improve usability and speed of spatial selection in point clouds through novel context-aware algorithms. They are able to infer a user's subtle selection intention from gestural input, can deal with complex situations such as partially occluded point clusters or multiple cluster layers, and can all be fine-tuned after the selection interaction has been completed. Together, they provide an effective and efficient tool set for the fast exploratory analysis of large datasets. In addition to presenting Cast, we report on a formal user study that compares our new techniques not only to each other but also to existing state-of-the-art selection methods. Our results show that Cast family members are virtually always faster than existing methods without tradeoffs in accuracy. In addition, qualitative feedback shows that PointCast and TraceCast were strongly favored by our participants for intuitiveness and efficiency.

Index Terms—Selection, spatial selection, structure-aware selection, context-aware selection, exploratory data visualization and analysis, 3D interaction, user interaction

1 Introduction

Exploratory data visualization and analysis [**?**] is a fundamental pillar of many visualization systems. It provides domain experts with tools to study unknown datasets, an activity during which the data is examined more closely to discover interesting or unexpected patterns. One essential tool for exploratory visualization is the ability to select different subsets of the dataset [**?**], based on the current state of the exploration and/or one's own intuitions. Selection is relatively easy for 2D datasets by means of picking, lasso-selection, or brushing. Three-dimensional datasets such as particle simulations, however, pose the much harder challenge of needing to specify one or more 3D volumes that enclose the intended subset of the data. Most existing selection mechanisms for 3D data [**?**] provide means to pick or ray-point at objects, but these fail if no explicit 3D shapes to pick exist—as it is the case for particle data.

Recently, a number of structure-aware selection techniques [**?**, **?**, **?**] were proposed that promise to deduce a person's selection intention based on simple input, the characteristics of the dataset, and the current viewing conditions. All these techniques use a 2D lasso drawn on the projection of the 3D dataset and then derive a selection volume, typically by analyzing the particle density [**?**, **?**] or the scalar properties of volume data [**?**]. While these approaches present a considerable improvement to a cylinder-based lasso selection in 3D space, all of them require the users of a visualization system to first find a good view for the selection interaction—a task that is often difficult or impossible in realistic scenarios with complex datasets. Moreover, the existing structure-aware approaches do not take the location or shape of the drawn lasso into account when determining which part of the enclosed data space may represent the intended selection region.

In order to address these issues, we present SpaceCast, TraceCast, and PointCast, a family of new interactive context-aware selection techniques (CAST) (see Fig. 1). SpaceCast (Fig. 1(a)) analyzes the parts of the clusters that project inside the 2D lasso, selecting only the part that has the best overlap with the input lasso. TraceCast (Fig. 1(b)) is inspired by stroke recognition techniques, such as those used for the input of Chinese characters. In contrast to SpaceCast, TraceCast does not use the lasso metaphor but interprets the path of the drawn selection stroke, selecting that candidate cluster whose 2D projection is best approximated by the drawn stroke. In particular, TraceCast does not restrict selection to within the drawn stroke. Both SpaceCast and TraceCast are useful to select clearly defined particle clusters, even if they are occluded (partially, or even completely) by others. PointCast (Fig. 1(c)), finally, uses the picking/ray pointing metaphor known from

- *Lingyun Yu is with Hangzhou Dianzi University, Zhejiang, China. E-mail: mail@yulingyun.com.*
- *Konstantinos Efstathiou is with the University of Groningen, the Netherlands. E-mail: k.efstathiou@rug.nl.*
- *Petra Isenberg is with Inria, France. E-mail: petra.isenberg@inria.fr.*
- *Tobias Isenberg is with Inria, France. E-mail: tobias.isenberg@inria.fr.*

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication xx Aug. 2015; date of current version 25 Oct. 2015.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

traditional 3D selection. It evaluates potential candidate clusters along the line of sight, selecting the first one that matches a density threshold determined based on the densities along the ray. These three techniques thus suit a variety of different selection needs and, together, they form a powerful selection toolkit for analyzing particle datasets.

After a discussion of related work in Section 2, we introduce the three context-aware selection techniques in detail in Section 3. We then present the results of a controlled user study in Section 4 in which we evaluated the performance of the Cast selection techniques and compared them to the state-of-the-art selection methods. In Section 5 we then critically reflect on these results and what they mean for realistic application scenarios, show and assess a number of examples taken from scientific domains, and discuss limitations and future work, before concluding the paper in Section 6.

2 Related Work

Selection is a fundamental component of virtually all interactive visualization systems, "as ubiquitous as selecting icons in a desktop GUI" [**?**] and many variants exist as Wills [**?**] shows with his taxonomy of selection techniques. For the specific case of 3D representations or scenes, numerous methods have been created to select items and shapes [**?**, **?**]. For example, many approaches for selection by means of 3D raycasting have been created (e. g., [**?**, **?**, **?**, **?**, **?**]). Also special techniques, for example, for 3D streamline selection (e. g., [**?**, **?**, **?**]) and selection in different data spaces (e. g., [**?**]) have been created. In our work, however, we are less interested in distinct objects because, in the case of particle datasets, the single object (i. e., particle) typically has no meaning and 3D pointing techniques [**?**] would not be appropriate to select single particles—instead, groups of particles or particle clusters are important for a further analysis. This means we need selection techniques based on a *spatial specification* of a 3D sub-space that contains the elements to be further explored, with the challenge that this sub-space does not explicitly exist in the data.

Initial approaches for spatial selections used simple shape primitives that were easy to manipulate in 3D space such as cones [**?**, **?**]. Haan et al.'s [**?**] IntenSelect extended this concept for the selection of moving targets by combining it with a constantly updated scoring function. Particularly related to our work is Lucas and Bowman's [**?**] Tablet Freehand Lasso which also uses cone-based selection volumes but lets users draw them on a 2D projection of the data, which inspired our own interaction design. Of course, due to their reliance on simple selection shapes such as cones or cylinders, generic spatial techniques will be imprecise by default. Progressive/iterative refinement strategies thus facilitate the specification of better selections, such as demonstrated by Bacim et al. [**?**], Elmqvist et al. [**?**], and Kopper et al. [**?**]. We can make use of similar progressive refinement strategies by means of Boolean operations based on the selected subspaces [**?**].

Another form of deriving spatial selections is to use dedicated user interaction strategies. Straight-forward techniques use some sort of dedicated 3D input device (e. g., 3D mice, optical 3D tracking, LEAP motion, etc.) and also simple selection primitives. For example, Ulinski et al. [**?**] and Cabral et al. [**?**] position a cube resp. a sphere with two-handed input, while Bacim et al. [**?**] use the concept of half spaces controlled through 3D gestural input. Similarly, Ren et al. [**?**] and Burgess et al. [**?**] use freehand gestures to control the selection. Dedicated techniques exist also for true volumetric devices [**?**] and special input surfaces [**?**]. Benko and Feiner [**?**], in contrast, used touch input to cleverly position a selection sphere in space. This interaction design uses a planar 2D surface as input, similar to our own approach.

However, even such dedicated interaction designs typically require dedicated hardware and/or many steps to achieve an effective selection in complicated datasets such as the simulation of galaxies (e. g., [**?**, **?**]). Therefore, structure-aware or context-aware selection techniques have been devised that are able to deduce a user's selection intention based on the underlying data and view conditions and obtain a suitable or, at least, approximate selection in the first place. For two-dimensional problems, for example, the perceptual grouping of objects was successfully used for structure-aware selection [**?**, **?**]. In three-dimensional space, however, the situation is more complicated. Not only do we face another spatial dimension, but we also are typically limited to providing input in a 2D space (such as through mouse movements, pen input, or touch input). Unless one is only interested in a single location (e. g., voxel) [**?**] or surface patch [**?**] determined (i. e., picked) in a context-aware fashion, this task therefore requires the more difficult specification of a suitable 3D enclosing volume.

Owada et al.'s [**?**] Volume Catcher, for example, solves this problem for unsegmented volume data by asking users to draw 2D strokes along the perceived borders of features for unsegmented volume data, and then deduces an appropriate segmentation of the dataset, i. e., a selection. In an approach designed for line data, Akers' [**?**] CINCH provides a marking interface for structure-aware 3D neurologic pathway selection. Yu et al.'s [**?**] TeddySelection and CloudLasso allow users to enclose spatial regions with a lasso drawn on the 2D projection and derive an appropriate selection subspace for particle data. They demonstrate, in particular, that their CloudLasso is able to handle complex scenarios in less time than a simple cylinder-based selection. Shan et al. [**?**] then discuss an extension to CloudLasso, in that they analyze the different clusters that are generated by the CloudLasso technique and only select the one with the largest 2D projection.

These context/structure-aware selections thus all specify what is interesting by deriving a sub-space, doing that in a manner that depends on the data, the view, and other parameters [**?**]. While Wills [**?**] derives his taxonomy of selection techniques for visual encodings of abstract data on the 2D plane, his distinctions of selections with respect to *space differentiation* and *data dependency* also apply to our three-dimensional datasets as he also discusses selection as a way to spatially specify what to investigate further. The Cast family of selection techniques that we discuss in this paper are, hence, *data-dependent tools* that provide intuitive means to specify selection through a lasso, through a drawn stroke, or through a specified point. We thus address the challenges of the existing structure-aware selection techniques [**?**, **?**, **?**] which are only able to make global decisions for the entire dataset (or the lasso-enclosed part), while SpaceCast, TraceCast, and PointCast derive criteria for more local, precise, and intuitive selections by the shape and location of the drawn primitive.

3 Description of the Selection Techniques

To achieve these goals, the Cast family members have to derive the selection intention of the user from the type and shape of the provided input—in a way that goes beyond a simple two-dimensional cut-off mask as done by previous structure-aware approaches [**?**, **?**]. We thus employ three dedicated heuristics that use different ways for interactively constraining the selection volume. These heuristics do not directly use the particle positions but rather a continuous density field $\rho(\mathbf{r})$ that represents the particle density at a point $\mathbf{r}$ in space.[1] We first compute the density field at the nodes $\mathbf{r}^{(n)}$ of a regular 3D-grid that covers the dataset. We then derive the value $\rho(\mathbf{r})$ of the density field at any other point $\mathbf{r}$ in space through linear interpolation from the values of ρ at the grid nodes closest to $\mathbf{r}$. For completeness, we briefly review the computation of the density field ρ later in Sect. 3.4.

3.1 SpaceCast

Compared to the previous techniques such as CloudLasso, an obvious intention may be to constrain the selection such that only those clusters are selected that are similar in their (projected) shape as the drawn lasso. For this purpose we provide SpaceCast as the first method in the Cast family. SpaceCast's user interaction and initial form of computation thus resembles that of CloudLasso: the user also draws a lasso L around the particles to be selected. If the user-drawn input stroke is not closed we convert it into a closed stroke by connecting its start and end points. If the input stroke self-intersects we only consider its largest closed part, starting and ending at the intersection. The lasso L thus defines a frustum F, i. e., a 3D-volume that projects to the screen inside L. We

[1] This aspect also makes it possible to apply the techniques to volumetric data which samples a chosen scalar field. This scalar field does not have to be a density, but any visually salient aspect of the data when visualized.

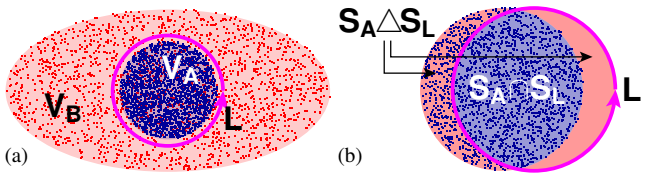

Fig. 2. (a) Small cluster in front of a larger one in SpaceCast. (b) Overlap measure m_A in TraceCast. The center area represents the common area $S_A \cap S_L$ that increases the measure while the two peripheral lobes represent the symmetric difference $S_A \triangle S_L$ that decreases the measure.

then restrict all subsequent operations to operate inside F. We compute an approximate average density ρ_F inside F as

$$\rho_F = \frac{1}{N_F} \sum_{n=1}^{N_F} \rho(\mathbf{r}^{(n)}),$$

where the sum runs over the set of grid-nodes $\mathbf{r}^{(n)}$ inside F and N_F is the number of such nodes. We then choose the initial density threshold as $\rho_0 = 0.2\rho_F$. Using the Marching Cubes algorithm [**?**, **?**], we identify the volume V (inside the frustum F) where the density ρ is above the threshold ρ_0. More precisely, we define the scalar quantity

$$f(\mathbf{r}) = \min\left\{ \frac{\rho(\mathbf{r}) - \rho_0}{\sigma_\rho}, \frac{\delta(\mathbf{r})}{D_L} \right\},$$

where $\delta(\mathbf{r})$ is the *signed* distance from the lasso of the projection $\mathbf{r}_s$ of $\mathbf{r}$ on the screen, that is, if we denote by $d(\mathbf{r}) \geq 0$ the distance of $\mathbf{r}_s$ from L then $\delta(\mathbf{r}) = d(\mathbf{r})$ if $\mathbf{r}_s$ is inside L and $\delta(\mathbf{r}) = -d(\mathbf{r})$ otherwise. To ensure that the physically disparate compared quantities have the same order of magnitude we scale distances by the lasso diameter D_L and densities by the density standard deviation σ_ρ. Volume V then results as the set for which $f(\mathbf{r}) \geq 0$.

In the previous step we automatically set the density threshold to ρ_0. To facilitate finer control of the selection volume we provide means to interactively adjust the threshold at runtime in the range $[\rho_0/16, 16\rho_0]$ by mapping a linear parameter $s \in [-4,4]$ to the threshold value using $\rho_s = 2^s\rho_0$. When s is adjusted, we thus recompute the scalar $f(\mathbf{r})$ for all grid nodes with ρ_0 replaced by ρ_s and obtain the iso-surface $f(\mathbf{r}) = 0$ using Marching Cubes. It is important to note that we only need to recompute $f(\mathbf{r})$ in this case because $\rho(\mathbf{r})$ and $\delta(\mathbf{r})$ remain constant, therefore adjusting the threshold is computationally much less expensive than the previous step and can be done interactively.

If V consists of a single connected component then we take this to be the selection volume and all particles inside V are marked as selected—SpaceCast would then work similar to CloudLasso. In general, however, V consists of more than one disjoint volumes $V_1, \ldots, V_K$. We identify these volumes using the algorithm described in Sect. 3.5.

In contrast to CloudLasso and WYSIWYG selection, however, we now use the shape of the drawn lasso to select one of the disjoint volumes. In particular, we project the volume V_k, for each $k = 1, \ldots, K$, to an area S_k on the screen. We then compare each area S_k to the area S_L enclosed by the lasso L and identify the area S_{k_M}, with $k_M \in \{1, \ldots, K\}$, that has the maximal overlap with S_L. We then determine the selection volume to be V_{k_M}, except in the following case. We choose the selection volume to be V_{k_N} if there is a volume V_{k_N} such that $S_{k_N} > 0.8S_{k_M}$ and V_{k_N} is closer to the eye compared to V_{k_M}, for the following reason. Let us suppose that the user tries to select a cluster V_A in front of a more extended cluster V_B as shown in Fig. 2(a). If the user then draws a lasso L encircling S_A we generally get $S_A < S_L$ (because L will not be perfectly wrapped around S_A) but $S_B = S_L$. Choosing the area with the largest overlap would then give V_B—although the intention was V_A. If more than one volume V_{k_N} exists that satisfies the condition $S_{k_N} > 0.8S_{k_M}$, we choose the one closest to the eye.

We note here that if the structure of the dataset and the density threshold are such that V consists of a single connected component then SpaceCast and CloudLasso give the same result. Furthermore, if the density threshold is set to a very low value then SpaceCast gives the same result as a simple cylinder-based selection.

3.2 TraceCast

A potential problem arises for SpaceCast in certain cases when a selection in complex dataset is required: SpaceCast requires the users to always outline the desired clusters fairly accurately and to draw closed (or almost closed) lasso loops. To ease such interaction we thus present TraceCast which, despite being only subtly different in its definition, can thus lead to significantly different interaction schemes as well as different selected sub-spaces.

With TraceCast the user draws a lasso-like stroke L around the particles to be selected as before. If the stroke is not closed, we close it as described in Sect. 3.1. However, unlike SpaceCast, we do not only consider the resulting frustum F but also take regions outside F into account for all subsequent operations. We first compute an approximate average density ρ_F inside F, as for SpaceCast. We then choose the initial density threshold as $\rho_0 = 0.2\rho_F$. Using the Marching Cubes algorithm, we identify the volume V in the whole space where the density ρ is above the threshold ρ_0. In particular, the volume V is where $f(\mathbf{r}) \geq 0$ but for this case we define the function f as

$$f(\mathbf{r}) = \rho(\mathbf{r}) - \rho_0.$$

whose threshold ρ_0 can interactively be modified as in SpaceCast.

We then consider the disjoint volumes $V_1, \ldots, V_K$ that form V and use the lasso shape to select one of these disjoint volumes. As in SpaceCast, we project the volume V_k, for each $k = 1, \ldots, K$, onto an area S_k on the screen. Note that in TraceCast the volume V_k can project (completely or partly) outside the area S_L that is enclosed by the lasso-like stroke L. We discard V_k if it projects completely outside S_L (that is, $S_k \cap S_L = \emptyset$). Otherwise, we compare the area S_k to the area S_L enclosed by the lasso L and assign it an *overlap measure* m_k, defined as

$$m_k = \text{area}(S_k \cap S_L) - \text{area}(S_k \triangle S_L) = 2\,\text{area}(S_k \cap S_L) - \text{area}(S_k \cup S_L),$$

where $\triangle$ represents the symmetric difference of two sets as shown in Fig. 2(b). We thus identify the area S_{k_M}, with $k_M \in \{1, \ldots, K\}$, that has the maximal overlap measure m_{k_M} and choose the selection volume to be the corresponding volume V_{k_M}. Our approach for TraceCast thus results in selecting that subset of space generally in the area of the lasso-like stroke that best approximates its shape.

Finally, we note that although TraceCast depends on the user-drawn lasso, the latter can be much less precise than in SpaceCast. This is clearly shown in Fig. 1(b) where a galaxy is selected using a "swirling" stroke. Using SpaceCast in the same dataset, we would need to draw a lasso that surrounds the target galaxy since SpaceCast does not consider any particle for the selection that falls outside the lasso.

3.3 PointCast

While SpaceCast and TraceCast are well-suited for larger clusters with often complicated structure, in practical applications we also frequently need to be able to quickly select many smaller structures—in a way like using ray-pointing to select dedicated objects in a 3D scene. We thus also introduce PointCast, a technique based on the concept of treating whole clusters as individual objects and using the simplest interaction method—a single click or touch action—to mark such a cluster.

With PointCast we thus start by projecting a ray into the dataset, guided by the marked point that is located on the interior of the target's projection to the screen. To be able to infer the intended selection volume, we start by identifying a density threshold as follows. We first sample the particle density ρ at equally spaced points along the ray, from the closest point $\mathbf{r}_0$ to the farthest point $\mathbf{r}_1$ that still lies inside the bounds of the dataset. Let

$$\rho(s) = \rho(\mathbf{r}_0 + s(\mathbf{r}_1 - \mathbf{r}_0)), \quad 0 \leq s \leq 1,$$

represent the particle density along the ray. By sampling $\rho(s)$ we then obtain the sequence $\rho_i = \rho(s_i)$ with $s_i = i/N_s$ for $i = 0, 1, \ldots, N_s$ where $N_s + 1$ is the number of sample points. We identify clusters along this ray by locating the intervals along the ray where $\rho(\mathbf{r}) \geq 0.1 \max\{\rho_i\}$. This produces a sequence of intervals $I_1 = [s_{a_1}, s_{b_1}]$, $I_2 = [s_{a_2}, s_{b_2}]$, etc., each of these intervals corresponding to a cluster.

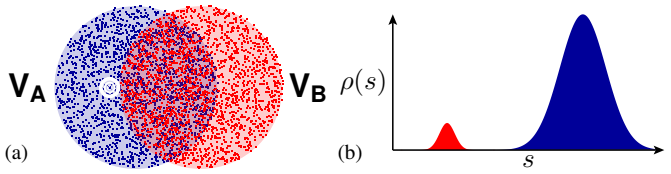

Fig. 3. (a) Near-hit in PointCast. The target cluster V_A is behind the cluster V_B. The selection ray passes nearby, but outside, V_B. (b) The graph of the density field along the selection ray. PointCast discards the spurious first increase in the density. The size of the colored areas represent the accumulated densities σ_A and σ_B.

With PointCast we usually assume that the desired selection volume is the one closest to the eye, i. e., the one corresponding to the interval I_1. Nevertheless, there are two cases where this may not be the correct choice. The first case is when a very faint cluster is in front of a more prominent one; we assume that in this case the user is interested in the prominent cluster behind. The second case is illustrated in Fig. 3. Here, the ray passes close to, but not through, a cluster V_A before reaching cluster V_B, the latter being the desired selection as shown in Fig. 3(a). The KDE method we use to obtain the density field ρ (see Sect. 3.4) smoothens out the density field and hence yields a thin density 'halo' around each cluster. We thus detect a small increase of the density $\rho(s)$ along the ray nearby V_A if the ray passes very close to V_A as shown in Fig. 3(b). Consequently, PointCast would pick the first cluster V_A along the ray—surprisingly for a user because this density 'halo' is invisible.

In order to account for these two cases we also consider the accumulated density in each interval I_k. In particular, we define the accumulated density for the interval I_k as

$$\sigma_k = \frac{1}{N_s}\sum_{i=a_k}^{b_k}\rho(s_i) \simeq \int_{s_{a_k}}^{s_{b_k}}\rho(s)\,ds$$

which is a measure of the total 'mass' contained in the interval. We then compare the accumulated densities σ_1 in the first interval I_1 and $\sigma_M = \max\{\sigma_k\}$ in the interval I_M with the maximal accumulated density. We finally select the cluster corresponding to the first interval I_1, *unless* $\sigma_1 < 0.1\sigma_M$ in which case we select the cluster corresponding to interval I_M. For the selected interval I_S we then compute the position $\mathbf{r}_S$ along the ray where the maximum ρ_S of ρ inside I_S is attained.

Finally, we set the density threshold to $\rho_0 = 0.2\rho_S$. Using the Marching Cubes algorithm we identify the volume V in the whole data space where the density ρ is above the threshold ρ_0. As with TraceCast, we derive the volume V using $f(\mathbf{r}) = \rho(\mathbf{r}) - \rho_0 \geq 0$. We then consider the disjoint volumes $V_1, \ldots, V_K$ making up V and use the previously described analysis to select one of these volumes. In particular, the final step in the method is to select the volume V_S that contains the point $\mathbf{r}_S$ identified earlier.

The threshold ρ_0 can be interactively modified as in the other two methods. This provides us with a powerful mechanism to trace connected structures: as ρ_0 is lowered, more volumes that were previously disjoint will merge, creating an ever-increasing connected component within the data that is seeded by the first cluster that was identified. We show examples for this interaction strategy later in Sect. 5.3.

3.4 Density Estimation

A common aspect of all methods in the Cast family is that they replace the particle position information by a continuous density function. We use the same continuous density, based on a standard Kernel Density Estimation (KDE) technique, as done by Yu et al. [**?**]. To aid comprehension, we briefly summarize its construction next. KDE methods obtain a smooth density function by 'smearing' each particle over a larger volume. We consider a rectangular box B that covers the dataset (or the region of interest if the latter is restricted) and a uniform grid with nodes at positions $\mathbf{r}^{(n)}$ where the superscript n identifies the node. For the estimation of the scalar density field we use the modified Breiman kernel density estimation method (MBE) with a finite-support adaptive Epanechnikov kernel [**?**, **?**]. This works as follows. First, for each direction $k = x, y, z$, we define the smoothing length

$$\ell_k = 2(P_k^{(80)} - P_k^{(20)})/\log N,$$

where N is the particle count in B and $P_j^{(q)}$ is coordinate k's q-th percentile value. Then, we derive the *pilot density* $\rho_{\text{pilot}}(\mathbf{r}^{(n)})$ at node n as

$$\rho_{\text{pilot}}(\mathbf{r}^{(n)}) = \frac{15}{8\pi N}\frac{1}{\ell_x\ell_y\ell_z}\sum_i E\big(\|\mathbf{r}^{(i;n)}\|\big),$$

where $\mathbf{r}^{(i;n)}$ is the vector with k-th component

$$\mathbf{r}_k^{(i;n)} = (\mathbf{r}_k^{(i)} - \mathbf{r}_k^{(n)})/\ell_k,$$

$\|\mathbf{r}^{(i;n)}\|$ its length, and $E(x) = 1 - x^2$ for $|x| \leq 1$ while $E(x) = 0$ for $|x| \geq 1$. The specific form of the Epanechnikov kernel, $E(x)$, implies that only particles for which the node n is inside an ellipsoid with semi-axes ℓ_x, ℓ_y, ℓ_z centered at the particle contribute to $\rho_{\text{pilot}}(\mathbf{r}^{(n)})$.

The pilot density $\rho_{\text{pilot}}(\mathbf{r}^{(i)})$ at the position of the i-th particle is then given by multi-linear interpolation with respect to the nearby nodes. For each particle i and with $k = x, y, z$ we then define the smoothing lengths $\ell_k^{(i)}$ which are unique to each particle as

$$\ell_k^{(i)} = \min\{\ell_k(m/\rho_{\text{pilot}}(\mathbf{r}^{(i)}))^{1/3}, 10\,s_k\},$$

where m is the arithmetic mean of $\rho_{\text{pilot}}(\mathbf{r}^{(i)})$ over all particles in B, and s_k is the distance between adjacent grid points in the k-th direction. Now, we can re-define the vectors $\mathbf{r}^{(i;n)}$ as

$$\mathbf{r}_k^{(i;n)} = (\mathbf{r}_k^{(i)} - \mathbf{r}_k^{(n)})/\ell_k^{(i)}.$$

The density $\rho(\mathbf{r}^{(n)})$ at the node n is then

$$\rho(\mathbf{r}^{(n)}) = \frac{15}{8\pi N}\sum_i \frac{1}{\ell_x^{(i)}\ell_y^{(i)}\ell_z^{(i)}} E\big(\|\mathbf{r}^{(i;n)}\|\big).$$

Finally, we compute the density $\rho(\mathbf{r})$ at an arbitrary position using multi-linear interpolation with respect to the nearby nodes.

3.5 Identification of Disjoint Volumes

A common step in the Cast methods is to identify the disjoint volumes that constitute the volume as computed by the Marching Cubes isosurface extraction. We do this using an algorithm similar to that of Shan et al.'s [**?**] WYSIWYG selection. In contrast to Shan et al.'s approach, however, we identify disjoint volumes by first identifying the connected components of the Marching Cubes isosurface. This leads to a more robust detection that does not have any problems with nearby but disjoint selection volumes that could be wrongly identified as part of the same component. Specifically, we first use a disjoint-set data structure to compute equivalence classes for the faces that make up the Marching Cube isosurface, where we consider two faces to be equivalent if they share an edge. Each such equivalence class represents a connected component of the isosurface. This task is significantly simplified by the fact that we keep track of the relations between vertexes, edges, and faces of the isosurface. We first assign an index $k \in \{1, \ldots, K\}$ to each equivalence class, and thus to each connected isosurface component. We then consider the grid-nodes $\mathbf{r}^{(n)}$. To each of these grid-nodes that lies inside the isosurface (i. e., for which $f(\mathbf{r}^{(n)}) \geq 0$) we assign the index of the connected isosurface component using a flood-filling algorithm. We start with a grid-cell that contains a face with a given index k. Then we find a node in this cell that is contained within the isosurface component with index k. We assign the index k to the found node and then we proceed to neighboring nodes assigning the same index k if they can be reached without crossing the isosurface. We thus identify all grid-nodes with index I. By repeating this process for all indices, we finally mark each node with an index $k \in \{1, \ldots, K\}$ if it lies inside the Marching Cubes isosurface. For completeness, we assign the index $k = 0$ to nodes outside the isosurface.

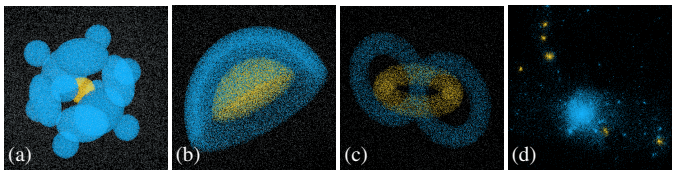

Fig. 4. Four datasets: (a) multiple particle clusters, (b) semispherical shell with half-ball inside, (c) three rings, and (d) an N-body mass simulation; named *clusters*, *shell*, *rings*, and *simulation* in the study description.

4 User Study

In order to understand user performance with our new selection methods in different situations we conducted a comparative quantitative user study. We compared the Cast methods with the traditional selection method based on generalized cylinders and with the structure-aware method CloudLasso, according to time and accuracy of the selection.

4.1 Study Description

Participants Twenty people (14 male, 6 female) participated in the study. 15 participants were students from different disciplines and five non-students. Eight of them had at least a Bachelor's degree. 16 participants reported prior experience with 3D computer games with playing games up to two times per day, with ten participants reporting at least weekly experience. Ages ranged from 18 to 35 years ($M = 23.4, SD = 4.8$). All participants reported to be right-handed.

Apparatus The experiment was performed on a Microsoft Surface Pro 2 (1280×800 pixels). The surface recognizes touch as well as a pen input. While participants had the freedom to use either direct touch or the pen as their input method, all decided to use the pen input.

Datasets Our study comprised four datasets, as illustrated in Fig. 4. The datasets contained target particles (orange), interfering particles (blue), and noise particles (light blue). These datasets were designed to have different features that made selection of targets challenging:

Clusters: The clusters dataset in Fig. 4(a) contained 15 compact clusters of particles with equal uniform densities inside a low density noise environment. The target cluster was located in the center so that no viewing direction allowed a clear view to the whole target.

Shell: The shell datasets in Fig. 4(b) included a half-ball of target particles partially surrounded by a semispherical shell of interfering particles. Both structures had the same uniform density.

Rings: The rings dataset (Fig. 4(c)) contained a circular 'ring' intertwined with a 'figure-8' (a 'double ring'), inside a low-density noise environment. Each ring had the same uniform density and participants were asked to select the circular one. In this dataset the structure of the target intertwined with other structures, making selection challenging.

Simulation: Fig. 4(d) shows a cosmological N-Body simulation dataset with a high density core in the center and many small and high density clusters all around the space. Participants were required to select several small clusters. This real-world dataset was chosen to represent a realistic selection scenario in astronomy. Considering that the high density core is relatively easy to select with any technique, we chose small clusters as the target to see how our selection strategies worked when multiple selections had to be performed.

Task and Procedure Participants were always asked to select all orange target particles and avoid selecting interfering or noise particles. Before the actual experiment, participants practiced with three additional training datasets to get accustomed to the selection techniques. The practice datasets were chosen to explain the differences between techniques and the interaction with the experimental software. These practice datasets included: 1) a simulation of two colliding galaxies (for practicing trackball rotation), 2) five randomly placed compact clusters of particles with equal uniform densities inside of a low-density noise environment, with one being set as the target cluster (for explaining how the method works and the differences compared with other methods), and 3) a higher-density volume of orange target particles with two simple geometric shapes: a pyramid and a torus (for explaining the method for different selection target shapes). In the training trials, we gave the participants as much time as they wanted, before proceeding to the actual experiment. Training took on average 14 mins for CylinderSelection, 8 mins for CloudLasso, 7 mins for SpaceCast, and 6 minutes each for TraceCast and PointCast.

We asked participants to perform their selections as quickly and accurately as possible but we did not tell them when they had accomplished the selection goal. For each trial-dataset combination we chose a unique dataset starting orientation, that changed between trials but was kept the same for different participants. We also counter-balanced the order of selection methods presented to the participants such that each possible starting-ending pair of methods was used exactly once and that no participant saw the same progression order. To avoid measuring unnecessary time spent on navigation, we only provided trackball rotation. A selection was activated after participants pressed a 'selection button' to change to selection mode, otherwise the 2DOF input on the data was used for rotation. We provided three possible selection modes, corresponding to three Boolean operations: union ($+$), intersection ($\cap$), and subtraction ($-$). We also provided a slider (Fig. 14) through which participants could adjust the automatically determined density threshold even after rotating the view. As Yu et al. [**?**] had previously discussed, subtraction and intersection are best done using CylinderSelection and not using structure-aware techniques. We thus implemented subtraction and set intersection as CylinderSelection in all trials.

We allowed participants to undo/redo the five most recent operations. Once they felt that they accomplished the selection goal or that they were not able to improve the result, they could press a finish button to advance to the next trial.

Design We used a repeated-measures design with the within-subject independent variables *selection technique* (CylinderSelection, CloudLasso, SpaceCast, TraceCast, and PointCast) and *dataset* (Clusters, Shell, Rings, Simulation). Each technique was used for each dataset in 3 repetitions. In summary, the design consisted of 20 participants × 5 methods × 4 datasets × 3 repetitions = 1200 trials in total.

Measures and Analysis After an inspection of timing results, we removed the first repetition of each dataset × technique combination in order to reduce the influence of learning effects. This left us with 40 trials per participant (800 trials in total). Traditionally, such study data would have been analyzed using null-hypothesis significance testing (NHST). The use of NHST, however, has been increasingly criticized as a tool to analyze user study experiments [**?**, **?**, **?**], also discussed in the context of visualization [**?**, **?**]. Consistent with recent APA recommendations [**?**], we thus report results using estimation techniques with effect sizes and confidence intervals rather than p-value statistics.

To compare the five techniques, we measured task completion time and computed two different accuracy scores, the F1 and MCC scores, similar to Yu et al.'s work [**?**]. Both accuracy scores are based on the identification of true positives (TP, number of correctly selected particles), false positives (FP, number of incorrectly selected particles), and false negatives (FN, number of target particles that were not selected). The F1 score is a weighted average of precision $P = TP/(TP+FP)$ and recall $R = TP/(TP+FN)$ and is often used in information retrieval to assess query classification. F1 is calculated as: $\mathrm{F1} = 2 \cdot (P \cdot R)/(P+R)$. The second accuracy measure, the Matthews correlation coefficient (MCC), additionally uses the number of true negative particles (TN, non-target particles that were not selected) to calculate an accuracy score. It is commonly used in machine learning for the assessment of binary classifiers. MCC is calculated as:

$$\mathrm{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}.$$

Both scores normalize accuracy: for F1, 1 indicates a perfect performance and 0 represents the worst possible result, while the corresponding values for MCC are 1 and -1. For accuracy scores we computed means and 95% bootstrap confidence intervals (all $n = 20$) [**?**].

Table 1. Mean task completion times, accuracy scores, and their 95% confidence intervals per technique.

Technique	Time	CI	F1	CI	MCC	CI
Cylinder	88s	[73,106]	.96	[.95,.97]	.96	[.95,.96]
CloudLasso	43s	[38,48]	.96	[.96,.97]	.96	[.95,.97]
SpaceCast	23s	[20,27]	.97	[.94,.97]	.96	[.93,.97]
PointCast	16s	[13,19]	.97	[.93,.98]	.97	[.93,.98]
TraceCast	18s	[16,20]	.97	[.97,.98]	.97	[.97,.98]

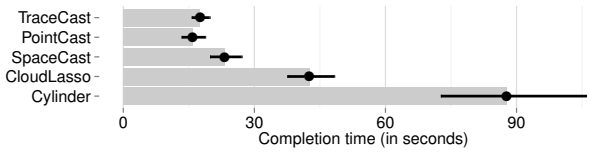

Fig. 5. Mean completion time (in seconds) across all participants for each selection technique. Error bars show 95% confidence intervals. Means are geometric means.

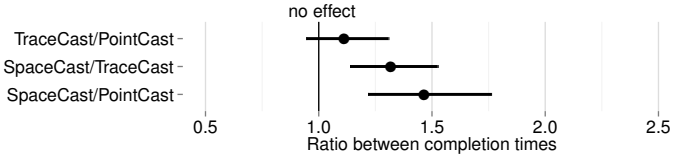

Fig. 6. Ratios between mean completion times for the Cast selection techniques. Error bars show 95% confidence intervals.

We analyzed completion time data using exact confidence intervals on log-transformed data (all $n = 20$) making outlier removal unnecessary [**?**]. For completion time, all reported means are thus geometric means, and comparisons between means are expressed as ratios [**?**].

4.2 Results

Here we report results of both the quantitative analysis as described above, the results of the post-session questionnaire, as well as a few observations from watching participants complete the study.

4.2.1 Overall Results: Time, Accuracy, and Preference

Fig. 5 shows the mean completion times and 95% confidence intervals for each technique. We can see that CylinderSelection and CloudLasso were much slower than the three Cast techniques. CylinderSelection was also much slower than CloudLasso, which is consistent with the results of Yu et al.'s [**?**] experiment. Exact average task completion times as well as confidence interval ranges are listed in Table 1. Given the high completion time difference for CylinderSelection, Fig. 5 is less clear as to the difference amongst the Cast techniques. Therefore, we provide pairwise ratios of completion times of the Cast techniques in Fig. 6. Here we can see that SpaceCast took about 1.3–2.2 times longer than PointCast on average and 1.1–1.8 times longer than TraceCast. Participants performed only about 1–1.4 times slower with TraceCast than PointCast. Thus, overall we have good evidence that both PointCast and TraceCast outperform SpaceCast, and some indication that PointCast may outperform TraceCast. At any rate, the differences among Cast methods are marginal compared to the differences between each Cast method and CloudLasso or CylinderSelection.

At the beginning of the study (before the trials began), we had asked participants how they would intuitively approach a selection of 3D particle data. More than 80% of participants had then reported that drawing a lasso would feel most natural. In the post-session questionnaire, however, 13 participants reported to favor the point-based method PointCast. Out of the remaining participants, six preferred the lasso-based method TraceCast and one participant preferred both PointCast and TraceCast over the other techniques. This is interesting also in the context of the actual results. The analysis of accuracy scores (refer to Fig. 5 for exact numbers) for the five techniques shows that

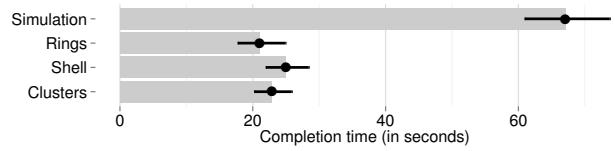

Fig. 7. Mean completion time (in seconds) across all participants for each dataset. Error bars show 95% confidence intervals.

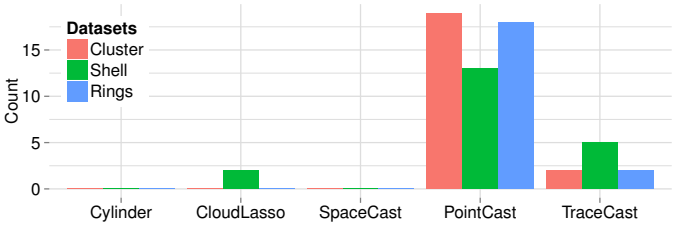

Fig. 8. Preferred techniques for each of the synthetic datasets.

participants were very accurate with all techniques. All F1 and MCC scores are $\geq .96$ and, therefore, close to a perfect result (1).

How often participants adjusted the slider depended on technique and dataset. Of the Cast techniques, the slider was on average adjusted most frequently for the PointCast technique. For all techniques most adjustments were necessary for the simulation dataset: 3.7 for SpaceCast, 4.5 for TraceCast, 6.25 for PointCast. All other datasets saw two or less adjustments on average per technique.

4.2.2 Analysis Per Dataset

Overall, participants completed all trials with similar completion times in just under 30s on average for the three synthetic selection problems Rings, Shell, and Cluster (see Fig. 7). Trials of the simulation dataset, on the other hand, were completed in over one minute on average, so the selection problem was more complex. Next we discuss the results for the three synthetic datasets and the real-world dataset separately.

Synthetic Datasets As can be seen in Fig. 9(a-c), the ranking of techniques according to task completion times for each of the datasets is similar to the overall ranking result. The three Cast techniques outperform CloudLasso and Cylinder for each of these three datasets. Amongst the Cast techniques, SpaceCast was generally the slowest with TraceCast and PointCast showing similar averages. Table 2 shows that the accuracy scores were also similarly high for each of the selection problems, when examined by technique.

In the post-session questionnaire we had asked the participants to choose the method they preferred for each dataset. PointCast was the preferred technique for all datasets, with exact numbers reported in Fig. 8. Participants preferred PointCast for the clusters datasets because it was fast (6×), accurate (9×), and because it was easier to select occluded particles (10×). Similar reasons were given also for the other datasets, with slightly different frequencies. TraceCast was the second most preferred technique across all datasets. Reasons for choosing TraceCast were that it offered a feeling of control and more accuracy, in particular for the Shell dataset where it was possible for PointCast to select unwanted clusters given certain viewing directions.

N-body mass simulation In this dataset we tested a multi-step selection process by asking participants to select several small clusters that were located in different places in space. Participants thus needed to rotate the whole space in order to find the targets. This explains the increased task completion time for this dataset (see Fig. 7) when compared to the three synthetic datasets. Interestingly the ranking of techniques according to task completion time changed for this dataset. Cylinder was still by far the worst technique for this task. The performance of CloudLasso, however, was much closer to the Cast techniques than for the other datasets (see Fig. 9(d)). In Fig. 10 we can see evidence that participants performed almost equally well with CloudLasso and SpaceCast as well as with PointCast. The highest ratio difference between techniques was found between TraceCast and SpaceCast, with

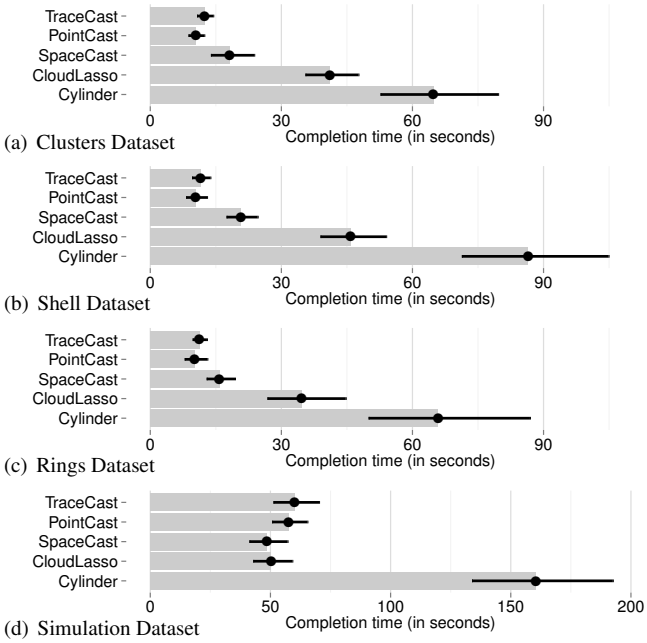

Fig. 9. Mean completion time (in seconds) across all participants for each selection technique and each dataset. Error bars show 95% confidence intervals. Means are geometric means.

Table 2. Mean task completion times, accuracy scores, and their 95% confidence intervals per technique and dataset.

Technique	Time	CI	F1	CI	MCC	CI
Clusters Dataset						
Cylinder	65s	[53,80]	.99	[.98,.99]	.99	[.98,.99]
CloudLasso	41s	[36,48]	.99	[.98,.99]	.99	[.98,.99]
SpaceCast	18s	[14,24]	.99	[.98,.99]	.99	[.98,.99]
PointCast	10s	[8.8,12]	.99	[.95,.99]	.99	[.96,.99]
TraceCast	12s	[11,14]	.99	[.98,1]	.99	[.98,1]
Shell Dataset						
Cylinder	86s	[71,105]	.97	[.97,.97]	.96	[.96,.97]
CloudLasso	46s	[39,54]	.95	[.92,.96]	.94	[.91,.96]
SpaceCast	21s	[17,25]	.95	[.86,.98]	.94	[.82,.97]
PointCast	10s	[8.2,13]	.97	[.93,.98]	.97	[.93,.98]
TraceCast	11s	[9.6,14]	.98	[.97,.98]	.97	[.97,.98]
Rings Dataset						
Cylinder	66s	[55,87]	.96	[.96,.97]	.96	[.94,.96]
CloudLasso	35s	[27,45]	.98	[.97,.98]	.97	[.96,.98]
SpaceCast	16s	[13,19]	.98	[.97,.98]	.97	[.96,.98]
PointCast	10s	[7.9,13]	.97	[.91,.99]	.97	[.91,.98]
TraceCast	11s	[9.7,13]	.98	[.97,.99]	.98	[.96,.98]
Simulation Dataset						
Cylinder	161s	[134,192]	.92	[.91,.93]	.92	[.91,.93]
CloudLasso	50s	[43,59]	.94	[.93,.95]	.94	[.93,.95]
SpaceCast	49s	[41,57]	.94	[.93,.95]	.94	[.93,.95]
PointCast	58s	[51,65]	.94	[.89,.96]	.95	[.90,.96]
TraceCast	60s	[51,70]	.95	[.93,.96]	.95	[.93,.96]

TraceCast performing 1.2× slower than SpaceCast. In summary, this comparison provides us with some evidence that SpaceCast performed slightly better than the other Cast techniques, but also that participants were almost equally fast with CloudLasso. Similar to the other datasets, however, accuracy was high for all techniques as we show in Table 2.

In the post-session questionnaire, nine participants reported that they preferred PointCast for this dataset because it helped them directly point out the particle clusters as 3D objects. Some of them 'complained' that

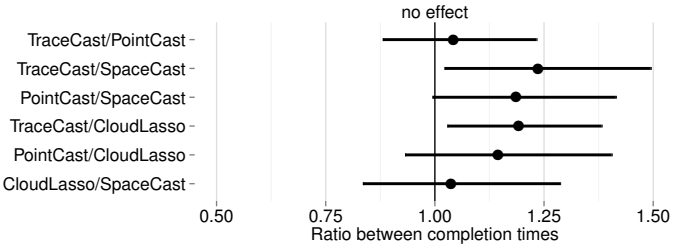

Fig. 10. Ratios between completion times for the simulation dataset. Error bars show 95% confidence intervals.

it would have been even better if they could have continuously picked clusters, without switching back to the rotation mode.[2] However, ten participants reported that they preferred a lasso-based selection method because the lasso gave them a better sense of control by allowing them to enclose all wanted particles with a lasso. This was especially true with CloudLasso, when all 3D target clusters were selected after participants added a 2D lasso and adjusted the threshold. The remaining participant reported that he liked all Cast methods.

5 Discussion

Based on these results and selection strategies, we now compare the evaluated selection techniques. We not only discuss the relative strengths and weaknesses of each method, but also contemplate the users' selection strategies in different situations.

5.1 Conceptual Foundation

Our new Cast techniques are all spatial, structure-aware selection techniques, a property they share with CloudLasso. However, we significantly improve upon CloudLasso as demonstrated by the user study. The fundamentally new idea put forward by the Cast methods is that the specific user interactions used to make the selection already contain rich information about a user's intention, and consequently we take advantage of this information for the Cast techniques. Even without a precisely draw lasso we thus enable users to obtain accurate and fast results, based on our heuristics that guess a user's intention. Ultimately, it is therefore the interacting user who decides which method to employ, based on the given dataset and exploration situation.

SpaceCast is conceptually closest to CloudLasso. Nevertheless, it improves upon it by only selecting a single component of the set of subspaces indicated by the lasso, facilitating different selection strategies: While CloudLasso can be used for multiple connected or non-connected targets in one step, SpaceCast allows users to specify a single intended target cluster though the shape of the drawn lasso.

TraceCast and PointCast deviate conceptually from SpaceCast and CloudLasso in that they no longer restrict the domain where a selection is made within some stroke or other input, thus no longer requiring an actual lasso to be drawn. Moreover, and more importantly, the conceptual basis of these two methods is that they treat clusters as distinct objects and no longer just as an arbitrary collection of particles. This new concept facilitates more freedom and flexibility in designing selection heuristics, either as simple click or touch operations with PointCast or using a shape recognition approach as in TraceCast.

5.2 Selection Scenarios

To better illustrate the different characteristics of the new selection methods we now discuss them with respect to changing selection scenarios: the selection of multiple clusters, the selection of a part of a cluster, and the selection of partially or completely occluded structures.

Multiple clusters SpaceCast and TraceCast are based on a user-drawn 2D stroke, while PointCast requires pointing at the target. Point-based methods such as PointCast are usually faster for selecting single targets because they do not require any drawing. The trade-off lies in the

[2]This issue is due to the interaction design with the mouse/pen-based input—for the future we envision to either provide a dedicated selection mode for such input modalities or to use bi-manual control for touch-based interaction.

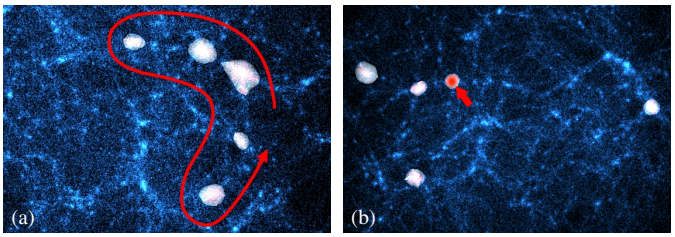

Fig. 11. CloudLasso (a) is able to get several disconnected parts with an interactively adjusted density threshold. PointCast (b) requires individual operation for each cluster.

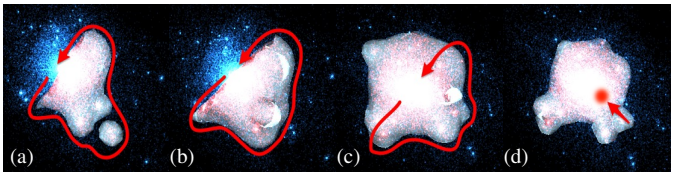

Fig. 12. CloudLasso (a) and SpaceCast (b) can select part of a cluster. TraceCast (c) and PointCast (d) treats clusters as individual objects. The four selections in each example are made from roughly the same viewpoint with respect to the particles.

resulting accuracy and needed fine-tuning since the 2D stroke used in SpaceCast and TraceCast provides more information about the likely user intention. Theoretically, all selection methods are able to select multiple clusters if there is no time limitation or restrictions on the number of Boolean operations. CloudLasso, in fact, supports selection of several clusters in one step, although users typically need to adjust the density threshold to arrive at the desired results (Fig. 11(a)). In the user study, however, we observed that participants tended to select clusters one-by-one and use Boolean operations, where necessary, to adjust their selection. Our new methods SpaceCast, TraceCast, and PointCast thus only select connected volumes and, therefore, require (at least) N operations for the selection of N clusters. The PointCast method is consequently particularly efficient for such selections of a few small clusters (Fig. 11(b)), while SpaceCast and TraceCast facilitate the selection of few more complex clusters.

Partial selection Both TraceCast and PointCast are based on the concept of treating clusters as individual objects, and they do not facilitate the direct selection of a subset of a cluster (although the latter is always possible through Boolean operations). SpaceCast (just as CloudLasso), in contrast, does facilitate partial cluster selection (Fig. 12) since the selection is always restricted within the drawn lasso.

Occlusion All Cast methods support the dedicated selection of partially occluded targets (Fig. 13)—in contrast to previous structure-aware selection methods. The impact of this property is reflected in the results of the post-session questionnaire, in which our participants rated all studied techniques on selecting in a noisy environment with occluded targets on a 7-point Likert scale. Cast methods were rated highly with an average of 6 (*agree*), CloudLasso was rated at an average of 4.5 (between *no opinion* and *somewhat agree*) and CylinderSelection received an average rating of 2.7 (between *disagree* and *somewhat disagree*). To achieve the selection of partially occluded clusters, SpaceCast requires users to trace the border of the target. PointCast, in contrast, only needs a small part of the target to not be occluded in order to facilitate picking it out from other clusters. TraceCast, finally, was regarded as the most effective method with respect to this point by the study participants. The reason was that, with TraceCast, the right cluster would be selected as long as its shape was closer to the input lasso than compared to that of other clusters—even if the target cluster was completely occluded.

5.3 Uncovering Global Structures

The selection volume created by PointCast selection depends to a large degree on the choice of the chosen density threshold, and at the same time is in no way restricted by a lasso. This property allows us to use

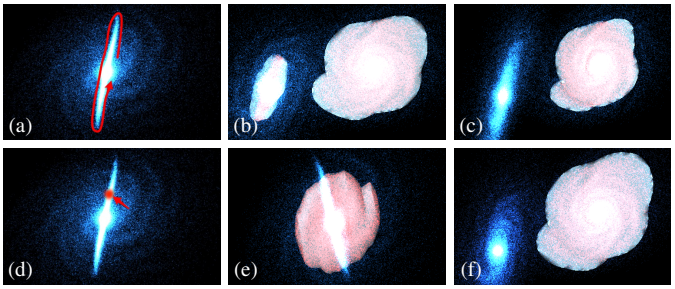

Fig. 13. Hidden structures: (a) lasso drawn around the target. CloudLasso (b) selects two disconnected components. TraceCast (c) selects the one whose 2D projection best matches the lasso, in this case the hidden one (both selection results viewed from roughly the same direction). PointCast (d) also facilitates the selection of partially hidden clusters: in (e) the front cluster is selected if the user points to the center, in (f) the partially hidden cluster is found if the user points to a suitable place.

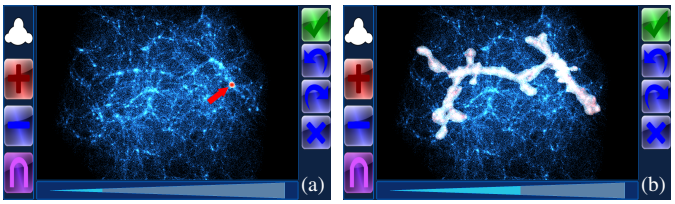

Fig. 14. Uncovering global structures in halos. (a) A small cluster is selected with PointCast. (b) Lowering the threshold reveals structures connecting the initially selected cluster to other clusters in the dataset.

PointCast to analyze how different parts of the dataset are connected with each other. In Fig. 14(a) we show an initial selection produced by PointCast using the automatically deduced density threshold, using a sample of the Millennium-II dataset [**?**] which represents galaxy halos. For the depicted selection case, the dataset contains more clusters that lie above the density threshold but which initially are not selected since they are not connected to the cluster under the clicked point. In Fig. 14(b) we show the situation when the density threshold has been lowered. Now, elongated structures have appeared that connect the previously disjoint clusters and thus produce a single connected volume that is now selected. This property of PointCast assists users in finding density concentrations in the dataset and helps them to understand how these are connected to each other on a more global level. The stroke based methods (SpaceCast, TraceCast, and the older CloudLasso) can produce similar structures, albeit with certain restrictions. Especially SpaceCast and CloudLasso can only reveal such connecting structures within the drawn lasso. TraceCast does not share this restriction, thus it can produce results similar to PointCast. Nevertheless, changing the density threshold can lead to the shape of the selection volume significantly diverging from the shape of the drawn lasso or stroke which may initially be surprising for some users.

5.4 Other Applications

The Cast methods were designed for selection in 3D particle datasets in general. While most examples we show in this paper come from astronomy, our methods can also be applied for selection in 3D particle datasets of different origin. Such datasets do not even need to have a spatial character but they could be based on other properties that can be mapped to 3D space. Such a representation can be generated from a multi-dimensional dataset by either choosing exactly three of these dimensions or by first performing a Principal Component Analysis and selecting the first three components. To show a practical example we present a subset of the Hurricane Isabel dataset[3] in Fig. 15(a). We used the properties pressure, water vapor, and x wind component and show selections with different Cast methods in Fig. 15(b)–(d).

[3]See `http://vis.computer.org/vis2004contest/data.html` or `http://www.vets.ucar.edu/vg/isabeldata/readme.html` for details.

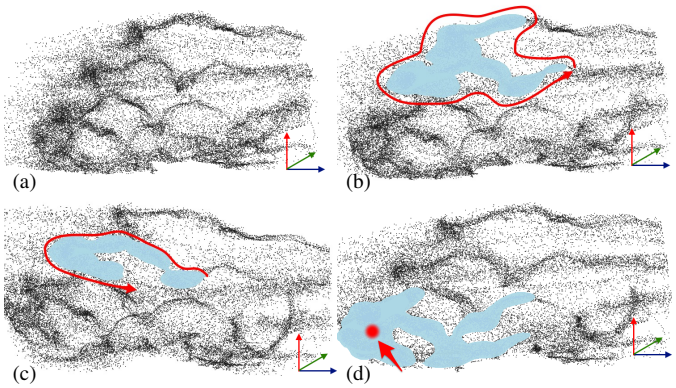

Fig. 15. Cast selection in a 3D scatter plot of an abstract dataset: (a) Pressure, water vapor, and x wind component from the Hurricane Isabel dataset. (b) SpaceCast, (c) TraceCast, and (d) PointCast selection.

5.5 Limitations and Further Improvements

The main limitation of the Cast selection methods is their dependence of an appropriate automatically determined density threshold. In datasets with steep density gradients it is a difficult problem to determine an appropriate initial value of the density threshold, and a manual adjustment of the threshold can improve the selection. Nevertheless, such manual adjustment is also a complication of the otherwise straight-forward interaction used by the Cast methods. Furthermore, density gradients are typically less steep at the edge of a cluster. This means that if a selection volume already roughly matches the cluster boundary, and the user tries to fine-tune the selection, then a small decrease of the density threshold produces a large increase of the selection volume. In the user study, however, we have not found this problem to affect the results.

Nevertheless, the occasional need to manually adjust the density threshold led some user study participants to remark that the TraceCast and PointCast methods are not as accurate as SpaceCast—in the latter the selection is always bound by the lasso. One way of improving this manual threshold setting would be by using a mapping from the slider position to the threshold value so that small movements of the slider produce correspondingly small changes of the selection volume. A very different approach would be to not use the density threshold for determining the cluster boundary but to use other edge-detection algorithms originating in the image recognition domain or using topological analysis to determine an appropriate density isosurface [**?**].

We note here that the Cast methods have been designed for selection in particle datasets with distinct clusters. Particle datasets with different structure characteristics may demand different heuristics for determining an appropriate selection volume.

Finally, in the future we want to combine different selection techniques in an integrated visualization and data analysis environment and perform user studies with real-world scenarios to validate the results of the present study. Since different methods are more appropriate for different types of selection, keeping only one of them in our toolkit would not be optimal. One possible combination would be that of the lasso-based CloudLasso which allows multiple selection with the point-based PointCast which allows efficient selection of individual clusters. Nevertheless, the user study showed that SpaceCast and TraceCast are also very strong contenders and a more thorough and extended study in real-world environments would be necessary to help decide between these very strong selection methods: CloudLasso, SpaceCast, TraceCast, and PointCast. Potentially, all of them could be included, together with Boolean operations that facilitate further fine-tuning.

6 Conclusion

With SpaceCast, TraceCast, and PointCast we introduced a new family of context-aware interactive selection techniques. Through a set of well-designed heuristics, these techniques derive a user's selection intention from the input he or she has provided. The techniques are *effective* because together they cover a wide range of possible selection goals, based on each technique's individual selection algorithm and user input gestures. Furthermore, our study showed that our new techniques are *efficient* as they allow users to arrive at selections faster than with other structure-aware techniques, without jeopardizing their accuracy.

Ultimately, our paper contributes to the development of flexible interaction environments for the exploration of complex data visualizations [**?**]. As selection is needed in virtually any exploratory data analysis, our new Cast family of techniques—together with the older CloudLasso and Boolean operations—provides a powerful toolkit that can support many different selection tasks and datasets in visualization. While we did not specifically design the techniques for any particular type of input device, SpaceCast, TraceCast, and PointCast are suitable not only for mouse and pen-based input but also for newer input and interaction paradigms [**?**] such as those based on touch (e. g., [**?**, **?**, **?**, **?**]).

Acknowledgments

From the University of Groningen, we thank Amina Helmi for the Aquarius Project [**?**] simulation data, Frans van Hoesel for the galaxy collision simulation (orig. from galaxydynamics.org), and Michael Wilkinson for his DATAPLOT [**?**] density estimation code. The Hurricane Isabel dataset was produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR and the NSF. Finally, we used a sample of the Millennium-II dataset [**?**] for some of our examples.