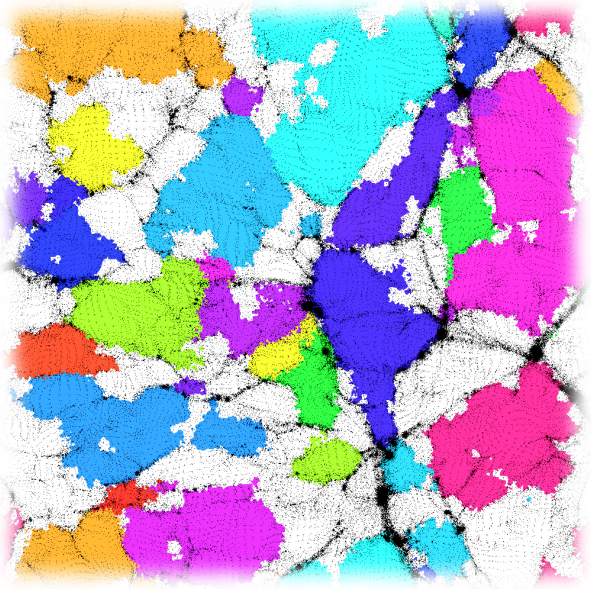
Ludwig–Maximilians–Universität München
Faculty of Physics



Bachelor's Thesis

Voids in Cosmological N–Body Simulations

Voids in kosmologischen Mehrkörpersimulationen

Florian Stecker

Adviser: Dr. Klaus Dolag

February 4, 2014

Contents

1. Structure Formation in the Large Scale Universe

Observations of galaxies up to distances of several hundred Megaparsec ($1\,\mathrm{pc} = 3.086{\cdot}10^{16}\,\mathrm{m}$) show that their positions are not uniformly distributed in space, but instead form a web–like structure. Filaments and clusters accumulating hundreds and thousands of galaxies are interspersed with vast empty spaces of several Mpc in diameter, the cosmic *voids*. In this thesis, we will study these voids using cosmological simulations.

1.1. Cosmological Dynamics

The evolution of our universe on large scales is governed by the laws of gravity, and can therefore be described by the theory of general relativity, in particular by the Einstein field equations. Since they form a complicated system of nonlinear partial differential equations, a solution can only be obtained using vast simplifications. One of the simplest and most successful such models for the universe is the *Friedmann–Robertson–Walker (FRW)* model. It assumes that in some frame of reference, spacetime is a product $I \times S$ of a time interval $I \subset \mathbb{R}$ and a 3–dimensional space S, that this frame of reference is a common rest system for all matter (it is called *comoving coordinates*), and that S is perfectly isotropic. Under these assumptions, it can be derived [ONe83] that spacetime carries a warped product metric with the line element

$$ds^2 = -\,\mathrm{d}t^2 + a(t)^2\,\mathrm{d}\sigma^2.$$

Here $\mathrm{d}t^2$ is the line element on I, $\mathrm{d}\sigma^2$ the line element of S carrying a metric of constant curvature k and $a\colon I \to \mathbb{R}$ a time dependent *scaling function*, which describes the expansion of space: If two material particles are separated by the distance d at time t_1, their distance is $a(t_2)a(t_1)^{-1}d$ at time t_2. On this spacetime, the Einstein equations simplify to the *Friedmann equations*

$$\frac{\ddot{a}(t)}{a(t)} = -\frac{4\pi G}{3c^2}\big(\rho(t,\mathbf{x})c^2 + 3p(t,\mathbf{x})\big) + \frac{\Lambda c^2}{3}, \qquad \frac{\dot{a}(t)^2 + kc^2}{a(t)^2} = \frac{8\pi G\rho(t,\mathbf{x})}{3} + \frac{\Lambda c^2}{3}. \tag{1.1}$$

In these equations, Λ is the universal *cosmological constant*, G the gravitational constant, c the speed of light and ρ and p are the density and pressure fields of the matter filling the universe. The equations show that these quantities depend only on the time coordinate, so the FRW model describes a completely homogeneous universe, which is accurate on very large scales ($\gtrsim 100\,\mathrm{Mpc}$). Measurements have shown that p is small except for a short time range at the beginning of the universe [BT11, figure 1.7] and it is suggested by theory as well as indicated by observations that the universe is flat, i.e. $k = 0$. If we thus simplify the Friedmann equations by setting $\rho = k = 0$, they can be solved by

$$a(t) = \left(\frac{1-\Omega_\Lambda}{\Omega_\Lambda}\right)^{1/3} \sinh(t/T)^{2/3}, \qquad T = \frac{2}{\sqrt{3\Lambda}c}, \qquad \Omega_\Lambda = \frac{\Lambda c^2}{3H_0^2}, \tag{2}$$

using the initial conditions $a(0) = 0$ and $\dot{a}(t_0) = H_0$, with t_0 being the point in time such that $a(t_0) = 1$, which is *today* by convention. It was found by observations that $\Omega_\Lambda \approx 0.76$ and $H_0 \approx 73\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$ [BT11, 1.73] and therefore $t_0 \approx 13.7\,\mathrm{Gyr}$. Interpreting this as the age of the universe is not completely accurate, since neglecting p can not be justified down to $t = 0$. However, the real value is almost the same [BT11, 1.77]. The scale factor a

is strictly increasing, so it can be used as a measure of time instead of t. It is also common in astronomy to use the *redshift* $z = a^{-1} + 1$ to measure time, since it is directly observable in the light spectrum of distant objects. In this measure, $z = 0$ corresponds to today, while the big bang 'happened' at $z = \infty$. Throughout this thesis, time will always be measured in redshift and the symbol z will always mean such a measure of time.

The FRW model is only a rough approximation, since the real universe is far from being homogeneous. If one assumes the real metric of spacetime to be only a small and slowly changing perturbation of the FRW metric, the equations of motion can be approximated by the Newtonian laws of gravity, with an additional ghostly force which uniformly tears the universe apart in any direction, the *Hubble expansion*. The rate of this expansion is then given by the scale factor a: Two objects separated by a comoving distance d physically have the distance $a(t)d$ at time t.

1.2. Primordial Density Fluctuations

A feature of the standard big bang model of cosmology is the absence of an initial point in time. Therefore we have to use the state at some positive time after the big bang as initial conditions for the study of the evolution of the universe. The earliest observable state is redshift $z \approx 1089$, which corresponds according to (2) to $t \approx 0.5\,\mathrm{Myr}$. The nature of density fluctuations at that time can be reconstructed from observations of the anisotropy of the *cosmic microwave background*, a radiation that was emitted during the event of *recombination*, where electrons and protons combined to form neutral atoms, making the universe transparent for electromagnetic waves.

Since only the current horizon can be observed this way, we cannot hope to reconstruct the complete density field, but can only gather statistical data about these *primordial density fluctuations*. They are found to be small compared to the overall density, i.e.

$$\rho(\mathbf{x}) = \overline{\rho}(1 + \delta(\mathbf{x})), \qquad |\delta(\mathbf{x})| \ll 1,$$

where δ is called the *overdensity*. We further see that the overdensity δ forms a realization of a statistically isotropic and homogeneous *Gaussian random field*, which means that the autocorrelation function

$$\xi(\mathbf{x}) = \langle \delta(\mathbf{y})\delta(\mathbf{x} + \mathbf{y}) \rangle_{\mathbf{y}}$$

only depends on $|\mathbf{x}|$ and completely determines the statistical properties of δ, in the sense that for any tuple $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of points, and any set $\Omega \subset \mathbb{R}^N$, the probability for a point $\mathbf{x}$ to satisfy

$$(\delta(\mathbf{x} + \mathbf{x}_1), \ldots, \delta(\mathbf{x} + \mathbf{x}_N)) \in \Omega$$

is given by the multivariate Gaussian distribution

$$\frac{1}{\sqrt{(2\pi)^N |\det \Sigma|}} \int_\Omega \exp\left(-\frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x}\right) \mathrm{d}\mathbf{x}, \qquad \Sigma = \begin{pmatrix} \xi(\mathbf{x}_1 - \mathbf{x}_1) & \ldots & \xi(\mathbf{x}_1 - \mathbf{x}_N) \\ \xi(\mathbf{x}_2 - \mathbf{x}_1) & \ldots & \xi(\mathbf{x}_2 - \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ \xi(\mathbf{x}_N - \mathbf{x}_1) & \ldots & \xi(\mathbf{x}_N - \mathbf{x}_N) \end{pmatrix}.$$

Since ξ is a real symmetric function, its Fourier transform is real and non–negative. Just like ξ, its value only depends on the norm of the argument. We can therefore define the

power spectrum P of the random field by $\xi(\mathbf{k}) = P(|\mathbf{k}|)$. It can be seen in observations, and deduced from theory, that the power spectrum of the primordial density fluctuations is of the form $P(k) \propto k$.

As time proceeds, the denser regions attract surrounding matter, which is drained out of regions of lower density. This movement of matter out of the primordial density minima results in an amplification of the overdensity. In the early stage of the evolution, up to $z \approx 20$, the density fluctuations are small enough to be well approximated by a linear growth of δ. After this point, the process becomes more diverse: Streams of matter being repelled out of the growing fields of low density, collide at the borders of these and get pressed into dense two–dimensional *sheets* along these borders. This gives the universe a structure similar to honeycombs, with dense walls surrounding vast regions almost devoid of any matter. These regions, the *voids*, reach diameters of up to 100 Mpc.

The density is even higher along the intersections of these sheets, since they accrete matter from several surrounding voids. This overdensity leads to a flow of matter from the sheets to their one–dimensional intersection regions. These *filaments* are usually the regions where matter gets compressed enough to form stars and galaxies. The intersection points of filaments form big galaxy clusters, which again pull in matter from their surrounding filaments. Using large catalogues of positions and distances of galaxies, the positions of these filaments and thus the voids surrounded by them can be identified. The quite old Abell catalogue [Abe58] already allowed to see a handful of voids, while the modern Sloan Digital Sky Survey [Aba+09] shows several hundreds of them [Sut+12; NH13].

As the shape of voids is not a product of chaotic nonlinear processes or complicated microphysics, but instead directly originates from the primordial density fluctuations (only influenced by gravitational forces), voids are believed to contain relatively raw information about these initial conditions. This is one of the reasons why they have been extensively studied during the last years.

2. Cosmological N–body SPH Simulations

The most fruitful approaches to studying voids theoretically use cosmological simulations. These simulations primarily model the gravitational evolution of a given initial distribution of matter. In addition to gravity, they often apply extra physics like hydrodynamics of the baryonic matter component, star formation and development, supernova feedback, interstellar winds, black holes, magnetic fields, etc. This is not strictly necessary to study voids since their formation is dominated by gravity. However, including the processes leading to star formation should allow a comparison of the simulation results to observational data.

For this thesis we will use data from *Dianoga*, a large dark–matter only simulation (thus modelling only gravitational interaction) of relatively low resolution, and *Magneticum Pathfinder*, a set of simulations of different sizes and resolutions involving a multitude of physical phenomena. Both simulations were created using the simulation software *Gadget 2* [Spr05]. Gadget is a multi–purpose N–body SPH (Smoothed Particle Hydrodynamics) code, which is used for huge cosmological simulations with up to 300 billion particles [Ang+12] as well as high resolution re–simulations of galaxies or isolated galaxy mergers [Sch+13].

2.1. Discretization

Instead of a continuous matter density, N-body simulation codes like Gadget deal with a set of pointlike, randomly distributed particles with fixed mass and variable position. In each step, gravitational and hydrodynamical forces between these particles are calculated and the previous positions and velocities of the particles are then updated using a kick–drift–kick procedure. In practice, Gadget carries out the force calculations using a smoothed density distribution obtained by convolving the particle distribution with the smoothing kernel

$$W(\mathbf{x}, h) = \frac{8}{\pi h^3} \begin{cases} 1 - 6h^{-2}|\mathbf{x}|^2 + 6h^{-3}|\mathbf{x}|^3 & 0 \leq |\mathbf{x}| < h/2, \\ 2 - 6h^{-1}|\mathbf{x}| + 6h^{-2}|\mathbf{x}|^2 - 2h^{-3}|\mathbf{x}|^3 & h/2 \leq |\mathbf{x}| < h, \\ 0 & |\mathbf{x}| > h \end{cases} \tag{2.1}$$

where h is the respective *smoothing length*. It is set to a constant small value for the gravity calculations, since in this case it is only intended to avoid numerical errors for very near particles. For the hydrodynamical part however, the smoothing length is particle dependent and the algorithm essentially depends on overlapping particle kernels.

2.2. Initial Conditions

In theory, cosmological simulations should reproduce the evolution of the universe in the whole time range from $z \approx 1089$ (where we know the density fluctuations from observations of the cosmic microwave background) to $z \approx 0$ (which allows comparison with observed nearby objects). A continuation to negative redshifts is believed not to show any new phenomena. At very high redshifts, the density fluctuations are still weak and numerical errors are therefore large compared to the physical processes. To avoid this, linear theory is used in this regime, i.e. δ is just multiplied with a time–dependent constant. In the Magneticum Pathfinder simulation for example, the primordial density fluctuations are linearly evolved up to $z \approx 23$, where they are used to generate initial conditions for the actual simulation.

To transform a density field into a set of particles, a random particle pattern of approximately uniform density is used and then these particles are displaced so that the desired density field is achieved. This density field is usually just a linear combination of sine waves of different frequencies and random phases, with their amplitudes depending on the frequency according to a predefined power spectrum.

2.3. Gravity

In the regime of cosmological simulations, using comoving coordinates, general relativity is simplified to Newtonian gravity with an artificial expansion factor following Equation (2). As long as matter is collisionless, i.e. not subject to any interaction except gravity, the motion of the individual matter particles is determined by the Hamiltonian function

$$H = \sum_i \frac{\mathbf{p}_i^2}{2m_i a(t)^2} - \frac{1}{2}\sum_{i,j} \frac{Gm_i m_j}{a(t)|\mathbf{x}_i - \mathbf{x}_j|},$$

where the sum runs over all matter particles i with momenta $\mathbf{p}_i$, positions $\mathbf{x}_i$ and masses m_i [Spr05]. This is applicable to the physical elementary particles as well as the virtual particles in cosmological simulations with masses of up to $10^{10}\, M_\odot$.

Gadget uses a tree N–body algorithm to calculate gravity. Due to the long–range nature of gravity, in an N–particle simulation, each particle is subject to $N-1$ interactions with other particles. With particle numbers of up to 10^{11} as they are used in current simulations, calculating the forces between every pair of particles separately in every time step would be far beyond computational capabilities. Instead, a tree algorithm like Gadget, when calculating the force exerted on a particle A, adds only the contributions of particles in the direct neighbourhood of A precisely. More distant particles are clustered together and substituted by virtual particles carrying their total mass.

This is achieved by partitioning the simulation volume into an octree and calculating the total contained mass for every node of this tree. Now, when calculating the force on A, the octree is traversed from the root downwards. At every node a heuristic criterion determines whether the virtual particle associated to the current node is a precise enough approximation, or if instead the node has to be 'opened' and its child nodes have to be considered recursively. By adjusting the opening criterion, precision can be traded against computational performance.

2.4. Hydrodynamics

Almost 85% of the mass in the universe is believed to be dark matter, which only interacts by gravity. The remaining $\approx 15\%$ are partly contained in stars, which can be approximated as collisionless and therefore follow dynamics similar to dark matter. The baryonic (i.e. non–dark) matter not bound in stars fills the cosmos as a thin gas with a distribution which resembles on large scales that of dark matter. However, it is also subject to hydrodynamical forces and thermal conduction. Hydrodynamics are simulated in Gadget by using the *smoothed particle hydrodynamics (SPH)* method. The point mass distribution is smoothed by the kernel shown in Equation (2.1), with an adaptive smoothing length chosen such that the kernels of multiple particles generously overlap. Any quantity like density, velocity, temperature etc. at any point can then be expressed as a linear combination of the respective

properties of the particles. The derivatives of these quantities are then linear combinations of the derivatives of the smoothing kernel, which in the case of Equation (2.1) is a simple polynomial expression. In this way, the hydrodynamic equations of motions simplify to algebraic expressions for the forces acting on the particles in each timestep.

2.5. Additional Physics

Star formation happens on much smaller scales than current cosmological simulations can resolve, so one relies on simplified schemes. Basically, when the gas contained in a region is dense and cold enough (radiative cooling is also included in the model), Gadget begins to transform a fraction of the gas particles to star particles. As a consequence the surrounding gas is heated, simulating the effect of supernova explosions [SH03]. Furthermore, Gadget employs schemes for the evolution of the stars, in particular their enrichment with heavier elements from hydrogen burning and supernovae. Galactic winds can be established as a result of supernova explosions, fed by a fixed fraction of the supernova energy. Black hole particles are placed in the center of sufficiently large mass accumulations, which grow by swallowing particles passing close enough. These are just some examples of the microphysical processes which Gadget can simulate in addition to gravity and hydrodynamics.

3. Finding Voids

Since massive objects like galaxies and dark matter halos tend to form rotationally symmetric shapes with a radially decreasing density, their extents can easily be defined as a spherical volume around the center of mass having a certain overdensity with respect to the surrounding universe. Voids, on the other hand, with their highly irregular shape and the lack of a physically meaningful center of mass, are significantly harder to outline. This problem is normally dealt with by defining an algorithm which marks certain regions in an observed or simulated mass distribution as 'voids' [Col+08]. This algorithm then serves as a definition of the extent of the void as well as a tool to find actual voids in a data set. Unfortunately, there are dozens of approaches to do this, and the obtained results often differ dramatically. This is nicely visualized by Figure 3.1, which was taken from [Col+08]. The authors extracted the $(60\,h^{-1}\,\mathrm{Mpc})^3$ region shown in the first panel from a big dark matter simulation and applied different void finding algorithms on it. The first panel in Figure 3.1 shows a projection of the used dark matter distribution. The seven other panels show the results of different algorithms: Regions identified as voids are marked in green, *void galaxies* (i.e. galaxies contained in voids) in blue. The red dot shows the *void center*, the definition of which also depends on the algorithm.

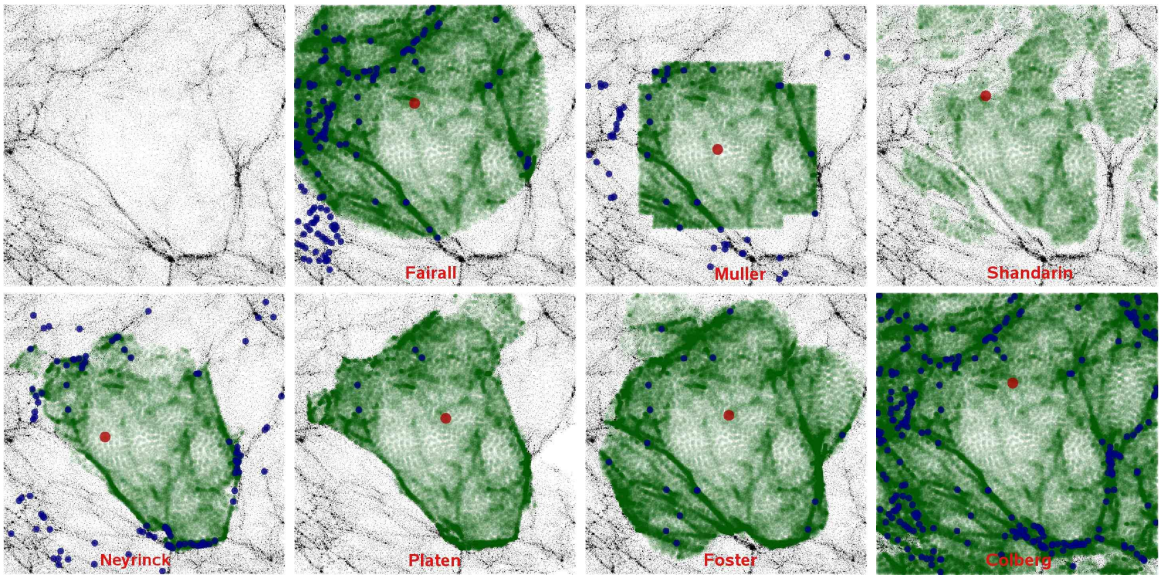


Figure 3.1: The results of different void finders can be completely different. This figure is a copy of Figure 1 in [Col+08], but rearranged and reduced by some panels.

In this thesis, we will use the void finding algorithm *ZOBOV* by Mark Neyrinck [Ney08], which corresponds to the first panel of the bottom row in Figure 3.1. It is a modification of his earlier cluster finding algorithm *VOBOZ* [NGH05]. ZOBOV excels through a particularly high agreement of the resulting void extents with the intuitive expectation when looking at a 2–dimensional section through the density distribution (see e.g. Figure 4.1 and Figure 3.1). As an additional advantage, ZOBOV uses only one parameter—a minimum density contrast between the void's center and its border—which makes it more suitable as a definition then

void finders that depend on more parameters. The main drawback of ZOBOV is its high computational complexity, which seems to have so far precluded its wider adaption.

3.1. The Watershed Analogy

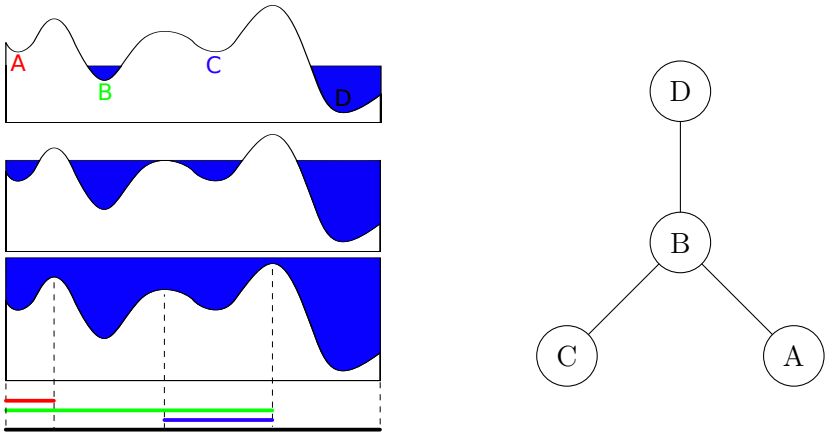
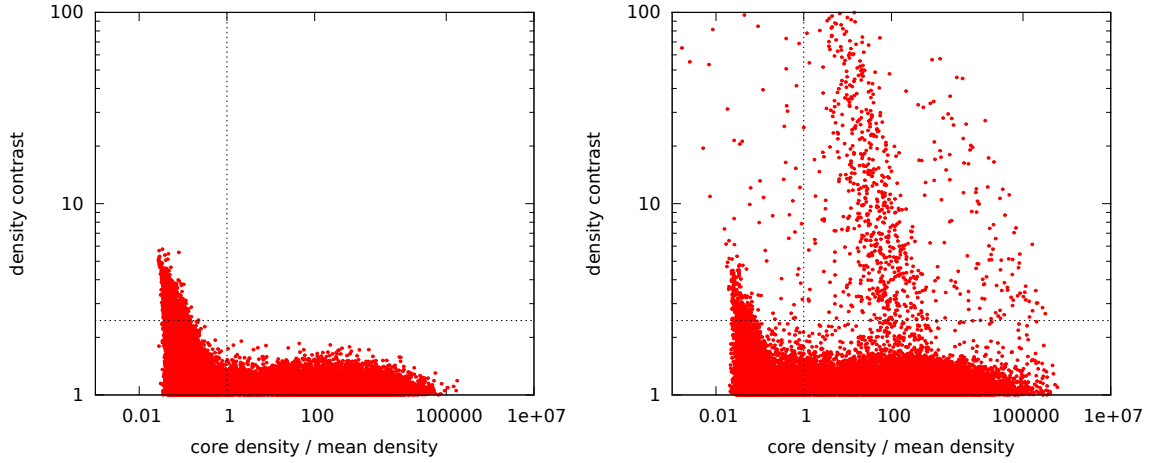


Figure 3.2: Left: An approach to finding watersheds in a mountain range. While the landscape is filled with water, lakes form around the local minima (upper panel), which combine at the saddle points to bigger lakes (center panel). Ultimately, the whole range is covered with water, resulting in one big lake (lower panel). Right: The resulting void tree in this setup.

The main idea of ZOBOV (as well as of the *Watershed Void Finder* [PvJ07], the second panel in the bottom row of Figure 3.1, which served as a prototype for ZOBOV) is derived from the geographic problem of finding watersheds based on the height profile of a mountain range. An idea to approach this problem is the following: We slowly fill the mountain range with water, by defining a time dependent *sea level*, which initially is lower than the minimum height of the deepest valley. As time evolves, we steadily rise the sea level and regard all parts of the landscape which lie below it as underwater (we don't care if there are any passages through which water can actually get there). Small lakes will form around the local height minima and will grow as the sea level rises (see Figure 3.2). If it has risen enough to reach a saddle point of the height profile, two lakes will join. We then declare the shallower of the two lakes to have reached its final size. From then on it is considered to be a part the deeper lake. This continues until the whole landscape is underwater. Now only one lake is left, namely the one originating from the global height minimum.

As indicated in the right part of Figure 3.2, the result is a partitioning of the total area into a tree of nested subsets. Every local minimum of the height profile has a lake associated to it, and the surface of the maximum extent of this lake (i.e. its surface when it was swallowed by a deeper lake) assigns to each local minimum an area of the mountain range, which is supposed to be its drainage basin. These drainage basins form the nodes of a tree structure, with the lake coming from the global minimum - filling out the whole area - as root and with the areas of the lakes swallowed assigned as children to the respective node. In the example, the root is D with B as its only child. The children of B are the two remaining drainage



(a) Box2 mr

(b) Box3 hr

Figure 3.3: Density contrast and core density of the voids found in Box2 mr and Box3 hr at $z = 0$.

basins, A and C.

Though it is disputable whether this algorithm is really suitable for determining watersheds, it can be adapted to the problem of finding voids. The density field of the universe—for example, the dark matter density from an N–body simulation—may be considered as a 3–dimensional landscape within 4–dimensional Euclidean space. Then the surfaces of the lakes correspond to void extents, and the algorithm yields a tree of voids and their subvoids, appropriately representing the hierarchical structure of void formation.

3.2. Description of the ZOBOV Void Finder

We will now describe in detail the ZOBOV algorithm for finding voids. The version presented here is equivalent to the original one in [Ney08], but it is slightly reformulated with regard to a simpler implementation using trees in order to enhance computational performance.

A practical problem of the watershed approach is that we first need to estimate somehow the density field using the simulation output. SPH simulations produce such an estimate as a byproduct, but only for the gas component, not for dark matter. Another issue is the complicated geometry of lake filling. We would like to have a more discrete structure, which is easier to handle computationally.

ZOBOV solves both these problems by using a *Voronoi tessellation*: Given particle positions $\{\mathbf{x}_i\}_{i=1,\dots,N}$ from an N–body simulation, the *Voronoi cell* of $\mathbf{x}_i$ is defined as the set of all points $\mathbf{x}$ with $|\mathbf{x} - \mathbf{x}_i| < |\mathbf{x} - \mathbf{x}_j|$ for all $j \neq i$. We estimate the density $\rho(\mathbf{x}_i)$ of the particle $\mathbf{x}_i$ as the quotient of its mass and its Voronoi cell volume. Since the Voronoi cells form a pairwise disjoint decomposition of the total volume (except for their borders), the mean of these densities is indeed the total mass of all particles divided by the whole simulation volume. In addition to this density estimate, we get a neighbouring relation of particles: Two particles are *neighbours* or *adjacent* if their Voronoi cells share a common border.

In practice, the Voronoi tessellation can be computed by lifting the particle positions to a 3–dimensional paraboloid in $\mathbb{R}^4$, mapping a point (x, y, z) to $(x, y, z, x^2 + y^2 + z^2)$. Then the convex hull of these points is computed, for example using the *quickhull* algorithm [BDH96]. The convex hull is a polyhedron, so its surface consists of polygons. It can be shown that two particles are neighbouring in the above sense if they are joined by an edge of one of these polygons [PS88].

Now that we have the neighbouring relations, we can use this to calculate the Voronoi cell volume of a particle $\mathbf{x}_i$. Each neighbour $\mathbf{x}_j$ of $\mathbf{x}_i$ gives rise to a *half–space* defined by the set of points whose distance to $\mathbf{x}_j$ exceeds the one to $\mathbf{x}_i$. The intersection of the half–spaces generated by all neighbours of $\mathbf{x}_i$ is the Voronoi cell of $\mathbf{x}_i$. For the calculation of convex polyhedrons and their volumes from half–space intersections convex hull algorithms [BDH96] can be applied again. The quickhull algorithm and its applications to various geometric problems are implemented in the publicly available *qhull* library (`http://www.qhull.org`).

The next step of the ZOBOV algorithm groups the Voronoi cells (or equivalently the particles) into connected *zones*: Each Voronoi cell which is a local density minimum, i.e. all its neighbours are of greater density, forms the core of such a zone. Every other Voronoi cell recursively belongs to the same zone as its least dense neighbour. A neighbouring relation of zones is induced: Two zones A, B are said to be neighbours if there are one or more pairs of particles $\mathbf{a}_1, \mathbf{b}_1, \ldots, \mathbf{a}_n, \mathbf{b}_n$ where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are contained in A and $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in B, such that $\mathbf{a}_i$ is adjacent to $\mathbf{b}_i$ for all i. The pair (A, B) of neighbouring zones then is assigned a *smallest linking density*

$$\mathrm{sld}(A, B) = \min_{i=1,\ldots,n} \max\{\rho(\mathbf{a}_i), \rho(\mathbf{b}_i)\}.$$

In the watershed analogy, the smallest linking density corresponds to the height of the saddle where the two lakes A and B combine. Also we assign every zone A a core density $\rho_{\mathrm{core}}(A)$, which is just the minimum of the densities of all particles contained in A. This represents the deepest point of the lake.

In a third step ZOBOV uses these data to arrange the zones in a tree structure. Initially, every zone defines its own tree with itself as the only node. These trees will subsequently be joined into one big tree. For this purpose, the pairs of neighbouring zones are gone through in ascending order of their smallest linking densities, simulating the rising sea level. For each such pair (A, B), both A and B are nodes of some tree. Let $\widehat{A}$ be the root node of the tree containing A and $\widehat{B}$ be the root node of the tree containing B. If $\widehat{A} = \widehat{B}$, both A and B are already in the same tree, so ZOBOV does nothing and proceeds with the next neighbouring pair. Otherwise, if $\widehat{A} \neq \widehat{B}$, the two trees are joined in a way depending on the core densities of $\widehat{A}$ and $\widehat{B}$. If $\rho_{\mathrm{core}}(\widehat{A}) < \rho_{\mathrm{core}}(\widehat{B})$, then $\widehat{B}$ becomes a child node of $\widehat{A}$. For later use, we then assign $\widehat{B}$ a property called the *minimum border density (mbd)*, by setting $\mathrm{mbd}(\widehat{B}) = \mathrm{sld}(A, B)$. If, on the other hand, $\rho_{\mathrm{core}}(\widehat{B}) < \rho_{\mathrm{core}}(\widehat{A})$, then the node $\widehat{A}$ becomes a child of $\widehat{B}$ and $\mathrm{mbd}(\widehat{A}) = \mathrm{sld}(A, B)$. In this way, the trees combine finally in a single tree containing all zones. The root of this tree will be the zone with the lowest core density. We set its mbd to infinity.

The resulting tree can be interpreted in different ways. Firstly we can regard any node as representing a void, with its extent defined to be the union of its zone and the zones of all subordinate nodes in the tree. The tree then represents a hierarchical structure of voids and their subvoids. These voids are of course not disjoint. On the contrary the root node

represents a huge 'void' comprising the whole simulation volume, of which all the other voids are subvoids. This is analogous to the situation shown at the bottom of Figure 3.2.

For most applications however, voids should be disjoint underdense regions. This can be achieved by splitting up the tree generated above. For this we first have to supply a criterion which determines whether a zone in the tree should be regarded as a void or just as random noise. [Ney08] proposes two properties appropriate to accomplish this distinction, which can also be used together: The first property is a *density contrast (dc)*, defined for each zone A as

$$\mathrm{dc}(A) = \mathrm{mbd}(A)/\rho_{\mathrm{core}}(A).$$

It evaluates the minimum density of the void border relative to the density minimum of the void, and can be regarded as a measure for the statistical *significance* of a void (see [Ney08]). The second property is just the core density $\rho_{\mathrm{core}}(A)$. Figure 3.3(a) shows a scatter plot of the density contrasts and the core densities for all zones found in a typical N–body simulation. Only those zones may be considered as real voids which exceed a certain threshold for the density contrast and/or remain below a certain threshold for the core density. This corresponds to the upper left quarter in Figure 3.3(a). However, we see in Figure 3.3(a) that a sufficiently high threshold density contrast also filters out all high density zones, which allows to omit the second criterion in this case.

The same plot for the higher resolution simulation Box3 hr, as shown in Figure 3.3(b), reveals a problem. In addition to the usual zones there are some seemingly randomly scattered data points. These will later correspond to very small voids (almost all of them have diameters of less than 5 Mpc) and are probably due to numerical error. Their exact reason remains a mystery so far. But as they do not occur in the lower resolution boxes, we will ignore this for now.

After we have specified a filter separating the zones into *real voids* and *fake voids*, the void tree is cut into disjoint void volumes using the following procedure: If a zone A has a sibling B in the tree, which is a real void, and $\mathrm{mbd}(B) < \mathrm{mbd}(A)$, then A is marked as *no void*. In the rising water analogy, these are the regions a real void gains after it has already combined with another real void. They are not uniquely assignable to one of the voids and turn out to be the higher density inter–void regions. Next to every real void the volume of all its descendant zones is assigned except for those that are 'no void' or another subordinate real void or a descendant of such. This yields a set of disjoint void regions, normally separated by sheets and filaments, which are interspersed with regions which belong to no void at all, as it can be observed e.g. in Figure 4.1.

4. Void Properties

We applied the ZOBOV algorithm to different cosmological simulations, which are listed in Table 4.1. Box3 mr, Box3 hr and Box2 mr are three different boxes from the Magneticum Pathfinder project, a set of cosmological simulations involving self–gravity, hydrodynamics, star formation and numerous other physical effects, which are of minor importance for our purpose. Dianoga, on the other hand, is a dark matter only simulation at relatively low resolution, primarily intended for the study of galaxy clusters.

name	component	V	N	divs	runtime	snapshots
Box3 mr	dark matter	$(128\,h^{-1}\,\mathrm{Mpc})^3$	216^3	2^3	$\sim 40\,\mathrm{min}$	36, $z \in [0, 23]$
Box3 mr gas	gas	$(128\,h^{-1}\,\mathrm{Mpc})^3$	$\lesssim 216^3$	2^3	$\sim 40\,\mathrm{min}$	36, $z \in [0, 23]$
Box3 hr	dark matter	$(128\,h^{-1}\,\mathrm{Mpc})^3$	576^3	6^3	$\sim 14\,\mathrm{h}$	1, $z = 0$
Box2 mr	dark matter	$(352\,h^{-1}\,\mathrm{Mpc})^3$	594^3	6^3	$\sim 13.5\,\mathrm{h}$	1, $z = 0$
Dianoga	dark matter	$(1000\,h^{-1}\,\mathrm{Mpc})^3$	1024^3	10^3	$\sim 84\,\mathrm{h}$	1, $z = 0$

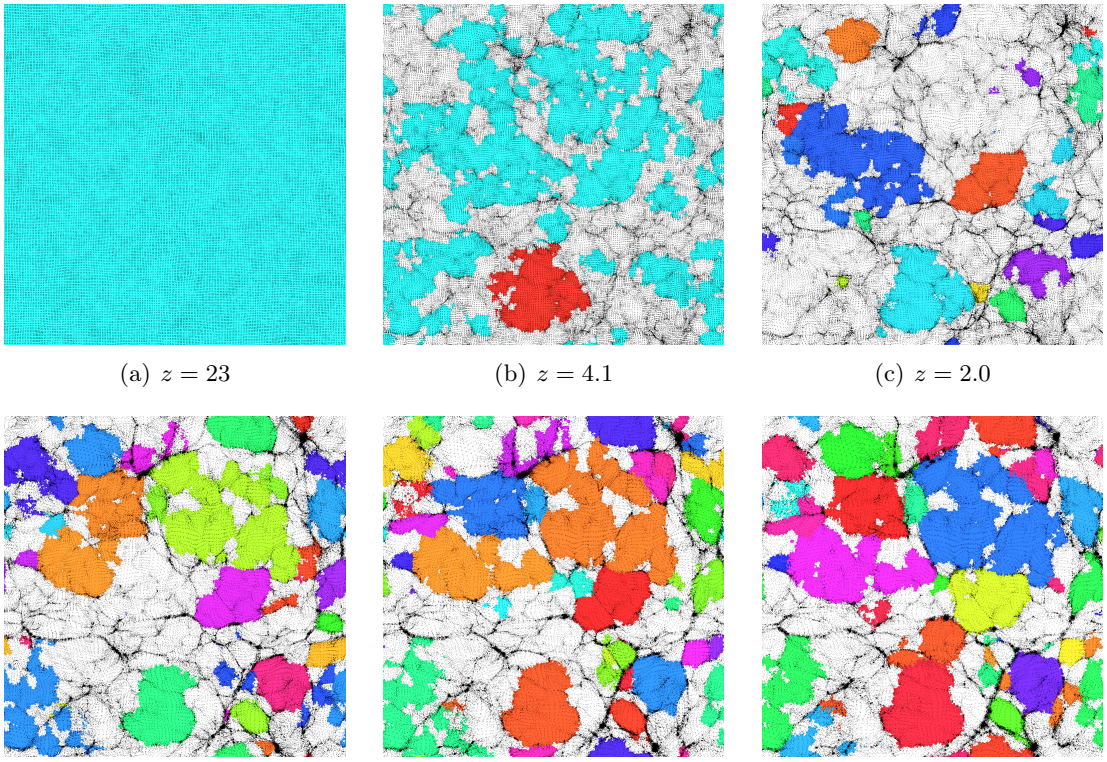
Table 4.1: The simulation boxes. Listed are the name of the simulation, the considered component, the volume (V) of the simulation box, the total number (N) of particles of the desired component, the number of subvolumes (divs) the box was divided into for the Voronoi tessellation, the runtime of a Voronoi tessellation for a single snapshot on a single processor core, and the number of examined snapshots with their redshift range

The volumes listed in column 3 are given in comoving coordinates. Physical lengths can be obtained from these comoving lengths by dividing them by $z + 1$. Box3, for example, starts at $z = 23$ with a physical extent of about 7.62 Mpc in each dimension and expands by the Hubble flow to a length of 183 Mpc at $z = 0$ (assuming $h = 0.7$).

The 'runtime' column lists the approximate time needed to perform the Voronoi tessellation (which is by far the most time consuming step of the ZOBOV algorithm) on a single processor core. It should be noted that due to the high memory consumption of about 3 GB per core it is usually not possible to use all cores available on the system. The 'divs' column shows the number of subvolumes the box was divided into for the Voronoi tessellation. This was chosen such that there were approximately 10^6 particles in each of the subvolumes, which is a number the quickhull algorithm can handle with a reasonable amount of computing resources.

For the Box3 mr simulation, voids were searched for in the dark matter as well as in the gas component, based in both cases on 36 snapshots at different times, ranging roughly from $z = 23$ to $z = 0$. For the other three simulations (Box3 hr, Box2 mr and Dianoga), only a single snapshot at $z = 0$ of the dark matter component was considered due to the relatively high computational cost.

Figure 4.1 visualizes the result of Box3 mr. Voids were filtered using the density contrast criterion described in subsection 3.2 with a threshold value of 2.89, without any additional constraint on the core density. For six different snapshots at redshifts $z \in \{23, 4.1, 2.0, 0.78, 0.34, 0\}$ Figure 4.1 shows a cross section through the simulation box at $x_3 = 100\,h^{-1}\,\mathrm{Mpc}$. The intersections of the void volumes with this plane are shown in different colors for each void. In addition, all dark matter particles in a 5 Mpc thick slice around



(a) $z = 23$ (b) $z = 4.1$ (c) $z = 2.0$

(d) $z = 0.78$ (e) $z = 0.34$ (f) $z = 0$

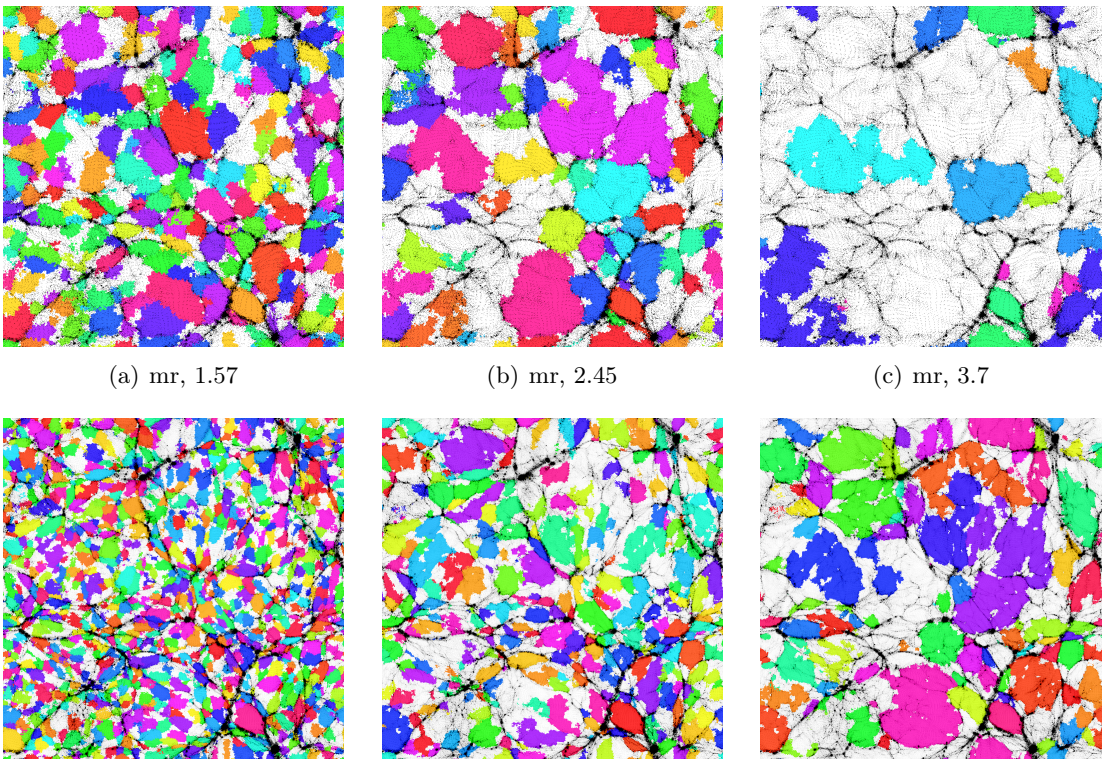
Figure 4.1: Slices of 5 Mpc thickness through different snapshots of the Box3 mr simulation and the voids found by ZOBOV with a density threshold of 2.89. The dark matter particles are shown as black dots and the voids are marked in distinct colors. White regions are not part of any detected void.

the $x_3 = 100\,h^{-1}$ Mpc plane appear as black dots.

In the $z = 23$ snapshot the density fluctuations are still much too weak to form anything that would satisfy the density contrast condition for a threshold of 2.89. The algorithm then regards the complete volume as one big void, as explained in subsection 3.2. The $z = 4.1$ snapshot shows a transition phase, in which density fluctuations have grown large enough to exceed a contrast of 2.89, but the sheets surrounding void regions are not developed enough to delimit the voids from each other. This results in a very low number of voids (only two in this figure), roughly covering the underdense regions, but not dividing them reasonably into distinct voids. At $z = 2.0$, the void finding algorithm finally works as intended. Now compact regions of sufficient underdensity to fulfill the density contrast condition are identified, though they are still rare. At $z = 0.78$ more regions have become underdense enough to satisfy the void criterion and the void count has increased significantly. In the last two snapshots the changes are less dramatic. The voids slightly expand as a result of the contraction of their surrounding sheets. New voids now mostly form by a big void splitting up into two or more separate voids when its internal structures have grown significant enough to qualify its subvoids as stand–alone voids.

4.1. Parameter Dependence

Before further analyzing the results, we want to examine the influence of parameters like the resolution of the simulation and the density threshold parameter of ZOBOV. We expect the distribution of the matter components (dark matter, gas, stars) to be qualitatively the same on large scales. Therefore we also hope to find similar results when applying ZOBOV to different components.



(a) mr, 1.57 (b) mr, 2.45 (c) mr, 3.7

(d) hr, 1.57 (e) hr, 2.45 (f) hr, 3.7

Figure 4.2: Slices of 5 Mpc thickness through different snapshots of the Magneticum Box3 simulation at medium and high resolution and for 3 different density contrasts (1.57, 2.45 and 3.7). The dark matter distribution is shown in black and void areas are marked by colors. Regions not contained in any void are white. The opacity of the black dots was chosen in a way that the different resolutions are comparable.

The high resolution (hr) version of Magneticum's Box 3 starts with the same initial conditions and features the same physics as the medium resolution version examined above, but represents the matter distribution by about 19 times as many particles. While the higher resolution is crucial for many applications in cluster and galaxy physics, the large scale structure is almost invariant, as it is a direct amplification of the initial density fluctuations. Figure 4.2 therefore exhibits a very similar dark matter distribution for both resolutions. However, the results of void finding seem to differ heavily. Figure 4.2 shows this for three density contrast thresholds, 1.57, 2.45 and 3.7. A higher similarity is observed when com-

paring the left upper image (mr, density threshold 1.57) with the middle lower image (hr, density threshold 2.45) and the middle upper image (mr, density threshold 2.45) with the right lower image (hr, density threshold 3.7). The higher resolution has almost the same effect on void finding as a lower density threshold parameter.

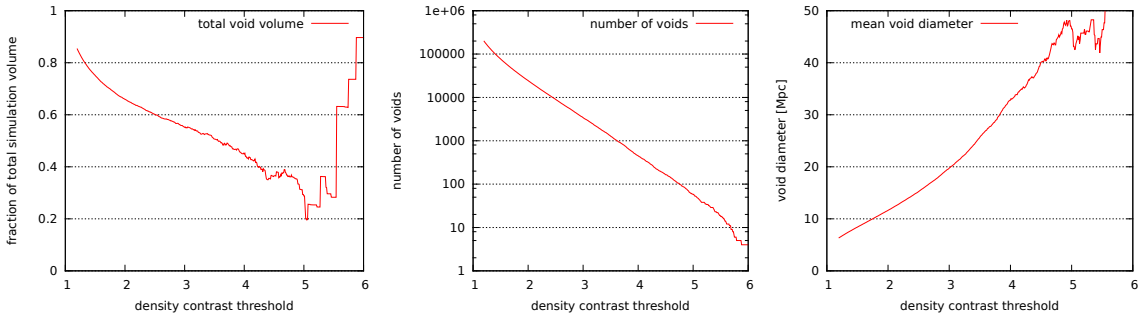


Figure 4.3: Void properties as a function of the density contrast threshold for Box2 mr at $z = 0$. Left: The fraction of the total simulation volume contained in voids. Center: The number of voids identified by ZOBOV. Right: The mean diameter of all voids found.

The effect of this parameter may be examined more closely. If a lower density contrast is admitted, more less significant density fluctuations are detected as voids. Therefore voids are now seen in regions which are slightly underdense, but did not match the void criterion under the higher density threshold parameter. Consequently the fraction of the simulation volume contained in voids increases with decreasing density threshold, as can be seen in in Figure 4.3(a). However, for high values ($\gtrsim 5$ in this case) the small number of remaining voids becomes an issue. By design, the algorithm expands a void at least to an extent that it touches another void. If there is no other significant void nearby in any direction it will grow much larger than physically reasonable, even including dense structure. This leads to an increase in the total void volume fraction up to almost 100%. However, this has to be regarded as a shortcoming of the ZOBOV algorithm and not a real feature.

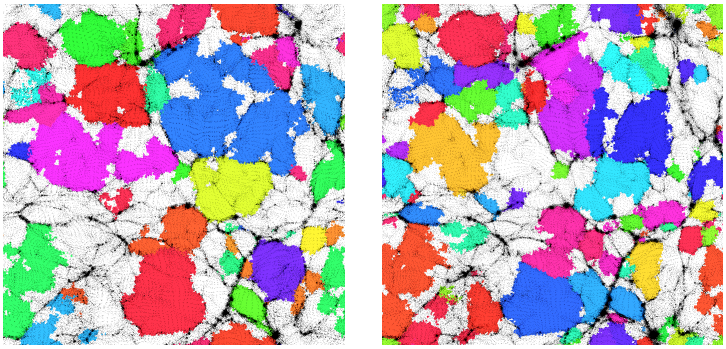
Figure 4.3(b) reveals that the number of voids found also increases with a lower density contrast threshold. This can not be explained by the growth of the total void area described above, but is caused by a second effect of a lower density contrast parameter, namely that the substructure of a former single big void can now be significant enough to be considered as a void on its own, and the big void thus decomposes into several smaller voids. This leads to the vast increase in the void number for low density contrast thresholds seen in Figure 4.3(b).

At very low density thresholds a third effect occurs. Since we do not specify a maximum core density for our void criterion, regions are detected as voids that are not underdense with respect to the mean of the whole simulation, but are surrounded by even much denser regions. These voids are likely a result of the imperfection of density estimation by Voronoi cells and not physically significant features. Fortunately, at medium resolution, these artifacts can easily be detected by their high density and exceptionally low volume and could be excluded from further analysis. In [Ney08] it was proposed to exclude all voids having a minimum density of more than $0.2\rho_{\text{mean}}$. But as these artifacts are only an issue at very low density

contrast thresholds, we can neglect this problem. However, at higher resolutions, it is much more difficult to filter these disruptions, since some of them become significantly underdense, as shown in Figure 3.3(b). Filtering by void volume remains a possible solution. However, the high resolution version was only included for comparison and this was therefore not investigated further.

Figure 4.3(c) shows the mean diameter of all voids found at a given density threshold. We define the diameter of a void as the diameter of a spherical region of the same volume, i.e. as the length d satisfying $V = \frac{1}{6}\pi d^3$ for the void volume V. The volume of a void is given by the sum of the volumes of all contained Voronoi cells.

Though ZOBOV is designed to be used for dark matter distributions, it is worth noting that it can be applied to search voids in an arbitrary set of particles, for example the distribution of gas or stars from the simulation. The application to stars is of particular interest, because it allows direct comparison with observations. E.g. in [Sut+12] the ZOBOV algorithm was used to find voids in a set of observed galaxies with redshifts up to $z = 0.44$ from the *Sloan Digital Sky Survey*.



(a) dark matter (b) gas

Figure 4.4: Voids found at a density threshold of 2.89 in the $z = 0$ snapshot of Box3 mr, using the dark matter and gas components, respectively. The matter distribution in a slice of 5 Mpc thickness is shown in black and void areas are marked by colors. Regions not contained in any void are white.

To check if voids are formed by all matter components in the same manner, we consider a $5\,h^{-1}$ Mpc thick slice through $x_3 = 100\,h^{-1}$ Mpc at the $z = 0$ snapshot of Box3 mr. The first panel of Figure 4.4 depicts the dark matter contained in this slice as well as the voids found by ZOBOV when applied to the dark matter distribution in this snapshot, which intersect the $x_3 = 100\,h^{-1}$ Mpc plane. The second panel of Figure 4.4 shows the same scene using the gas distribution instead of dark matter. As expected, the result is very similar, though not exactly the same. In particular, very large voids found in the dark matter distribution often split up to several distinct voids in the gas component. This difference is also reflected in the total void number: While the snapshot under consideration yields 611 voids for the gas component, only 190 voids were found in the dark matter distribution. This leads to the assumption that void substructures are more prominent in the gas component.

The star component is even more interesting due to its comparability to observations. However, though ZOBOV was already used for observed star distributions [Sut+12], it did

not yield reasonable results applied to the star distributions from the Magneticum Pathfinder simulations. The 'voids' found consisted of just a view particles and were randomly scattered through the simulation volume. Possibly the density threshold parameters need to be chosen some orders of magnitude higher to achieve results similar to those of the dark matter and gas components.

4.2. Void Shape Statistics

We now want to examine properties of the void regions found and how they evolve in time. We have already defined the diameter $d = (6V/\pi)^{1/3}$ of a void. Figure 4.5 shows its probability density function for all voids found in the dark matter component of Box2 mr, using different values for the density contrast parameter. As expected, a higher threshold results in a shift to larger voids. Also the decreasing void number is noticeable: At high density contrasts relative fluctuations increase due to the reduced sample size. However, the accuracy is still sufficient to determine that void diameters are almost exactly log–normally distributed.

Until now, the only property used to describe the voids was their volume, and the equivalent ball diameter derived from it. While this spherical approximation of voids is sufficient for simple size comparisons as in Figure 4.3(c), it does not appropriately reflect the irregular shape of most voids. This is much better captured by ellipsoid fits, which approximate the unhandy real shape of the void by a more manageable one by replacing it with an ellipsoid carrying the same inertia tensor. Since voids are underdense regions, it is not meaningful to consider the actual mass weighted inertia tensor. Instead, we can use the volume weighted analogue

$$I_{ij} = \sum_k V^{(k)} \left(|\mathbf{x}^{(k)} - \mathbf{c}|^2 \delta_{ij} - (\mathbf{x}_i^{(k)} - \mathbf{c}_i)(\mathbf{x}_j^{(k)} - \mathbf{c}_j) \right).$$

Here k runs over all matter particles within the void, $V^{(k)}$ denotes the volume of their Voronoi cells and $\mathbf{x}^{(k)}$ their positions, and $\mathbf{c}$ is the position of the void center, defined by its volume weighted center of mass

$$\mathbf{c} = \sum_k V^{(k)} \mathbf{x}^{(k)}.$$

After calculating this inertia tensor and diagonalizing it, the matching ellipsoid is found to have its half–axes oriented along the eigenvectors of I and their lengths $a \geq b \geq c$ are given by

$$\begin{aligned} a &= \sqrt{\frac{5}{2V}(I_1 + I_2 - I_3)} \\ b &= \sqrt{\frac{5}{2V}(I_3 + I_1 - I_2)} \\ c &= \sqrt{\frac{5}{2V}(I_2 + I_3 - I_1)}, \end{aligned}$$

where V is the total void volume and $I_1 \geq I_2 \geq I_3$ are the eigenvalues of I.

The outcome of this procedure is visualized in Figure 4.6. In addition to the colored void areas and the dark matter particles in the $5\,h^{-1}\,\mathrm{Mpc}$ thick slice around $x_3 = 21.76\,h^{-1}\,\mathrm{Mpc}$, the intersection of the $x_3 = 21.76\,h^{-1}\,\mathrm{Mpc}$ plane with the fitted ellipsoid of each void is

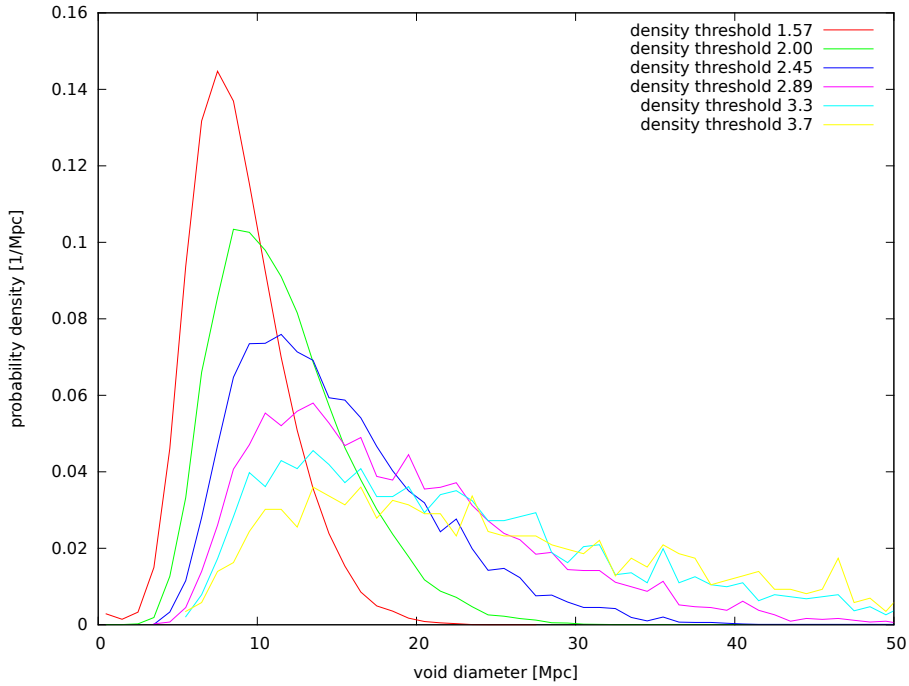


Figure 4.5: The PDF for the equivalent diameters of all voids found in Box2 mr.

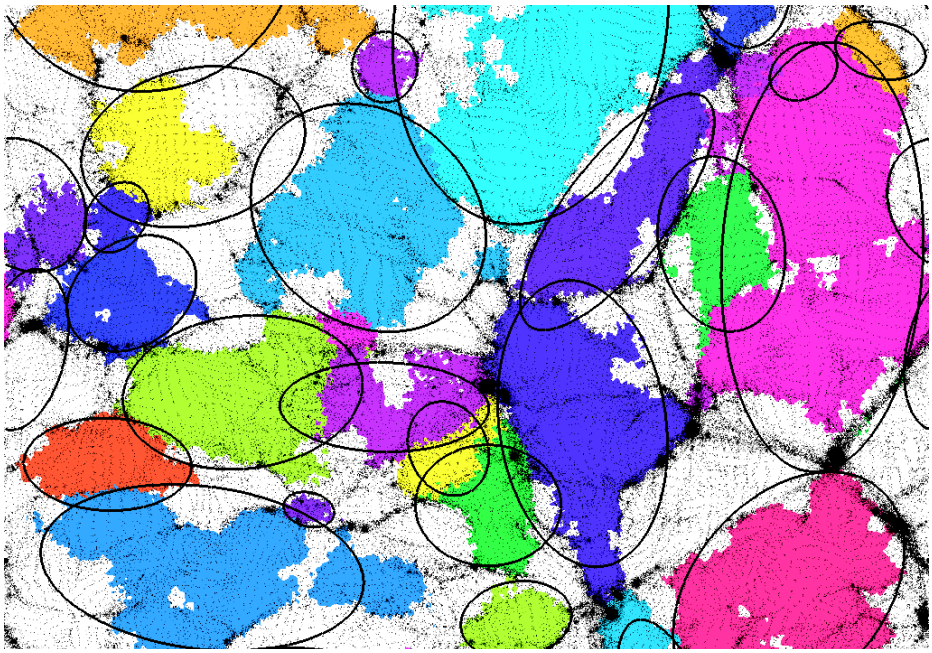


Figure 4.6: A part of the $x_3 = 21.76\,\mathrm{Mpc}$ plane in Box3 mr. The dark matter particles in a 5 Mpc thick slice around the plane are represented by black dots. The intersections of void areas and their fitted ellipsoids with the plane are shown coloured and by black ellipses, respectively.

depicted. In some cases (for example the yellow void in the top left corner of Figure 4.6) the ellipse seems not to match the corresponding void, but this is merely a consequence of the 2–dimensional presentation, whereas the fit respects the whole 3–dimensional void shape. However, most voids match their ellipses surprisingly well even in the 2–dimensional representation.

Though these ellipsoidal fits capture a lot of geometrical information that was lost in the simple volume–preserving spherical fits above, they do not necessarily contain the same volume as the original void shape. In Figure 4.7(a) the real void volume is compared to the volume of the fitted ellipsoid for the set of all voids found in the $z = 0$ snapshot of Box2 mr with a density contrast threshold of 2.45. The green line corresponds to a coincidence of both volumes. A systematic deviation towards a higher volume of the ellipsoid fits (about 50% on average) can be observed. This is expected since far out regions contribute stronger to the inertia tensor than the inner regions, which are dominant in the concentrated ellipsoid shape. This is compensated by a larger volume of the ellipsoid.

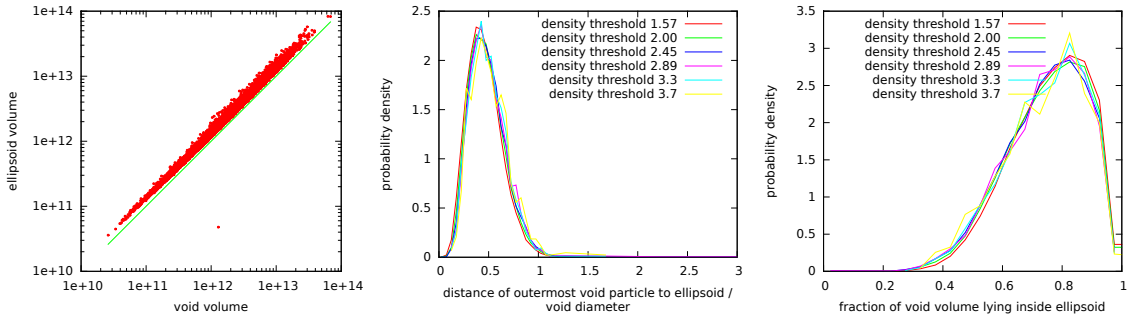


Figure 4.7: Left: Comparison of the void volume and the volume of the fitted ellipsoid. Each dot represents a void. Center: Probability density of the distance of the outermost void particle to the ellipsoid surface, normed by the equivalent void diameter. Right: Probability density of the fraction of the void volume lying inside the corresponding ellipsoid.

For a further assessment of the accuracy of the ellipsoid fits, Figure 4.7(b) displays the maximum expansion of the voids beyond their fitted ellipsoid. For this figure the distance of all dark matter particles lying outside the ellipsoid to the ellipsoid surface was measured. The figure shows the maximum of these distances for each void, divided by the diameter of the void. These distances go above the diameter in extreme cases and typically are in the order of the void radius. This leaves doubt whether ellipsoid fits are suitable for examining such irregular shapes. On the other hand, as Figure 4.7(c) shows, the void volume is mostly covered by the ellipsoid and the extreme deviations represent only little volume.

One of the simplest and probably most interesting properties captured by ellipsoidal fits is the ellipticity, which is defined by $\epsilon := 1 - c/a$, where $a \geq b \geq c$ are the half–axes of the ellipsoid in descending order. An ellipticity of 0 indicates a perfect spherical shape, while larger values show that the fitting ellipsoid is stretched in one or two dimensions. The largest possible value, $\epsilon = 1$, would be attained for the degenerate case of a completely flat void. In [Bos+12], void ellipticities were examined using the *Watershed Void Finder (WVF)* [PvJ07], which is very similar to ZOBOV. It was found that, in a ΛCDM cosmology, the

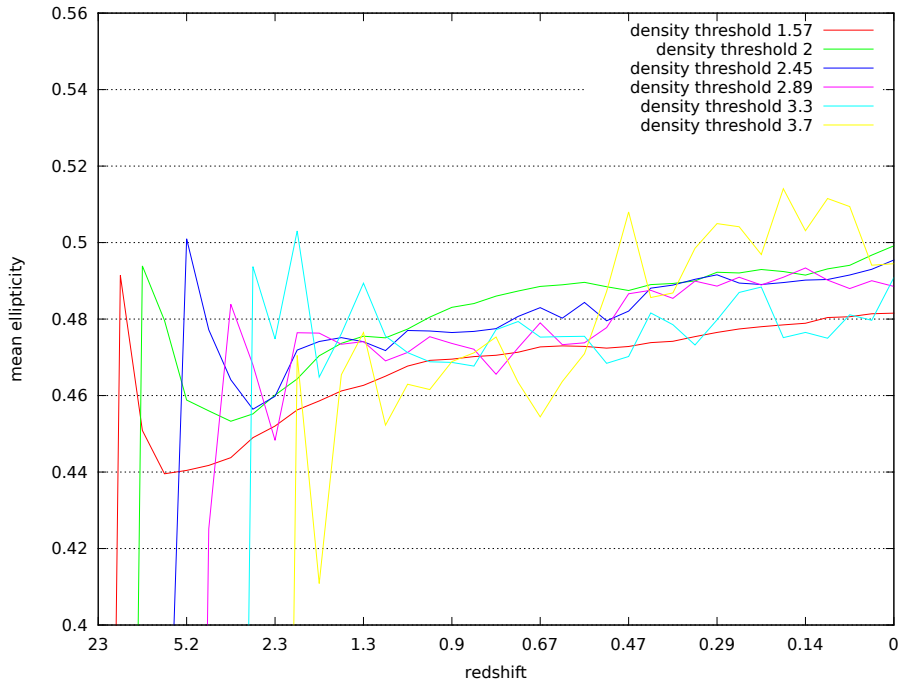


Figure 4.8: The ellipticity $\epsilon = 1 - c/a$ averaged over ellipsoid fits of all voids in a snapshot of Box3 mr. The horizontal axis runs through time while the colors represent different density threshold parameters of the ZOBOV algorithm.

ellipticities increase over time, from a mean value of ≈ 0.44 at $z = 2$ to ≈ 0.455 at $z = 0$, slightly depending on the density contrast parameter.

Figure 4.8 shows the corresponding relation for our voids, using Box3 mr with different density thresholds. An increase over time is clearly visible, regardless of the used density threshold parameter. The ellipticities of ZOBOV's voids are slightly greater than those of the corresponding WVF voids; they have values of 0.465 ± 0.01 at $z = 2$ and 0.49 ± 0.01 at $z = 0$. The erroneous behaviour of Figure 4.8 at early times (high redshifts) is a consequence of the low amplitude of density fluctuations. This causes ZOBOV not to find any voids, but to identify the whole volume as one large 'void', as explained above based on Figure 4.1.

As with the void diameters above, we can also measure the distribution of ellipticities in one particular snapshot, most interesting at $z = 0$. Figure 4.9 provides such a probability density function. In this case the density contrast threshold apparently has almost no effect on the outcome. This is not surprising due to the fact that this parameter can be seen as a determination of the void scale, and ellipticities are invariant under scaling.

Figure 4.10 allows a further distinction between prolate and oblate voids, by showing the ratios b/a and c/b of the half–axes $a \geq b \geq c$, called the *oblateness* and *prolateness* of the ellipsoid, respectively. The scatter plot Figure 4.10(a), which compares prolateness and oblateness of all void ellipsoids in the $z = 0$ snapshot of Box2 mr, suggests them to be approximately independent and identically distributed. However, this is not exactly the case: Figure 4.10(b) compares the probability density functions of the oblateness and the prolateness in the same snapshot, and reveals that c/b peaks at slightly higher values than

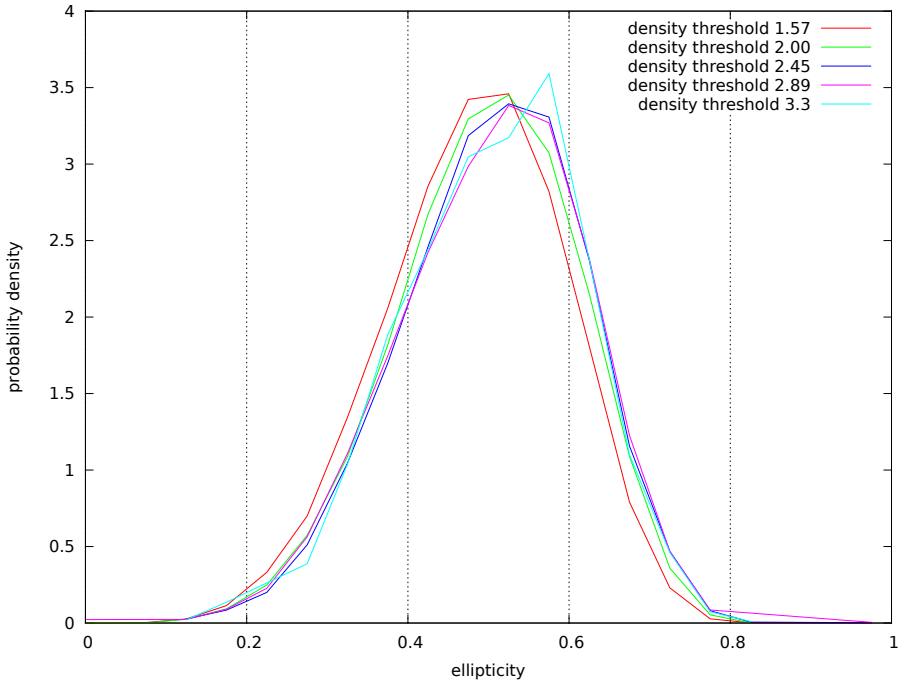


Figure 4.9: The ellipticity PDF for the ellipsoid fits of all voids found in Box2 mr at $z = 0$.

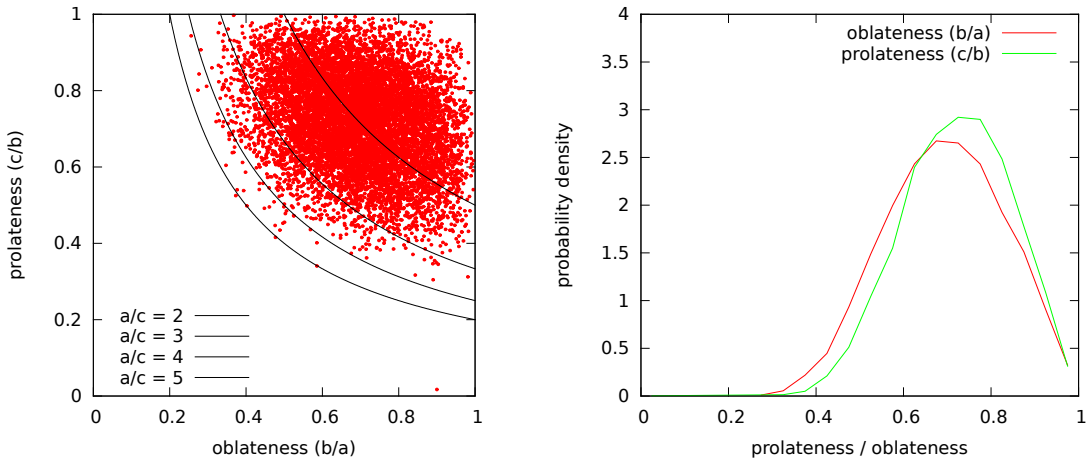


Figure 4.10: The oblateness and prolateness of the ellipsoid fits of all voids found in Box2 mr, $z = 0$, with a density contrast threshold of 2.45

b/a, i.e. the void ellipsoids tend to be prolate.

The curved black lines in Figure 4.10(a) represent values of 2, 3, 4 and 5 for the half–axis ratio a/c, which correspond to the ellipticities 0.5, 0.66, 0.75 and 0.8. We see again that extreme ellipticities are rare: In the sample of 9906 voids found at a density threshold of 2.45 in Box2 mr, the ratios of their half–axes never exceeds 5.

A. The ZOBOV Implementation

In order to apply the ZOBOV algorithm described in [Ney08] to the output of large simulations created by Gadget, it seemed easiest to reimplement it instead of using the reference implementation offered on the ZOBOV website (`http://skysrv.pha.jhu.edu/~neyrinck/voboz/`). The main reasons for this were

1. Convenient support of the Gadget data format without the necessity to convert data before running ZOBOV.
2. Additional support for binary output instead of space separated text files only, which can simplify further processing, especially when handling large amounts of data.
3. Increase in performance when handling big simulations.
4. Some minor bugs in the original implementation.

In the following, I want to describe my implementation, in particular how to use it and which data formats it writes.

A.1. Building

We first need the qhull library (`www.qhull.org`). Download the source code, extract it into a subfolder of the ZOBOV source and build it. It works with the version `2012.1` of qhull, however I have not tested any other version. Next, make sure that the path to the qhull directory is correctly set at the beginning of ZOBOV's `Makefile` and run `make`. This should create two executables, `subvolume_voronoi` and `zobov_voidfinder`.

A.2. Overview

`subvolume_voronoi` is a small program which uses qhull to calculate a Voronoi tessellation out of a snapshot of a simulation output and particularly ensures that the periodic boundaries are handled correctly. It then calculates the volume of each Voronoi cell and a graph of cell adjacencies. Since tessellating the whole simulation output at once would overstrain the memory capacity of most systems, the volume is subdivided into n subvolumes in each direction, resulting in a total of n^3 equally sized cubes. `subvolume_voronoi` must be run for each of these subvolumes separately. The result is saved in files of the form `%s.vol.%02d.%02d.%02d` and `%s.adj.%02d.%02d.%02d`, the numbers indicating the current subvolume. It is advisable to use a shell script to iterate over all subvolumes.

`zobov_voidfinder` only has to be run once. It reads the snapshot files and the files created by `subvolume_voronoi` and applies the ZOBOV algorithm to them. For this purpose, it first combines the files from `subvolume_voronoi`, which are split to subvolumes, into two big files, `%s.vol` and `%s.adj`. Then it estimates the density in each Voronoi cell using its volume and its mass read from the snapshot file. These densities are being saved in `%s.dens`. In the next step, the Voronoi cells are combined to form voids and their properties are measured. This results in 4 output files: `%s.zone`, `%s.voiddata`, `%s.txt` and `%s.void`. The first of these is just a big array which assigns to each particle (or equivalently Voronoi cell) the ID of the zone it belongs to (see subsection 3.2). `%s.voiddata` contains a number of properties for each of

these zones, for example its size, density minimum, minimal border density and the ID of the parent zone. These data can be used afterwards to cut the void hierarchy into distinct voids, as described in subsection 3.2. `%s.txt` and `%s.void` are space separated text files which also hold properties of the zones in a format compatible to the original implementation. However, using them for further processing can be exasperating when handling big simulations.

A.3. Usage

A.3.1. Voronoi Tessellation

Suppose we have a cosmological simulation with 216^3 dark matter particles in a cube of volume $(128\,h^{-1}\,\mathrm{Mpc})^3$, the output of which was saved by Gadget into the directory

```
/Simdata/users/HydroSims/Magneticum/Box3/mr/
```

which contains for each snapshot files of the form `snapdir_%d/snap_%d.%d`, for example the files `snapdir_144/snap_144.0` up to `snapdir_144/snap_144.7` for the 144th snapshot.

The parameters for `subvolume_voronoi` can be seen by running it without any arguments

```
$ ./subvolume_voronoi
Usage: zobov_new <data> <material type (bitmask)> <output prefix>
       <size> <border> <div> <x> <y> <z>
```

The *data* argument specifies the path to the snapshot data. It expects the whole path of the snapshot files except of the trailing dot and number. The *material type* tells the program which types of particles to consider. It is given as a sum of values 2^m where m is a material type ranging from 0 to 5. In all simulations I tried the material types were

0	gas
1	dark matter
4	stars
5	black holes

So if one wanted for example to include only dark matter and black holes, one would have to specify $2^1 + 2^5 = 34$ as material type. The *output prefix* is a path where the output files should be stored. Note that the program does not create any folders, so it will fail if the directory in the specified path does not exist. *size* is the width of the simulation cube in kpc.

border specifies in kpc how far the Voronoi tessellation should extend over the boundary of the subvolume. If this is too small, the algorithm can not guarantee the compatibility of the Voronoi tessellations of the different subvolumes, so it will fail with an appropriate error message. If this happens, just rerun the failed subvolume with a larger border value. If the border width is chosen too big (i.e. in the same order of magnitude as the subvolume size), memory consumption and execution time will suffer. In the above example, 1000 is a good first guess. The details of this mechanism are explained in [NGH05].

div is the number n of subdivisions in each direction. It has turned out that a reasonable choice of the number of subdivisions n ensures that each subvolume contains about a million particles. However, the minimum allowed value is $n = 2$. In our example, this amounts to

$64^3 = 262144$ particles per subvolume, so we go with that choice. *x, y, z* specify the index of the subvolume to be tessellated. They have to be in the range $\{0, \ldots, n-1\}$.

A suitable command line for the above example would be

```
./subvolume_voronoi
    /Simdata/users/HydroSims/Magneticum/Box3/mr/snapdir_144/snap_144 2
    /Simdata/users/HydroSims/Bachelor/fstecker/data/box3_dm_144/box3_dm_144
    128000 1000 2 0 0 0
```

This would create the files `box3_dm_144.vol.00.00.00` and `box3_dm_144.adj.00.00.00` in the directory

```
/Simdata/users/HydroSims/Bachelor/fstecker/data/box3_dm_144
```

provided it exists. Afterwards, the same command has to be rerun another 7 times, replacing the trailing zeroes with any possible combination of the numbers 0 and 1.

A.3.2. Combining Cells to Voids

The arguments for `zobov_voidfinder` can again be seen by executing it without specifying any:

```
$ ./zobov_voidfinder
Usage: ./zobov_voidfinder [voronoi data] [box data] [material type] [div]
```

Voronoi data is the same path specified above for the output of `subvolume_voronoi` and *box data* is the path to the snapshot file as above for `subvolume_voronoi`. *Material type* and *div* are also just copies of the corresponding arguments for `subvolume_voronoi`. The above example then executes as follows:

```
$ ./zobov_voidfinder
    /Simdata/users/HydroSims/Bachelor/fstecker/data/box3_dm_144/box3_dm_144
    /Simdata/users/HydroSims/Magneticum/Box3/mr/snapdir_144/snap_144 2 2
Reading volumes Done.
Calculating densities Done.
Initializing cells Done.
Finding minimal neighbors Done.
nzones = 28360
Jumping Done.
Sorting zone list Done.
Post-Jumping Done.
Building zone adjacency list Done.
nzoneadjs = 507333
Sorting adjacency list Done.
Joining zones to voids Done.
Deleting temporary file Done.
Correcting void tree Done.
Measuring voids Done.
Writing output Done.
```

A.4. File Formats

As indicated in subsection A.2, the program `zobov_voidfinder` writes 4 output files with the endings `void`, `txt`, `voiddata` and `zone`. It also combines the output of `subvolume_voronoi` to additional files with endings `vol` and `dens`.

A.4.1. Indices

Since the particle indices used by Gadget are not continuous, our ZOBOV tools number the particles in a different order. First they are ordered by file number, then by particle type (where all unused particle types are also not considered in the numbering), next by the order they appear in the snapshot file. For example, if a snapshot consisted of 8 files and one used dark matter and star particles, the particles would be indexed in the following order: Dark matter particles from file 0, star particles from file 0, dark matter particles from file 1, star particles from file 1, and so on.

A.4.2. The `vol` and `dens` Files

These files contain the volume and estimated density (particle mass divided by volume) of each Voronoi cell, arranged as a big array of 32 bit floats, indexed as described in subsubsection A.4.1. The units are as in the snapshot files, usually kpc^3 and $M_\odot/\mathrm{pc}^3$.

A.4.3. The `txt` File

The `txt` file is a space–separated plain text file containing 11 fields for each of the 'zones':

1. A consecutive number.
2. The 'Zone ID'.
3. The index (in the sense of subsubsection A.4.1) of the minimum density particle.
4. The minimum density of all particles / Voronoi cells contained in the zone.
5. The zone volume, i.e. the sum of all contained Voronoi cell volumes.
6. The number of particles contained in the zone.
7. The number of zones in the subtree which has the current zone as root.
8. The total volume of the subtree which has the current zone as root.
9. The number of particles in the subtree which has the current zone as root.
10. The 'density contrast' as described in subsection 3.2.
11. The void error probability as described in [Ney08].

The rows in the `txt` files are sorted by their density contrasts in descending order.

A.4.4. The `zone` File

The zone file is an array of 64 bit integers, indexed by particle indices (see subsubsection A.4.1) which assigns to each particle the ID of the zone it is contained in.

A.4.5. The `voiddata` File

The `voiddata` file is a binary file which contains a much more detailed version of the ZOBOV results than the `txt` file. It is an array of `zone_t` structures, one for each zone, which have the following layout:

```
typedef struct {
  long partid;     // ID of minimum density particle (in the sense of A.4.1)
  long zoneid;     // zone ID
  long parent;     // index of parent zone in the voiddata file
  long next;       // index of next sibling zone in the voiddata file
  long children;   // index of first child zone in the voiddata file
  long nchildren;  // number of child zones
  double min;      // minimum density
  double sld;      // smallest linking density
  double zonevol;  // zone volume
  long nzonepart;  // number of particles in zone
  double voidvol;  // if zone is a void core, the volume of the void
  long nvoidpart;  // if zone is a void core, nr. of particles in the void
  long nvoidzones; // if zone is a void core, nr. of zones in the void
} zone_t;
```

The last three elements are not filled with valid information in the file, since these quantities are not calculated by `zobov_voidfinder` but rather in a following postprocessing step. The `voiddata` file also serves as input for `postproc.c`, a small post–processing tool capable of dissecting the void tree into disjoint void regions based on a density contrast threshold.

A.5. Postprocessing

The often needed step of dissecting the void tree into distinct voids is not part of the `zobov_voidfinder` program. This separation is a key feature of ZOBOV, as it facilitates playing with the parameters by not having to rerun the time–consuming main program. The file `postproc.c` is a small collection of functions which do this postprocessing step and provide some useful tools for handling the data. The sample `test.c` demonstrates its usage. After compiling it using

```
gcc -std=gnu99 -O3 -lm -o test test.c postproc.c
```

it provides a simple consistency check of the ZOBOV output by computing minimum void diameters and void volume fractions as in Figure 4.3. `postproc.c` provides the following functions, defined in `postproc.h`:

```
void load_voids(const char *voidfile,
                double mindenscontrast, double maxdens,
                zone_t **zones_out, long *nzones_out,
                zone_t ***voids_out, long *nvoids_out);
```

The `voiddata` file to be processed has to be specified as the first argument. The second and third argument are the desired density contrast threshold and the maximum core density to require for a significant void. The density is specified in the typical comoving Gadget units. In the case of Magneticum Pathfinder at $z = 0$, that is in $10^{10}\,\mathrm{h}^2\,\mathrm{M}_\odot/\mathrm{kpc}^3$. If we do not want to specify a maximum core density, we can just use the constant `INFINITY`. The arguments `zones_out` and `nzones_out` require pointers to variables to store the list of zones and its length in. The `voids_out` and `nvoids_out` arguments analogously return the sublist of zones which are cores of significant voids. More precisely, the triple pointer `voids_out` points to a variable of type `zone_t**`, which `load_voids` fills with a pointer to the first element of an array of `zone_t*` type variables, which are pointers to entries of the zone list. Both the list of zones and the list of voids are allocated using `malloc`, so if not used any more they have to be freed using:

```
free(*voids_out);
free(*zones_out);
```

Note that the returned objects of type `zone_t` do have valid last three entries. Also, the elements `parent`, `next` and `children` are now pointers of type `zone_t*`, pointing to the respective zone objects in the returned array. The correct `zone_t` structure as used by `load_voids` is defined in `postproc.h`.

For access to the `dens`, `vol` and `zone` files, `postproc.c` additionally defines the following four functions:

```
int open_zonemap(const char *mapfile, uint64_t **map_out, long *maplen_out);
void close_zonemap(int file, uint64_t *map, long maplen);
int open_volumemap(const char *mapfile, float **map_out, long *maplen_out);
void close_volumemap(int fd, float *map, long maplen);
```

These take as first arguments the path of the `zone` file (in the case of `open_zonemap`) and the `dens` or `vol` file (in the case of `open_volumemap`). The argument `map_out` points to a variable which is filled with a pointer to the first element of the desired array of indices, densities or volumes. The last argument `maplen_out` points to a variable which the function fills with the length of this array. The return value of `open_zonemap` and `open_volumemap` is an integer file descriptor, which should be saved since it is required by the close functions. These close functions have as arguments the just mentioned file descriptor, the data array and the length of the map (exactly the dereferenced versions of arguments 2 and 3 of the open functions). These functions do not read the whole file into memory, but rather use the `mmap` mechanism to transparently load only the necessary portions.

References

[Aba+09] K. N. Abazajian et al. "The Seventh Data Release of the Sloan Digital Sky Survey". In: *The Astrophysical Journal Supplement* 182, 543 (June 2009), pp. 543–558. arXiv: 0812.0649.

[Abe58] G. O. Abell. "The Distribution of Rich Clusters of Galaxies." In: *The Astrophysical Journal Supplement* 3 (May 1958), p. 211.

[Ang+12] R. E. Angulo et al. "Scaling relations for galaxy clusters in the Millennium-XXL simulation". In: *Monthly Notices of the Royal Astronomical Society* 426 (Nov. 2012), pp. 2046–2062. arXiv: 1203.3216 [astro-ph.CO].

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. "The Quickhull algorithm for convex hulls". In: *ACM transactions on mathematical software* 22.4 (1996), pp. 469–483.

[Bos+12] E. G. P. Bos et al. "The darkness that shaped the void: dark energy and cosmic voids". In: *Monthly Notices of the Royal Astronomical Society* 426 (Oct. 2012), pp. 440–461. arXiv: 1205.4238 [astro-ph.CO].

[BT11] J. Binney and S. Tremaine. *Galactic Dynamics: (Second Edition)*. Princeton Series in Astrophysics. Princeton University Press, 2011.

[Col+08] J. M. Colberg et al. "The Aspen-Amsterdam void finder comparison project". In: *Monthly Notices of the Royal Astronomical Society* 387 (June 2008), pp. 933–944. arXiv: 0803.0918.

[Ney08] M. C. Neyrinck. "ZOBOV: a parameter-free void-finding algorithm". In: *Monthly Notices of the Royal Astronomical Society* 386 (June 2008), pp. 2101–2109. arXiv: 0712.3049.

[NGH05] M. C. Neyrinck, N. Y. Gnedin, and A. J. S. Hamilton. "VOBOZ: an almost-parameter-free halo-finding algorithm". In: *Monthly Notices of the Royal Astronomical Society* 356 (Feb. 2005), pp. 1222–1232. arXiv: astro-ph/0402346.

[NH13] S. Nadathur and S. Hotchkiss. "A self-consistent public catalogue of voids and superclusters in the SDSS Data Release 7 galaxy surveys". In: *ArXiv e-prints* (Oct. 2013). arXiv: 1310.2791 [astro-ph.CO].

[ONe83] B. O'Neill. *Semi-Riemannian Geometry With Applications to Relativity, 103*. Pure and Applied Mathematics. Elsevier Science, 1983.

[PS88] F.P. Preparata and M.I. Shamos. *Computational geometry: an introduction*. Texts and monographs in computer science. Springer-Verlag, 1988.

[PvJ07] E. Platen, R. van de Weygaert, and B. J. T. Jones. "A cosmic watershed: the WVF void detection technique". In: *Monthly Notices of the Royal Astronomical Society* 380 (Sept. 2007), pp. 551–570. arXiv: 0706.2788.

[Sch+13] A.T.P. Schauer et al. *The mystery of the σ–bump—a new signature for major mergers in elliptical galaxies?* 2013. arXiv: 1312.1337. to appear in ApJ Letters.

[SH03] V. Springel and L. Hernquist. "Cosmological smoothed particle hydrodynamics simulations: a hybrid multiphase model for star formation". In: *Monthly Notices of the Royal Astronomical Society* 339 (Feb. 2003), pp. 289–311. arXiv: `astro-ph/0206393`.

[Spr05] V. Springel. "The cosmological simulation code GADGET-2". In: *Monthly Notices of the Royal Astronomical Society* 364 (Dec. 2005), pp. 1105–1134. arXiv: `astro-ph/0505010`.

[Sut+12] P. M. Sutter et al. "A Public Void Catalog from the SDSS DR7 Galaxy Redshift Surveys Based on the Watershed Transform". In: *The Astrophysical Journal* 761, 44 (Dec. 2012), p. 44. arXiv: `1207.2524`.

Declaration of Authorship

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

Florian Stecker
February 3, 2014