Laniakea in a Cosmological Context

or

Detection of galaxy superclusters in simulated cosmological structures

Sergio Daniel Hernández Charpak
200922618

Advisor: Jaime E. Forero-Romero

May 14, 2016

Universidad de los Andes
Facultad de Ciencias, Departamento de Física
Bogotá, Colombia

1 Acknowledgements

I would like to thank my advisor, Jaime E. Forero-Romero, for his guidance during this process and of course my family, Nathalie, Jose Tiberio, Yván David, Gabriel Luis, Nico, Dani and all the others for their great patience and always good advices. I also acknowledge the University of Los Andes for its mayor financial support during my studies. Finally I couldn't have finish this work without the support of Andrés, Nicolás, Jorge and all the special friends I am lucky to have. Georges would be very happy.

2 Abstract

Recently Tully et al. (2014) [1] used local cosmic flow information to define our local supercluster, Laniakea. In this work we present a study on large cosmological N-body simulations aimed at establishing the significance of Laniakea in a cosmological context. We explore different algorithms to define superclusters from the dark matter velocity field in the simulation. We summarize the properties of the supercluster population by their abundance at a given total mass and its shape distributions. We find that superclusters similar in size and structure to Laniakea are relatively uncommon on a broader cosmological context. We finalize by discussing the possible sources of systematics (both in our methods and in observations) leading to this discrepancy.

Contents

3 Motivation

We live in the planet Earth. Our planet orbits around the Sun in the Solar System, under the influence of the gravitational forces. Our Solar System forms, along with many other stars and systems, our home galaxy, the Milky Way. Galaxy themselves, under the same forces, form even larger structures. It is these structures we call galaxy superclusters.

Detecting these galaxy superclusters is a challenging task. Observationally there are There exists different algorithms to identify them based on physical criteria [2]. One possible physical criterion relates the velocity flows with the density.

Indeed, due to gravitational attraction, it is possible to relate velocity fields with density fields. A proposal to find galaxy superclusters is to analyze the velocity field of galaxies as matter tend to flow to dense region. Spacial regions where the galaxy flow is convergent represent galaxy superclusters.

Recently a team of astronomers built a galaxy velocities flow map of the local universe in a scale of hundreds of million light years [3]. In this map, the team found converging velocity flows to identify identified Laniakea, the galaxy supercluster which includes our galaxy, the Milky Way [1]. These observations are difficult. As a result of these difficulties, it is not possible to build a statistical sample superclusters obtained from the observational approach.

We propose a method to detect galaxy superclusters in cosmological simulations following physical criteria similar to the one used to define Laniakea. After an interpolation of the velocity and density fields we apply a region growing algorithm based in predefined parameters. We test our method on N-body simulations and successfully detect structure we compare to Laniakea.
We aim to quantify if Laniakea can be considered as an atypical structure in the Universe.

The definition of Laniakea by Tully et al [1] is based on the peculiar velocity flow. Chon et al provide an interesting work on a different definition of galaxy superclusters, a gravitationally bound dense large scale structure [4] and [5]. On this different criteria, Laniakea and how Laniakea fits their definition for supercluster differs.

4 Introduction

4.1 The Standard Cosmological Model

The current standard cosmological model is known the Lambda - Cold Matter Model, or Λ CDM Model. It assumes Einstein's equations of General Relativity.[6].
The letter Λ makes a reference to the cosmological constant Ω_Λ, which is associated with dark energy and the accelerated expansion of the Universe according to General Relativity. Dark energy is a fairly unknown form of energy. It is proposed to explain the accelerated rate of expansion of the Universe [7].

The dark matter is a hypothetical type of matter which is nor ordinary matter (baryonic matter), nor neutrino particles nor dark energy. It is dark as it does not seem to produce any radiation (light). There has been indirect observations of dark matter as difference between gravitational matter implied by velocities of stars and observed matter and gravitational lensing between others [8]. Cold means that the velocity of the hypothetical fundamental particles conforming the dark matter are not relativistic. A consequence of this is a bottom-up scheme. With bottom-up, we mean that small structures merge into one larger structure. The opposite would be hot matter, meaning that the hypothetical fundamental particles could achieve relativistic velocities. In this hot matter model, there is a top-down scheme, where an initial large structure breaks into smaller structures. [6]

The simplest Λ-CDM Model is based on the following 6 general parameters.

1. The Hubble parameter, H, defined as the time dependent Hubble parameter

$$H = \frac{\dot{a}}{a},$$

where $a(t)$ refers to the scale factor of the Universe. For nearby galaxies, H is found to be constant and thus is defined as the Hubble constant H_0.

$$H_0 = 100\ h\ km\ s^{-1}\ Mpc^{-1} = 67 \pm 7 km\ s^{-1}\ Mpc^{-1}.$$

Mpc is a cosmological length unit. $1Mpc = 10^6\ pc$ where $1pc = 3.087 \times 10^{16} m$. h is the dimensionless parameter defined by the equation above, $H_0 = 1\ km\ s^{-1}\ Mpc^{-1}\ h^{-1}$. In this document the length units are Mpc/h.

2. Critical density ρ_{cr} found when the cosmological constant is 0 in Einstein's equations, condition needed to have a close universe. $\rho_{cr} = \frac{3H^2}{8\pi G} = (8.62 \pm 0.12) \times 10^{27} kg/m^3$.

3. Baryon density parameter Ω_b. The density of baryonic matter expressed as a fraction of the critical density. $\Omega_b = \frac{\rho_b}{\rho_{cr}} = 0.0486 \pm 0.0010$.

4. Dark matter density parameter Ω_c. The density of dark matter expressed as a fraction of the critical density.$\Omega_c = \frac{\rho_c}{\rho_{cr}} = 0.2589 \pm 0.0057$.

5. Matter density parameter Ω_m or Ω_0. It corresponds to the sum of Ω_c and Ω_b, $\Omega_0 = 0.3089 \pm 0.0062$

6. Dark energy density parameter, cosmological vacuum energy density (or cosmological constant), Ω_Λ. As we stated before, it is associated to the accelerated rate of expansion of the Universe. $\Omega_\Lambda = 0.6911 \pm 0.0062$

The total density parameter, a constraint, is the sum of the density parameters and has to be equal to one. $\Omega_{tot} = \Omega_b + \Omega_c + \Omega_\Lambda = 1$ or $\Omega_{tot} = \Omega_0 + \Omega_\Lambda = 1$.

Another important concept is the redshift. The redshift z can be defined by the following equation[7]:

$$1 + z = \frac{\lambda_{obs}}{\lambda_{em}}.$$

It is a measure of the difference between the observed wavelength λ_{obs} and the wavelength received from a distant object λ_{em}. It is a convenient way to label time, with $z = 0$ being the present time.

4.2 Cosmic flows and peculiar velocities

Our method to detect superclusters in cosmological simulations is based on the peculiar velocity field analysis in a similar fashion as Tully et al. did to find Laniakea [1] [9].

- Basic theoretical considerations

The peculiar velocity is defined as the velocity relative to a rest frame. Its radial component corresponds to the velocity of a galaxy that cannot be explained by the expansion of the Universe. Hubble's law states that nearby galaxies move away from us at a speed proportional to the distance from us (the proportion is Hubble's constant). In other words,

$$v_p = v_{obs} - H_0 \times d_{obs}.$$

With v_p the peculiar velocity, v_{obs} and d_{obs} the observed velocity and distance, H_0 the Hubble constant as described in section 4.1.

From the linear perturbations in the Newtonian limit of the ΛCDM model presented in section 4.1 There is a linear relation between the peculiar velocity and the peculiar acceleration [10].

$$\vec{v_p} \simeq \frac{2}{3H\Omega_b}\Omega_b^{0.6}\vec{g}.$$

With H the Hubble parameter, Ω_b the baryonic density parameter and $\vec{g}$ the peculiar acceleration with $\vec{g} = \frac{-1}{a}\nabla\phi$ with ϕ the gravitational potential and a the scale factor of

the universe.
In other words we obtain an expression:

$$\vec{v_p} \propto -\nabla\phi.$$

If we take the divergence we obtain:

$$\nabla \cdot \vec{v_p} \propto -\nabla^2\phi.$$

But $\nabla^2\phi$ responds a Poisson-like equation with the density field ρ. We can finally write:

$$\nabla \cdot \vec{v_p} \propto -\rho.$$

We define the overdensity δ as the excess density in relation with the mean density of the universe. $\delta = \rho/\bar{\rho} - 1$ with $\bar{\rho}$ the mean density. δ is an adimensional quantity.

Our equation for peculiar velocity can be written as [1]:

$$\nabla \cdot \vec{v_p} = -H_0 f(\Omega_m, \Omega_\Lambda)\ \delta.$$

With $\vec{v_p}$ the peculiar velocity field, H_0 the Hubble constant, Ω_m the matter density parameter, Ω_Λ the Dark matter density parameter, δ the excess density field and $f(\Omega_m, \Omega_\Lambda)$ a linear function of Ω_m and Ω_Λ.
With this relation we can directly link the peculiar velocity field and the excess density field. They will have related behaviours.

- Observational methods

We now proceed to describe some of the observational methods used Cosmicflows-2 [3]. To define Lanieakea, the catalogue of peculiar velocities is detailed within 100 Mpc with data covering out to 400 Mpc. Six main methods were jointly used to estimate the galaxy distances.

1. Cepheid Period-Luminosity Relation (Cepheid PLR) applied to the Hubble Space Telescope (HST) distance scale key project. The HST data is covers distances within 10 Mpc. Their observation can be difficult due to luminosities of neighbouring objects.

2. Tip of Red Giant Branch (TRGB). Red Giants are stars fainter than the Cepheids but are located in regions with small luminosity noise. This method, complementary to the Cepheid PRL method covers distances within 10 Mpc.

3. Surface Brightness Fluctuation (SBF) method measures the variance in the distribution of light emitted in a galaxy. The noisy fluctuations we observe are proportional to the distance of the galaxy. This method covers distances within 100 Mpc.

4. Elliptical fundamental plane. The quality of these measurements is not very good. Despite of that, it can be used for large samples of galaxies in a large volume.

5. Spiral luminosity-rotation. This method also provides less accurate distances, but can be applied to large sets of galaxies.

6. Supernovae Ia method. These supernovas are very bright and can been seen from great distances. The quality of these measurements is good and can cover large distances. Nevertheless, samples of these supernovas are small.

Finally, after the observation were made, a Wiener Filter was applied. A Wiener filtering method is a tool to reconstruct large scale structure information from some of the incomplete and noisy data [11]. It was applied by Tully et al. [1] to to obtained a better Signal/Noise proportion in the peculiar velocity field from the Cosmicflows-2 observations.

4.3 The V-Web Algorithm

The contour of the region of Laniakea were reconstructed using the V-web Algorithm [1]. The basis of our method for the detection superclusters in cosmological simulation is the V-Web Algorithm, first introduced by Hoffman et al [12].

This algorithm is deeply connected to the velocities. Its core is the shear velocity tensor defined as:

$$\Sigma_{\alpha\beta} = -\frac{1}{2H_0}\left(\frac{\partial v_\alpha}{\partial x_\beta} + \frac{\partial v_\beta}{\partial x_\alpha}\right),$$

With v_α and x_α representing the α component the comoving peculiar velocity and the comoving position, respectively.

$\Sigma_{\alpha\beta}$ can be represented by a 3×3 symmetric matrix with real values. This makes it possible to diagonalize it and find its 3 eigenvalues.
These eigenvalues λ_1, λ_2 and λ_3 carry information about the motion of the fluid. By convention they are ordered such as $\lambda_1 > \lambda_2 > \lambda_3$. The V-Web scheme refers to the web classification based on how many of these eigenvalues are above a predefined threshold.

Respectively if 0, 1, 2 or 3 eigenvalues are above the threshold the point is classified as a void, sheet, filament or knot.

Our method uses indirectly these eigenvalues through the Fractional Anisotropy and the trace of $\Sigma_{\alpha\beta}$.

4.4 Fractional Anisotropy (FA) and Velocity Divergence (VDH)

Our method uses on the Fractional Anisotropy (FA) and the trace of the Velocity Shear Tensor.

The Fractional Anisotropy (FA) was first conceived in the medical context by Peter Basser [13]. Libeskind et al [14] first introduced the FA in the cosmological context. It has been used as a tracer of cosmic voids by Bustamante et al. [15].

The FA can be written as:

$$FA = \frac{1}{\sqrt{3}}\sqrt{\frac{((\lambda_1-\lambda_3)^2+(\lambda_2-\lambda_3)^2+(\lambda_1-\lambda_2)^2)}{\lambda_1^2+\lambda_2^2+\lambda_3^2}},$$

With λ_1, λ_2 and λ_3 the eigenvalues obtained from the V-Web algorithm. The FA is dimensionless and takes values between 0 and 1. We interpret FA = 0 as an isotropic velocity distribution (where $\lambda_1 = \lambda_2 = \lambda_3$) and FA = 1 as a highly anisotropic velocity distribution.

- The trace of the Velocity Shear Tensor (VDH)

We describe in section 4.2 the divergence of the peculiar velocity field is proportional to the overdensity:

$$\nabla \cdot \vec{v}_p \propto -\delta$$

The trace of the velocity shear tensor $\Sigma_{\alpha\beta}$, $\lambda_1 + \lambda_2 + \lambda_3$ is equal to the divergence of the velocity divided by a factor H_0, the Hubble constant.

$$\lambda_1 + \lambda_2 + \lambda_3 = \frac{-\nabla \cdot \vec{v}}{H_0}.$$

This is an dimensionless quantity, proportional to the overdensity δ.

We name the trace of $\Sigma_{\alpha\beta}$ or $\frac{-\nabla\cdot\vec{v}}{H_0}$ as VDH which states as Velocity Divergence divided by the Hubble constant.

In the next section we observe how we use the FA and VDH to find superclusters in cosmological simulations.

5 Methods

We run our own cosmological N-body simulations. We proceed to compute the FA and VDH from the simulations data in a series of intermediate steps. Finally, our algorithm uses the FA and VDH computed from N-body simulations to detect superclusters.

Our code and algorithm are available for the public in Github `https://github.com/sercharpak/Monografia/`

5.1 Cosmological Simulations

To make our own cosmological simulations we use the N-Body simulation software Gadget-2 [16] widely used in the scientific community to generate different cubic boxes of dark matter particles (DM).

By DM particle we mean a computational particle which only interacts through gravity interaction. In the simulation each particle represents a fluid element. The DM particles are placed in a 3D grid layout first and then are perturbed following a Gaussian distribution before the simulation start, during the initial conditions generation. Once the simulation starts, the particles interact only with gravity. Gadget-2 simulates this fluid. The simulations ran on the HPC cluster at UNIANDES. The main properties (boxsize, number of particles, number of CPUs used and time of the simulation) are described in table 1.

We are looking for structures of the same order of magnitude of Laniakea (radius of 80 Mpc/h). Our simulations need to be of at least the same order of magnitude. We also need a sufficient number of DM particles to be able to describe the density field on the smaller scales probed by observations (around 5 Mpc/h). From these constraints we decided to run three simulations as listed in table 1.

Name	Boxsize [Mpc/h]	DM particles	# CPU	Time [hours]
Medium	500	512^3	48	10
Small (Dense)	250	512^3	48	6
Medium (Dense)	500	1024^3	48	80

Table 1: Different N-body Simulations

The initial conditions were generated using Volker Springel's N-Genic software. For the initial conditions we specify the cosmological constants described in section 4.1. They are described in table 2

.

Ω_0	Ω_Λ	Ω_b	H
0.3	0.7	0	0.7

Table 2: Cosmological Constants in the N-body Simulations

As we specify in section 4.1, Ω_0 corresponds to the Cosmological matter density parameter in units of the critical density at red shift z = 0. Ω_Λ is the cosmological vacuum energy density (cosmological constant) in units of the critical density at red shift z = 0 (present). For the standard cosmological model, $\Omega_0 + \Omega_\Lambda = 1$. Ω_b is the baryon density in units of the critical density at red shift z = 0. Finally, H denotes the Hubble constant at red shift z=0 in units of 100 $km\ s^{-1}\ Mpc^{-1}$.

We then use different algorithms to identify the superclusters within the simulation.

5.2 Intermediate Steps

Our algorithm uses the FA and VDH computed from the N- body simulations[1]. There are three steps involved before. First, the interpolation of the particle data informations into a grid. Second, the V-web computation on this grid. Finally, the computation of the FA and VDH on this grid.

- The Cloud In Cell (CIC) interpolation method

The Cloud In Cell (CIC) is a method to interpolate a quantity such as the density in a grid. The (CIC) method is a derivation of the Particle in Cell (PIC) method, a Particle Mesh (PM) method, first introduced by Francis Harlow in the 1950's in the field of computational fluid dynamics [18]. CIC is very popular among the scientific community [19]. We form, in our case, a N^3 grid. This grid is superpose with the cubic box resulting from the simulation. We then form particle clouds (cubes in our case) around each particle. For each particle cloud, each cell where the particle cloud superposes is assigned a weighted part of the physical quantity we look to mesh proportional to the superposed volume (in 2-D, superposed area). A function can then be applied to smooth the obtained field. We perform this CIC for the density field and the velocity field. In figure 1 a graphical explanation is provided for the 2-D case.

[1] We also explored an approach of the magnitude of the velocity in the different simulations. We work with all the particles and points given by the simulation. As our final results are not based in this naive approach we leave it as an appendix here. It is explained in further detail in Appendix A.1

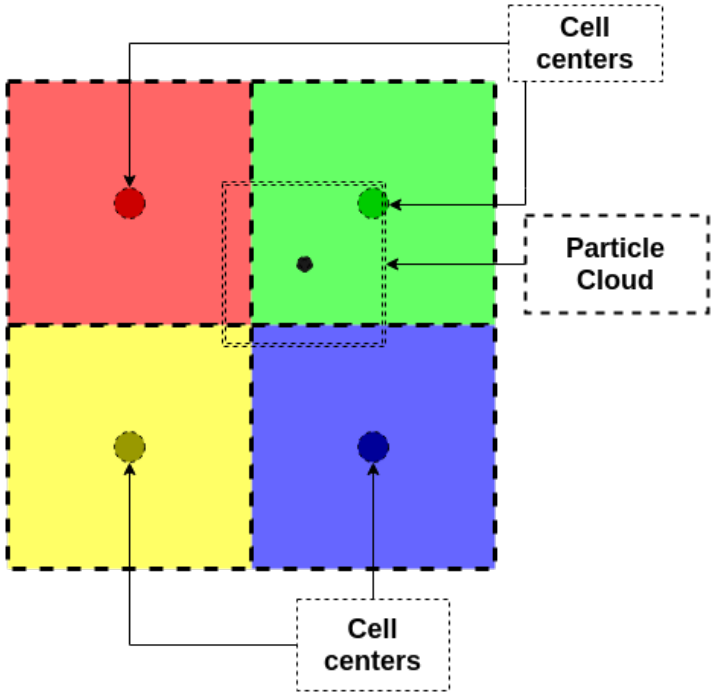

Figure 1: Visualization of the Cloud In Cell Method (CIC)

- The V-Web scheme, the FA and the VDH

We describe the V-Web scheme in section 4.3 and the FA and VDH in section 4.4. The FA gives us a dimensionless quantity to analyse isotropy and the VDH gives us a also another dimensionless quantity which behaves in a similar fashion as the matter overdensity δ.

In summary, our approach follows the steps:

1. Calculate the density field using CIC on the simulation data.
2. Calculate the velocity field using CIC on the simulation data.
3. Use finite differences to compute the velocity shear tensor $\sum_{\alpha\beta}$.
4. Diagonalize $\sum_{\alpha\beta}$ and find its eigenvalues λ_i and eigenvectors $\hat{u}_i$
5. Compute the FA and VDH grids as described in section 4.4
6. Apply the algorithms described in section 5.3 to find the superclusters
7. Extract the properties (volume, shape, mass) from the superclusters
8. Compare the superclusters detected in the simulation to Laniakea

This methodology is very close with the approach used to define Laniakea [1] which is important to validate our later comparison of Laniakea with our detected structures.

5.3 Superclusters Finding Algorithms

Our main algorithm is a region growing algorithm as described by Gonzalez [20] mainly used in image segmentation.

5.3.1 Region Growing Algorithm: Basic Description

The region growing algorithm is used to identify regions sharing common properties. First we identify points, as few as possible, which we know for sure are contained in these regions. These points are the seeds of our regions. We then define a certain growth predicate, if the neighbours of a seed validate this predicate they are considered as members of the group. These new members are then treated as seeds and ask their own neighbours if they validate the predicate. This goes on until all the points have been evaluated or until a limit defined by the user is reached.

We can write in a more formal way:

$f(x, y, z)$ denotes the input data. $S(x, y, z)$ denotes a seed array, with value 1 where the particle can be define as a seed and 0 where not. $S(x, y, z)$ is the same size of $f(x, y, z)$. Q denotes a predicate which is to be apply to the input data and determines if the region which starts at the seeds grows or not.

1. We search for seeds from $f(x, y, z)$ based in a criterion
2. For each detected seed, we change the values of $S(x, y, z)$ to 1 at the seed's coordinates.
3. For each detected seed, we iterate its neighbors if they validate the predicate Q.
4. If the particle c at x_c, y_c, z_c satisfies Q, it is marked as part of the region, $S(x_c, y_c, z_c) = 1$
5. Apply the algorithm to c, and so on (Recursion).

The algorithm is based on a connectivity concept (definition of neighbours). Here we are working on a 4-connectivity. 4 connectivity means that no diagonal particles are considered as neighbours. The following matrix shows this principle in a 2 dimensional case. Only the elements labelled as 1 are considered as neighbours of the center C.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & C & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

So, if a particle is in coordinates (x, y, z) its neighbours are $(x-1, y, z)$, $(x+1, y, z)$, $(x, y-1, z)$, $(x, y+1, z)$, $(x, y, z-1)$ and $(x, y, z+1)$.

Another widely used connectivity is the 8-connectivity. As it is intuitive, in 8-connectivity, the diagonal particles are also considered as neighbours. We decided to use 4-connectivity to simplify our own implementation.

5.3.2 Region Growing Algorithm with the FA and the trace of the velocity shear tensor

Our implementation is based in the FA and the VDH. It is uses 4-connectivity as described before.[2]

- Seeds

  We are looking for overdense structures. Therefore we will look for particles with $VDH \geq 1.0$. We also want to analyse how the FA fits. We know that the centres of our structures should be fairly isotropic so we will explore seeds with FA lesser than a threshold. We will perform several runs of the algorithm, so we can explore the different criterion for the choice of the seeds. These can be seen in table 3.

FA_{seed}	VDH_{seed}
0.5	1.0
0.6	1.0
0.7	1.0
0.8	1.0
0.9	1.0

Table 3: Different values of the FA and VDH for the choice of the seeds

- Labelling the regions

  We label groups as a function of its respective seed to extract properties (volume, shape, mass) for each one of them.

- Predicate Q

  We define, as we did for the choice of the seeds, thresholds for the FA and for VDH. If the neighbour cell validates these thresholds, and hasn't been labelled yet, it is labelled as the group of the seed particle. So in other words:

$$Q : (FA(x, y, z) \leq Thresh_{FA}) \text{ AND } (VDH(x, y, z) \geq Thresh_{VDH})$$

  As we are looking for overdense regions (no voids) we define the threshold for VDH as 0.0. As we are exploring the different values for the FA. each run will follow a row of the table 4.

[2] A simplified version code can be seen in appendix A.2.1. Results of this implementation can be seen in appendix A.2.2.

FA_{growth}	VDH_{growth}
0.5	0.0
0.6	0.0
0.7	0.0
0.8	0.0
0.9	0.0

Table 4: Different values of the FA and VDH for the growth of the regions

Finally to find the groups we used the Friends-of-Friends (FoF) algorithm.

The FoF algorithm looks at the distances between points in a given set. If the distance between a point a and a point b is less than a predefined threshold, a and b now form a region. For a region to be created by the FoF algorithm it also needs to have a predefined minimum number of members.

We choose as set of points all the particles which validate the predicate Q (following table 4, as maximum distance the distance between one particle and its neighbours as described by 4-connectivity. And we use the FoF algorithm to form the groups.

We search for the seeds following table 3 and obtain a list of the seeds. We then proceed to validate which of the groups obtained by the FoF is a valid group. To be considered as a valid group, it needs to contain at least one seed particle. This procedure allow us to take care of the case where two seeds form two groups which eventually get together. Now we can make sure these groups are merged into one and there is no need to search for this case individually.

6 Results

The following results correspond to our **Small (Dense)** described in table 1.

6.1 Fractional Anisotropy and Divergence in the Simulation

6.1.1 Fractional Anisotropy

We compute the FA after interpolating the density and velocity fields on a 256^3 grid using the CIC method and applying the V-Web algorithm. Figure 2 shows the FA histogram. Figure 3 shows a 2-D cut of the FA grid.

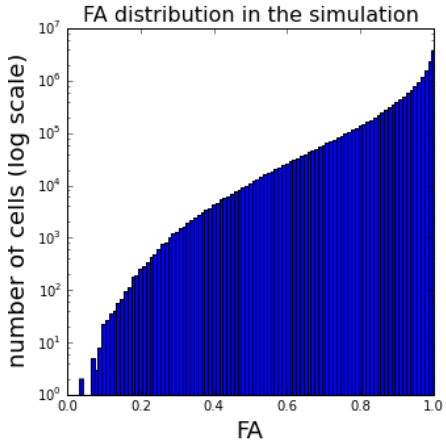

Figure 2: FA Histogram in the simulation

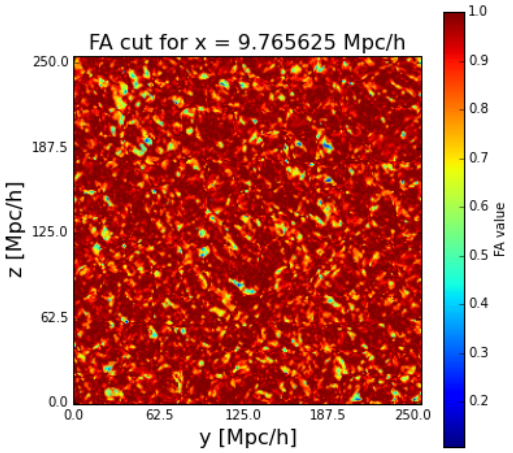

Figure 3: A cut of the FA in the x axis

As we saw in section 4.4, we expect to have few highly isotropic particles. The histogram in figure 2 we confirms this expectation.
The 2-D cut of the FA in figure 3 shows the filamentary structures we would expect of the Cosmic Web. This cut is very similar to the ones presented by Bustamante et al. [15] in their work. This is a good indicator that our simulation, CIC procedure V-Web application and FA forming are correctly executed. Bustamante et. al explain that void regions should have a $FA < 0.95$. We expect superclusters to have a similar isotropic behaviour as voids. Therefore we explore similar FA values in our method. To make sure we do not find voids we make use of the VDH.

6.1.2 VDH, Velocity Divergence

As we did for the FA, we visualize the histogram (Figure 4) and a 2-D cut (Figure 5) of the VDH.

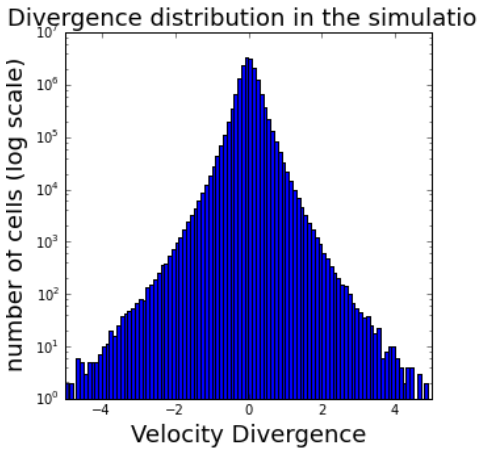

Figure 4: VDH Histogram in the simulation

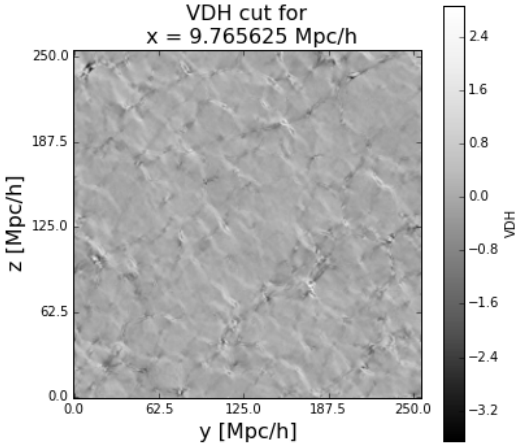

Figure 5: A cut of the VDH in the x axis

The VDH distribution is symmetric around 0.0. As we are searching for overdense structures (superclusters) we can then diminish our search space to half of it only by selecting the points with $VDH > 0.0$.
Observing a 2D cut of the Velocity Divergence as shown in figure 5 also gives a good idea of good idea of the Cosmic Web as we can easily recognize filaments (highly underdense, $VDH \leq -1.0$) and highly overdense regions ($VDH \geq 1.0$). Our candidates for seeds of supercluster structures are the highly overdense regions with $VDH \geq 1.0$.

6.2 Region Growing Algorithm and FoF with the FA and VDH

We apply all the combinations for thresholds in table 3 (seeds) and table 4 (growth).

Our algorithms run successfully on a regular laptop. Two examples of detected superclusters are shown individually on figure 6 and at the scale of the simulation (of size 250 Mpc/h) on figure 7. Finally we can see all the detected regions on the simulation with one selection of thresholds in figure 8.

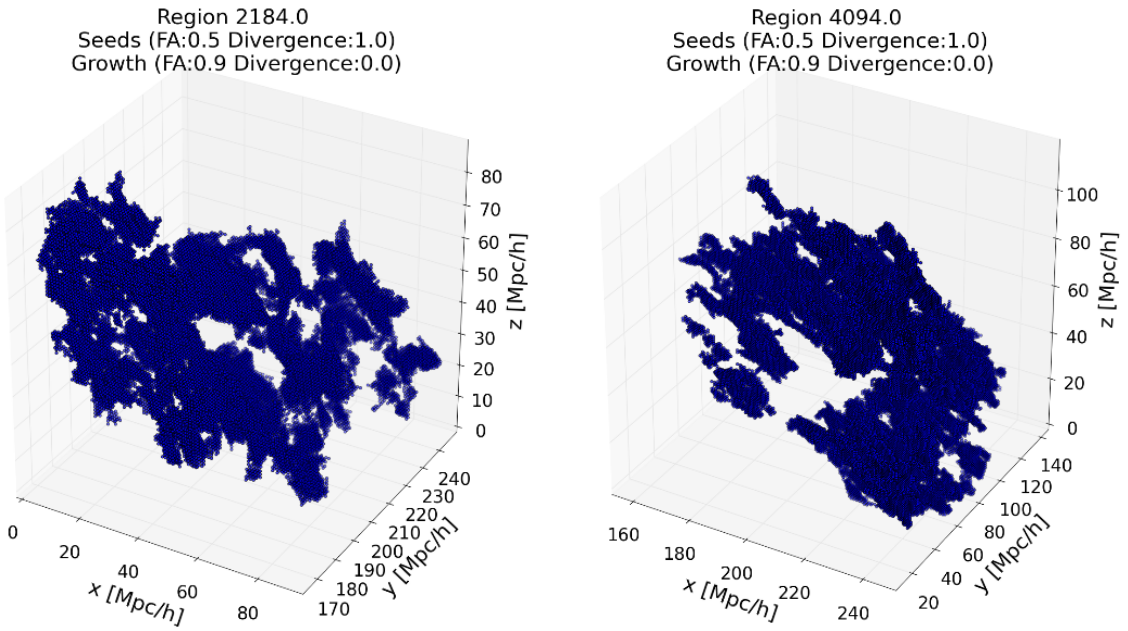

Figure 6: Two detected superclusters in the Small (Dense) simulation with thresholds: Seed (FA: 0.5, Div: 1.0) and Growth (FA: 9.0, Div:0.0).

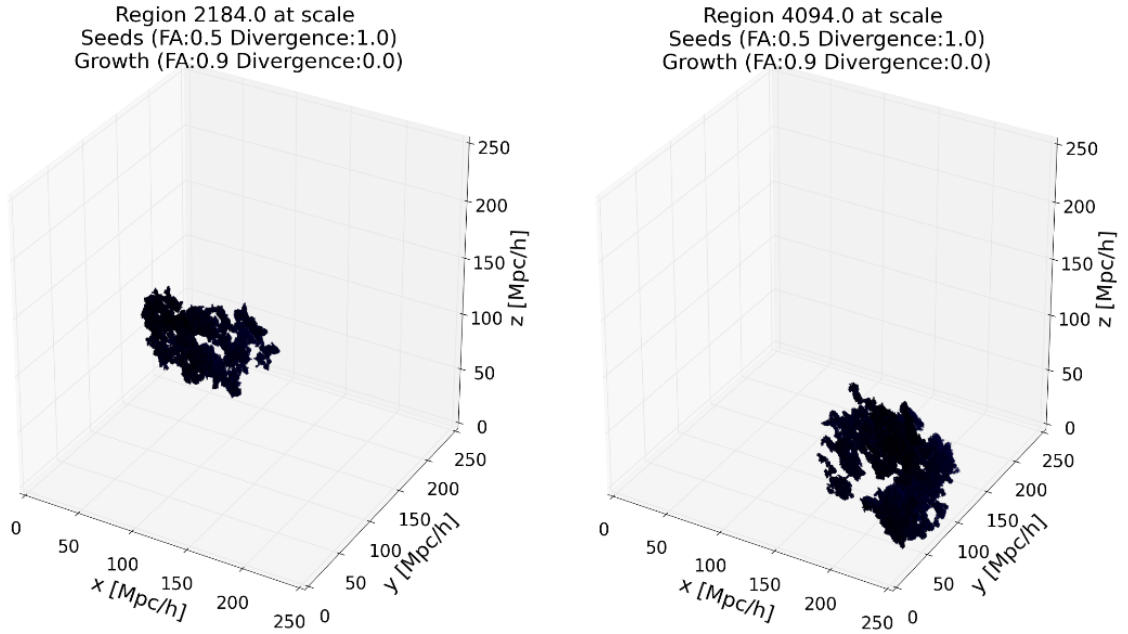

Figure 7: Two detected superclusters in the Small (Dense) simulation at scale with thresholds: Seed (FA: 0.5, Div: 1.0) and Growth (FA: 9.0, Div:0.0).

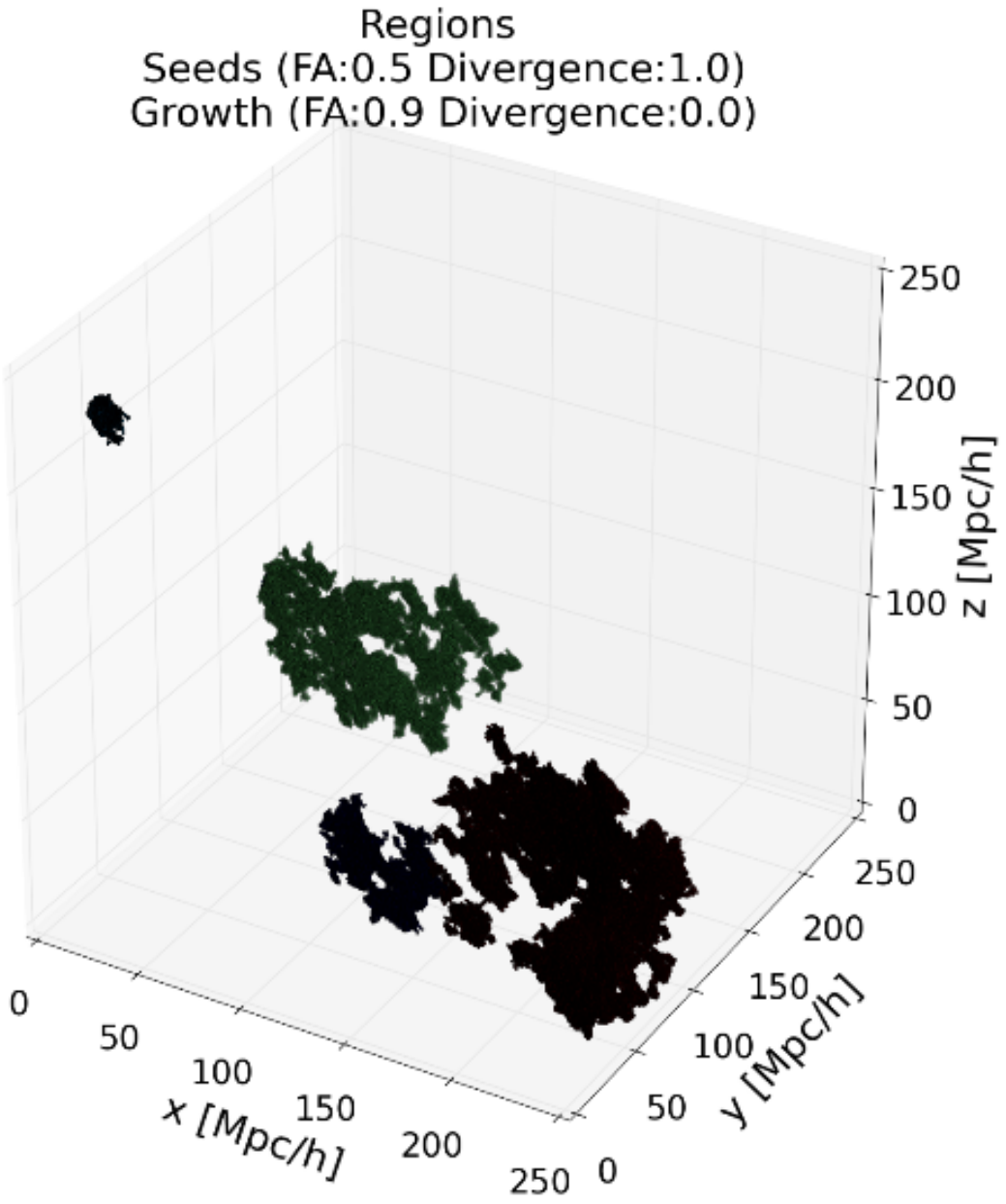

Figure 8: All the superclusters detected in the Small (Dense) simulation with thresholds: Seed (FA: 0.5, Div: 1.0) and Growth (FA: 9.0, Div:0.0). Different colors indicate different regions.

We can observe from figures 8, 7 and 6 that the structures have different shapes and sizes. There are no hints for a specific shape and size. In the next section we analyse with more detail how FA thresholds influence these results.

6.2.1 Influence of the FA in the seeds

We proceed to observe how changing the FA threshold for the seed selection process described in section 5.3 influences the detected structures.

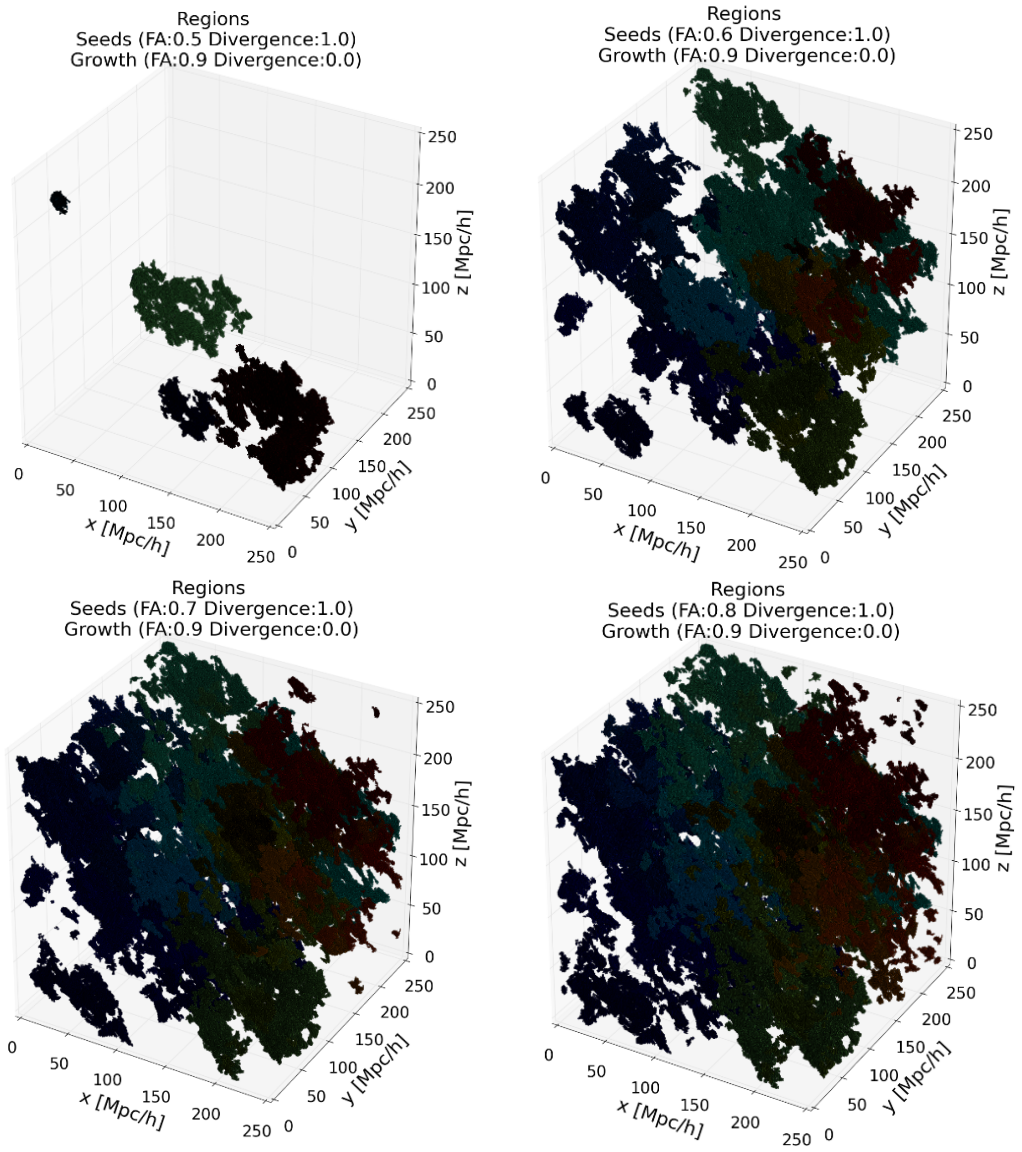

Figure 9: Effect of increasing the FA threshold in the seed selection. From upper left to down right the FA for the seeds are: 0.5, 0.6, 0.7 and 0.8. The growth FA is 0.9 and $tr\left(\sum_{\alpha\beta}\right)$ is 0.0. The $tr\left(\sum_{\alpha\beta}\right)$ for the seeds is 1.0.

In figure 9, we observe qualitatively increasing the FA threshold in the seed selection increases strongly the number of regions detected. This indicates that the number of seeds for the algorithm increases as well with the FA threshold in the seed selection. The FA histogram (figure 2) shows this as well as most part of the particles have a high value of FA.

6.2.2 Influence of the FA in the growth

We now observe how changing the FA threshold for the growth space affects our results.

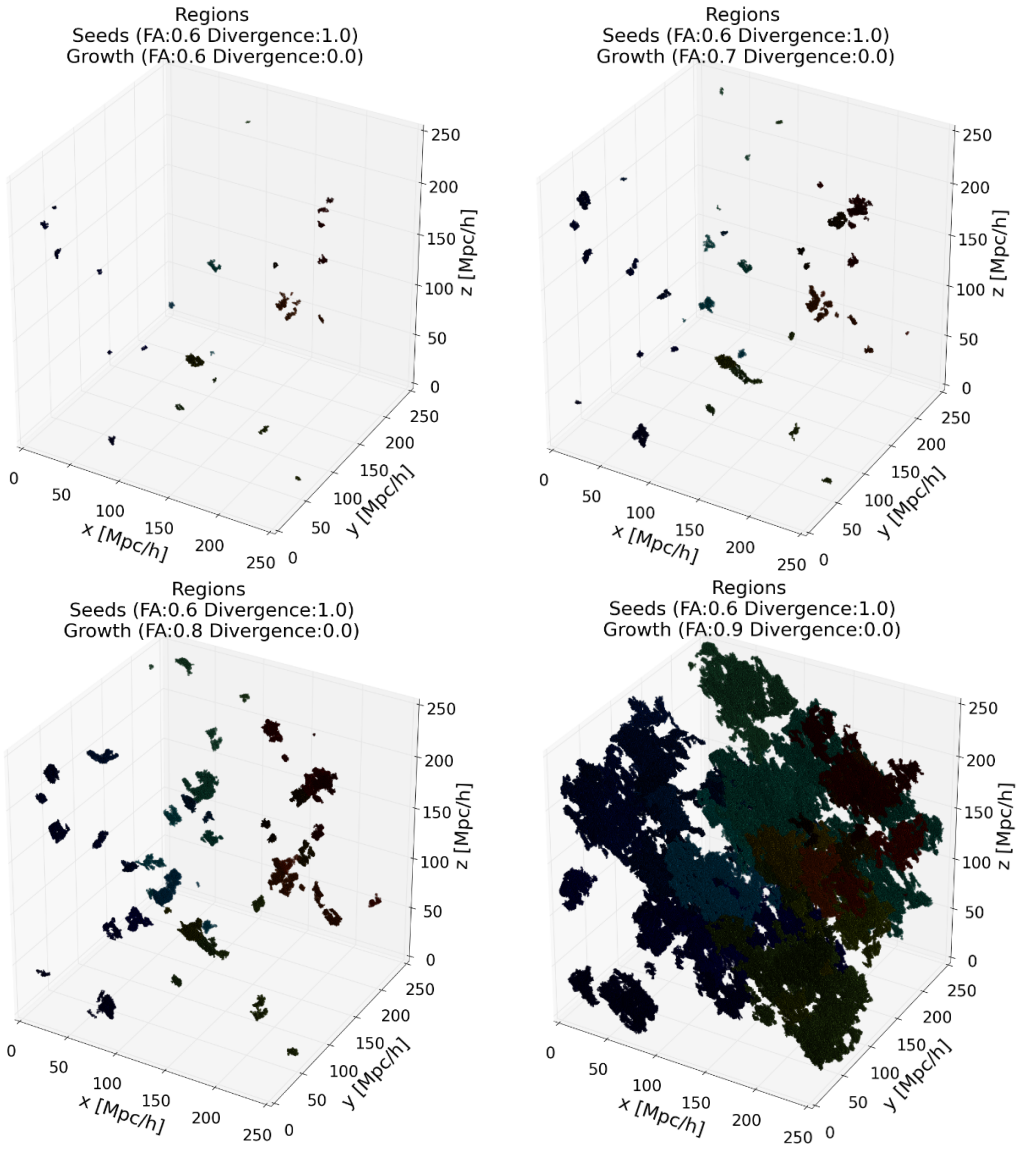

Figure 10: Efect of increasing the FA threshold in the growth. From upper left to down right the FA for the growth is: 0.6, 0.7, 0.8 and 0.9. The seed FA is 0.6 and VDH is 1.0. The VDH for the growth is 0.0.

Figure 10 shows qualitatively how augmenting the FA thresholds for the growth changes drastically the volumes of the regions detected, specially from the transition form FA threshold 0.8 to 0.9. This can also be seen comparing the possible growth space volume with the volume of the largest region detected ratio (Figure 11).

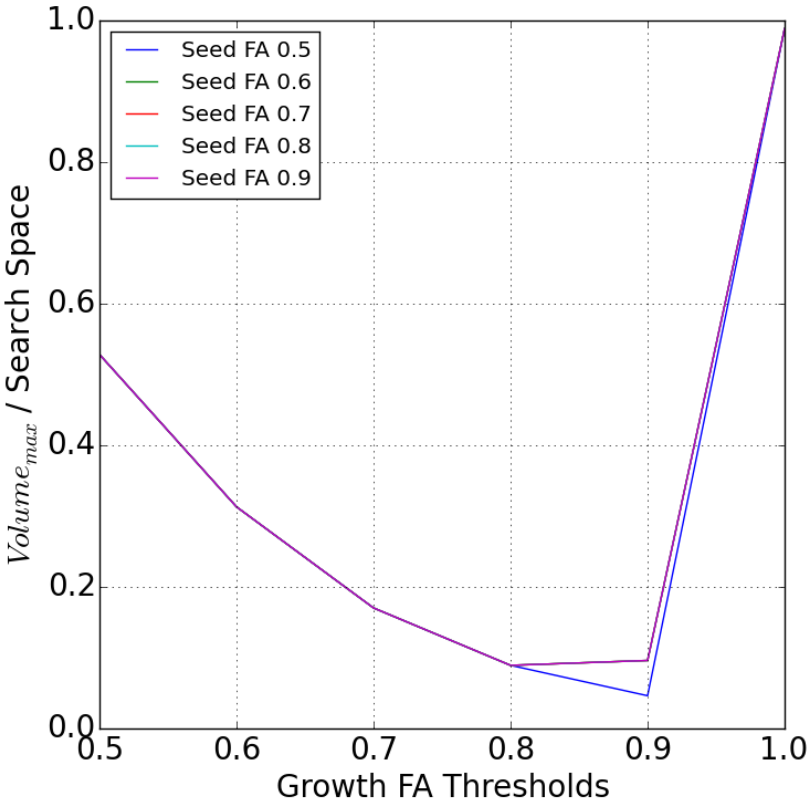

Figure 11: Ratio of the volume of the largest structure and Volume of the growth space increasing the FA threshold in the growth. From upper left to down right the FA for the seeds is: 0.5, 0.6, 0.7 and 0.8. The growth VDH is 1.0. The VDH for the seed is 0.0.

Figure 11, shows that changing the growth FA threshold follows the same pattern for the different seed FA thresholds. We can observe that when maximizing the growth FA threshold ($FA < 1.0$) the detected regions converge into a single structure which occupy all the search space.

6.2.3 Volume distribution

The detected structures vary in volume and shape. The number and size of the detected structures strongly depend on the choice for FA thresholds (seed and growth) as it

is presented in sections 6.2.1 and 6.2.2. We proceed to visualize the distribution of detected structures volume for FA thresholds where we have a significant number of detected structures.

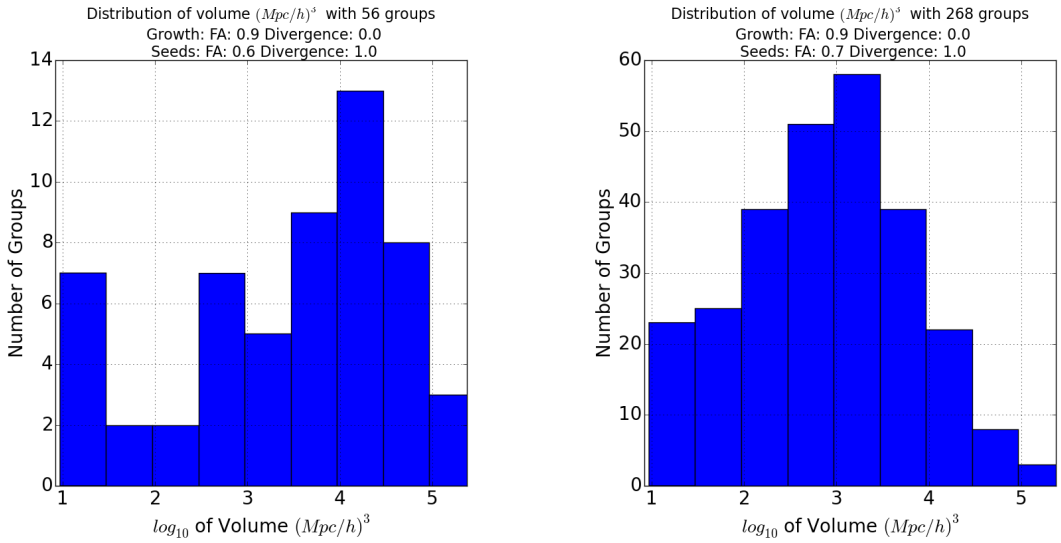

Figure 12: Volume distributions for low values of seed FA threshold (0.6 and 0.7)

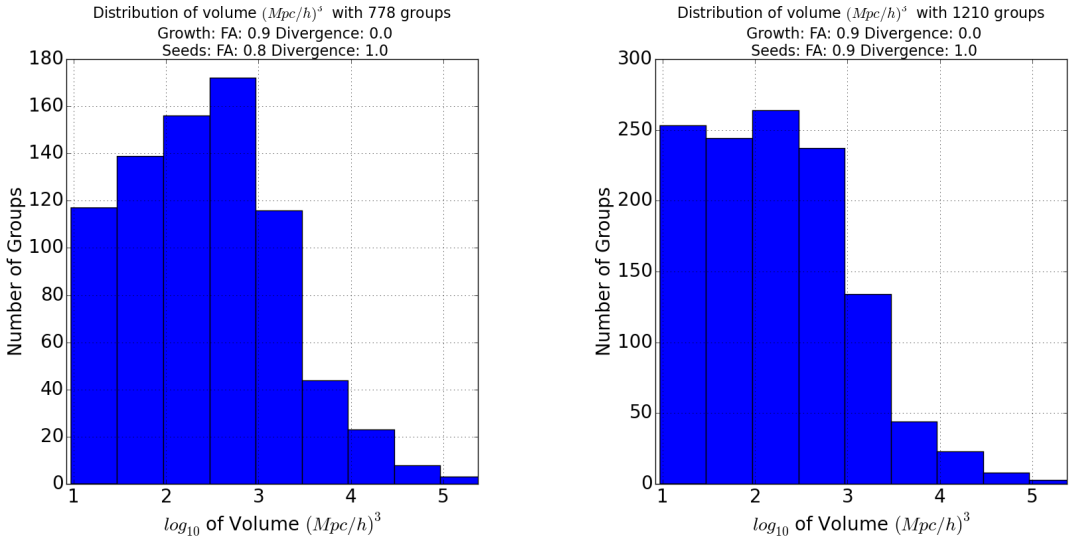

Figure 13: Volume distributions for higher values of seed FA threshold (0.8 and 0.9)

We find that as the seed FA threshold increases, the volume distribution tend to center to smaller volumes.

6.2.4 Shape of the regions

To study the shape of the regions in a quantitative way we proceed, for each region, to form the Inertia Tensor, calculate its eigenvalues and observe how these eigenvalues differ from each other.

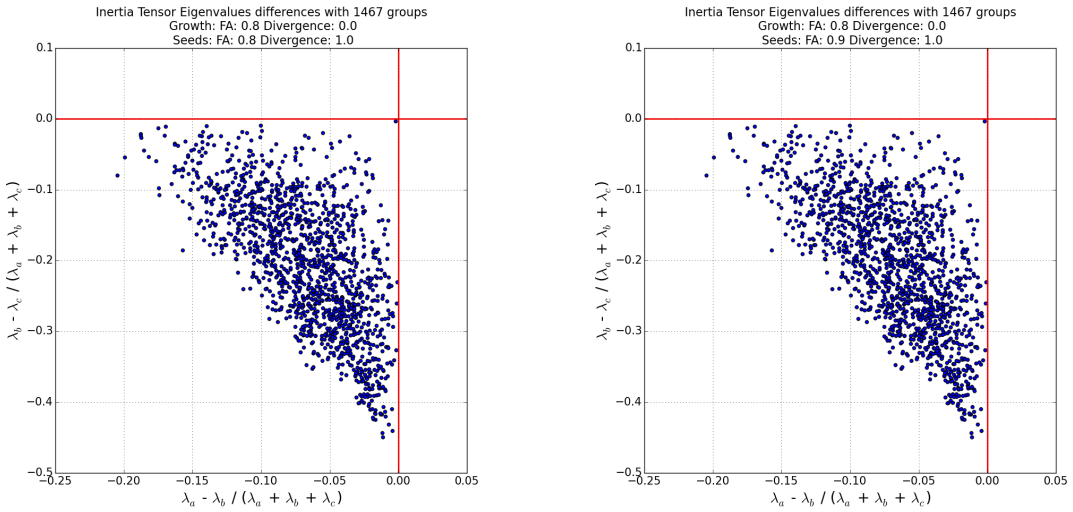

Figure 14: Inertia Tensor eigenvalues differences (normalized) for two different cases

Figure 14 shows that most of our detected structures correspond to the case where $\lambda_a < \lambda_b < \lambda_c$ (asymmetric structures). Nevertheless, if we set a tolerance we can observe that there are cases where $\lambda_a \sim \lambda_b < \lambda_c$ (oblate symmetric structures), where $\lambda_a < \lambda_b \sim \lambda_c$ (oblate symmetric structures), and very few where $\lambda_a \sim \lambda_b \sim \lambda_c$ (spherical structures).

6.3 Comparisons with Laniakea

Let us remember the properties of Laniakea we can compare our detected structures with are described in table 5

Radius (Mpc/h)	80
Mass (Solar Masses)	1×10^{17}

Table 5: Properties of Laniakea we can compare with

- Size and Volume

Laniakea is a spheroid of radius 80 Mpc/h [1]. We can state it has approximatively a volume of 2.14×10^6 $[Mpc/h]^3$.

We proceed to see how Laniakea fits in our volume distributions.

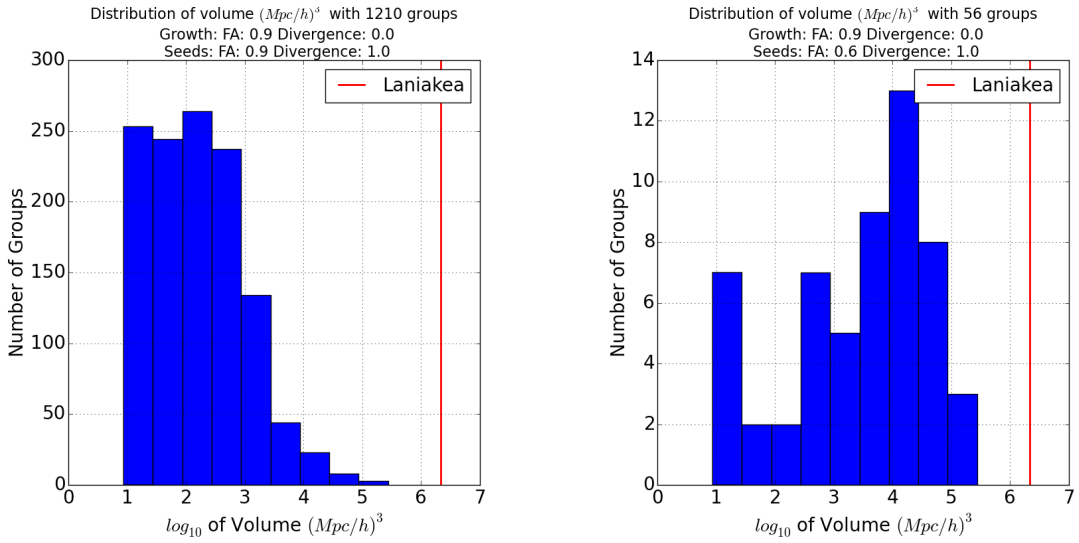

Figure 15: Two distributions of volumes compared with the volume of Laniakea where Laniakea is larger.

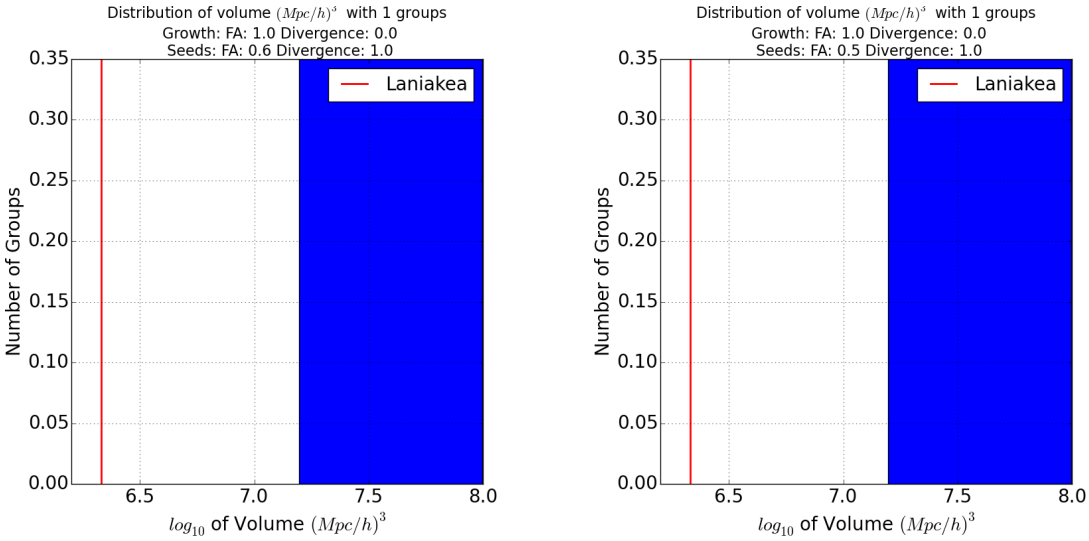

Figure 16: Two distributions of volumes compared with the volume of Laniakea where Laniakea is smaller.

We can observe in figure 15 that for Growth FA thresholds different than 1.0, Laniakea is always larger than our detected structures. Figure 16, where the Growth FA threshold of 1.0 corresponds to the case where the detected structure occupies all the available space as exposed in section 6.2.2.

7 Conclusions and Future Work

Our method, based on the Fractional Anisotropy (FA) and the Velocity Divergence, successfully find structures in cosmological simulations.

Nevertheless it is very sensitive to its parameters (FA and Divergence thresholds). An exhaustive calibration can must be performed before interpreting the results.

We state that Laniakea is, in the Cosmological context, atypically larger. We visualise that, as long as the growth FA threshold is not the maximal FA value (1.0) and that the volume of our detected structure is not equal to the available growth space, Laniakea's volume is always larger than the detected structures volumes. However, our simulation volume is small

We provide here results for the Small (Dense) simulation. Nevertheless we generated data from the two other simulations exposed in table 1. A detailed analysis of how the detected structures behave for the different simulations is to be done.

A more exhaustive FA thresholds analysis is to be done similar to the one performed in this work. The behaviour for Growth FA thresholds for values between 0.9 and 1.0 is to be examined in more detail.

In order to have a larger sample of detected large structures, a bigger simulation is to be performed with parameters Boxsize: 1000 Mpc/h and 2048^3 DM particles.

References

[1] R. Brent Tully, Hlne Courtois, Yehuda Hoffman, and Daniel Pomarde. The Laniakea supercluster of galaxies. *Nature*, 513(7516):71–73, September 2014.

[2] J. Richard Gott III, Mario Juri, David Schlegel, Fiona Hoyle, Michael Vogeley, Max Tegmark, Neta Bahcall, and Jon Brinkmann. A Map of the Universe. *The Astrophysical Journal*, 624(2):463–484, May 2005.

[3] R. Brent Tully, Hlne M. Courtois, Andrew E. Dolphin, J. Richard Fisher, Philippe Hraudeau, Bradley A. Jacobs, Igor D. Karachentsev, Dmitry Makarov, Lidia

Makarova, Sofia Mitronova, Luca Rizzi, Edward J. Shaya, Jenny G. Sorce, and Po-Feng Wu. *COSMICFLOWS-2* : THE DATA. *The Astronomical Journal*, 146(4):86, October 2013.

[4] Gayoung Chon, Hans Bhringer, Chris A. Collins, and Martin Krause. Characterising superclusters with the galaxy cluster distribution. *Astronomy & Astrophysics*, 567:A144, July 2014.

[5] Gayoung Chon, Hans Bhringer, and Saleem Zaroubi. On the definition of superclusters. *Astronomy & Astrophysics*, 575:L14, March 2015.

[6] Gianfranco Bertone, Dan Hooper, and Joseph Silk. Particle dark matter: evidence, candidates and constraints. *Physics Reports*, 405(5-6):279–390, January 2005.

[7] P. J. E. Peebles and Bharat Ratra. The cosmological constant and dark energy. *Reviews of Modern Physics*, 75(2):559–606, April 2003.

[8] Virginia Trimble. Existence and Nature of Dark Matter in the Universe. *Annual Review of Astronomy and Astrophysics*, 25(1):425–472, September 1987.

[9] Elmo Tempel. Cosmology: Meet the Laniakea supercluster. *Nature*, 513(7516):41–42, September 2014.

[10] T. Padmanabhan. *Theoretical astrophysics*. Cambridge University Press, Cambridge ; New York, 2000.

[11] S. Zaroubi, Y. Hoffman, K. B. Fisher, and O. Lahav. Wiener Reconstruction of the Large-Scale Structure. *The Astrophysical Journal*, 449:446, August 1995.

[12] Yehuda Hoffman, Ofer Metuki, Gustavo Yepes, Stefan Gottlber, Jaime E. Forero-Romero, Noam I. Libeskind, and Alexander Knebe. A kinematic classification of the cosmic web: The cosmic web. *Monthly Notices of the Royal Astronomical Society*, 425(3):2049–2057, September 2012.

[13] Peter J. Basser. Inferring microstructural features and the physiological state of tissues from diffusion-weighted images. *NMR in Biomedicine*, 8(7):333–344, November 1995.

[14] N. I. Libeskind, Y. Hoffman, J. Forero-Romero, S. Gottlober, A. Knebe, M. Steinmetz, and A. Klypin. The velocity shear tensor: tracer of halo alignment. *Monthly Notices of the Royal Astronomical Society*, 428(3):2489–2499, January 2013.

[15] Sebastian Bustamante and Jaime E. Forero-Romero. Tensor anisotropy as a tracer of cosmic voids. *Monthly Notices of the Royal Astronomical Society*, 453(1):497–506, October 2015.

[16] V. Springel. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134, 2005.

[17] Volker Springel, Simon D. M. White, Adrian Jenkins, Carlos S. Frenk, Naoki Yoshida, Liang Gao, Julio Navarro, Robert Thacker, Darren Croton, John Helly, John A. Peacock, Shaun Cole, Peter Thomas, Hugh Couchman, August Evrard, Jrg Colberg, and Frazer Pearce. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*, 435(7042):629–636, June 2005.

[18] Francis H Harlow. The particle-in-cell computing method for fluid dynamics. *Methods in computational physics*, 3(3):319–343, 1964.

[19] Yu N. Grigoryev, V. A. Vshivkov, and M. P. Fedoruk. *Numerical "particle-in-cell" methods: theory and applications*. VSP, Utrecht, 2002.

[20] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 3rd ed edition.

[21] R. Thompson. pyGadgetReader: GADGET snapshot reader for python. Astrophysics Source Code Library, November 2014.

A Appendix

A.1 Approach by the Speed

A.1.1 Region Growing Algorithm with the Speed

We present here a naive way to approach to the problem.

1. We calculate the magnitude of the velocity (the speed) of each particle.
2. We look for regions where the center has the highest speed and from it the speed decreases while the particle is further away from the center.
3. We define the limits of this region the regions where the speed begins to increase with the distance from the center.

We use a region growing algorithm, a simple image segmentation algorithm used to identify different regions in a image [20].

$f(x, y, z)$ denotes the input data. $S(x, y, z)$ denotes a seed array, with value 1 where the particle can be define as a seed and 0 where not. It is the same size of $f(x, y, z)$. Q denotes a predicate which is to be apply to the input data and determines if the region which starts at the seeds grows or not. Here Q denotes: "if the speed of the close particle is lower than the marked one but greater than a threshold, mark that particle and continue."

1. We choose seeds to begin the growth. In this case we choose the particles with high velocities. Here
$$|v| > th_{high}$$

2. We open a window of particles to look for 2 close particles from the seed s which do not for part of $S(x, y, z)$. Here:
$$window = 20000$$

3. Once we found the 2 close particles c we apply the predicate Q. Here:
$$Q := |v_s| > |v_c| > |th_{low}|$$

4. If the particle c satisfies Q, it is marked, $S(c) = 1$

5. Apply the algorithm to c, and so on (Recursion).

Here we defined empirically different thresholds in order to observer the results. First we used:
$th_{low} = v_{min} + \frac{\sigma_v}{2}$ and $th_{high} = v_{max} - \sigma_v$.
Secondly we used:
$th_{low} = \vec{v} + 2\sigma_v$ and $th_{high} = \vec{v} + 8\sigma_v$.

The first version of the algorithm was written in python using the module pyGadgetReader [21] to read the data and transform it to NumPy arrays in Python. In Appendix A.1.3 we attach the source code used.

A.1.2 Results of the Speed Approach

This approach was tested on the simulation of boxsize 500 Mpc/h and 512^3 dark matter particles (DM) described in table 1.

We first visualize the speed histogram in figure 17.

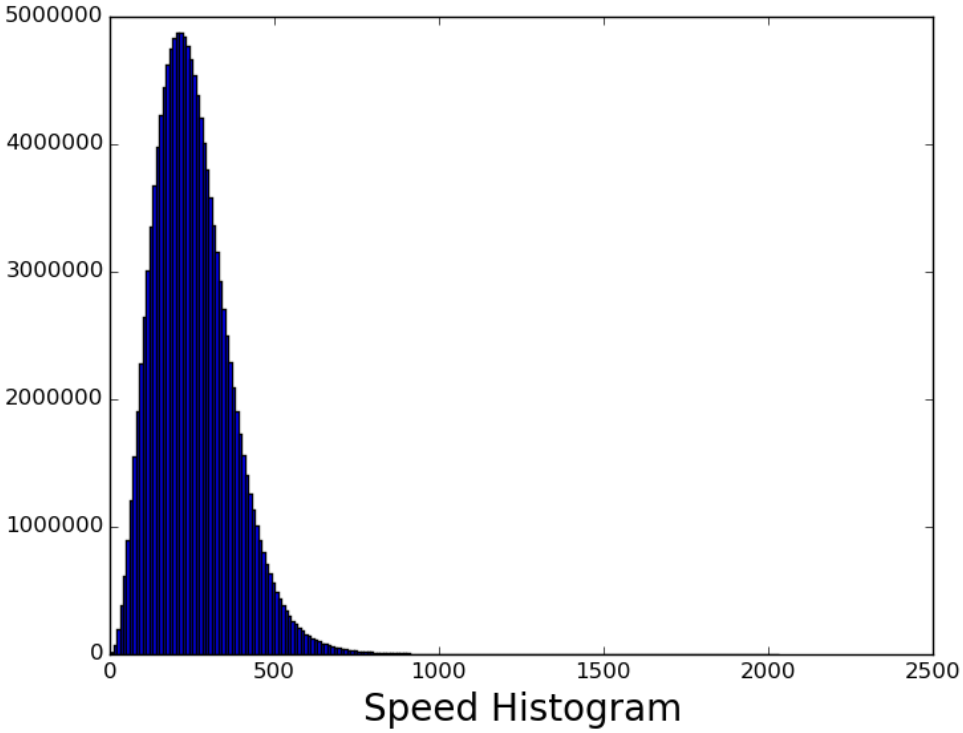

Figure 17: Speed Histogram of the simulation

The histogram resembles a Poisson distribution. This is a consequence of the perturbations generated in the initial conditions generation.

v_{max}	2023.87
v_{min}	0.34
σ_v	117.75

Table 6: Properties of the speed distribution

Our hypothesis is based on the most direct implication of gravity. The lower speed particles will mainly be in the border regions and the higher speed particles will be in the center regions.

For this we must choose a threshold for low velocities and a threshold for high velocities. We obtain: $th_{low} = v_{min} + \frac{\sigma_v}{2} = 59.2$ and $th_{high} = v_{max} - \sigma_v = 1906.12$

We first have a look at the particle distribution of particles with speed higher than the threshold for low velocities.

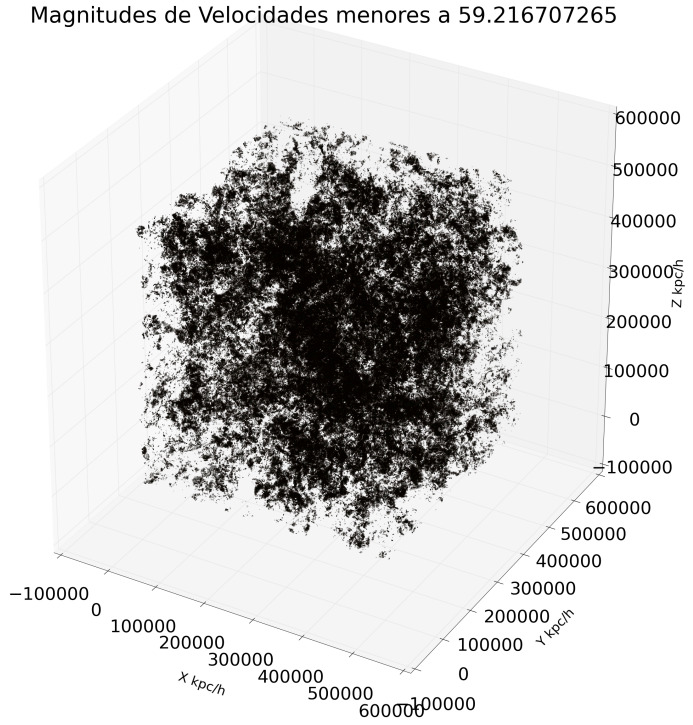

Figure 18: DM particles with $|v| < 59.2$

There seem to be some structures, but it is not clear enough. This is due to the high number of DM particles used in this simulation (512^3).

We produce cuts in the z-direction to visualize in 2-D the speed distribution.

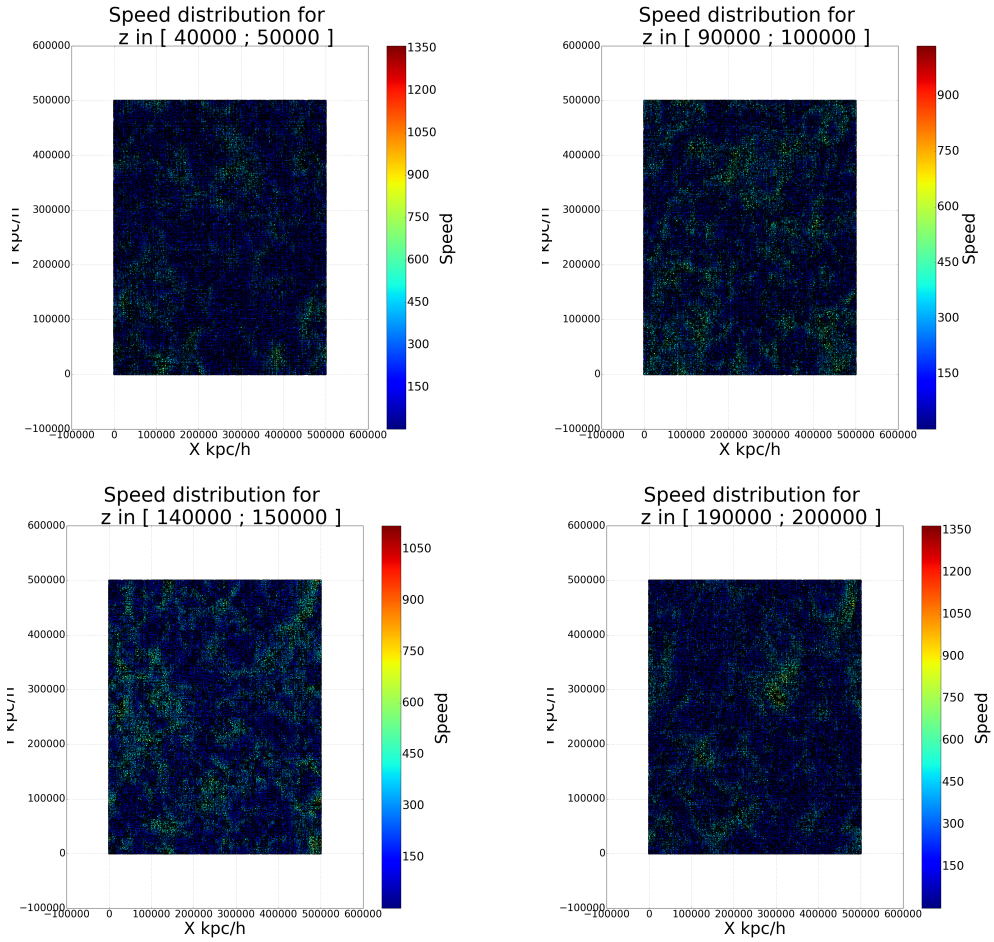

Figure 19: 2D slices of the speed distribution

We can observe in figure 19 intuitive signs of structures related to the speed distribution. We can observe filament structures

It is more clear if we observe from one side the particles with speed greater than a th_{high} and the particles with speed lesser than a th_{low}.

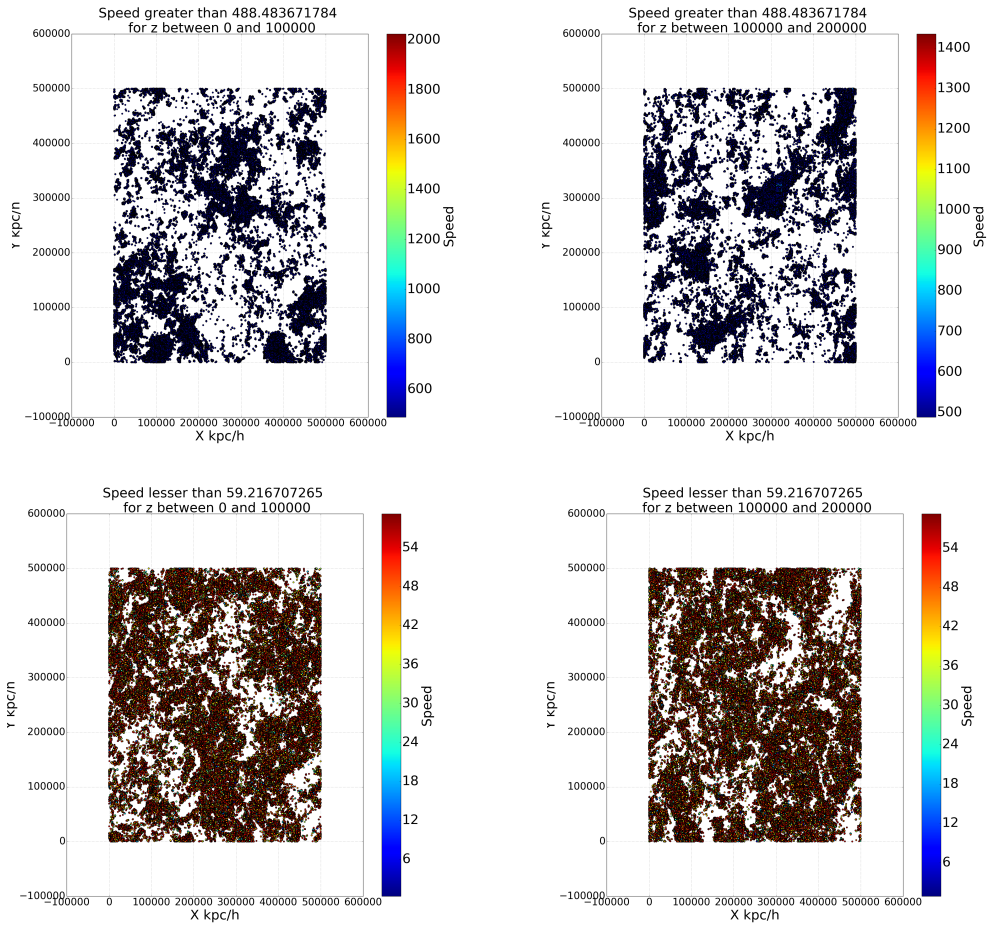

Figure 20: 2D slices of the speed distribution. The upper slices with $|\vec{v}| > th_{high} = 488$ and the two lower slices with $|\vec{v}| < th_{low} = 59$ for the same cuts, $z = \{100, 200\} Mpc/h$

As we can see in figure 20 the cuts with low and high thresholds are complementary. It is evident that structures are present.

Finally we apply our region growing algorithm and we obtain the following results.

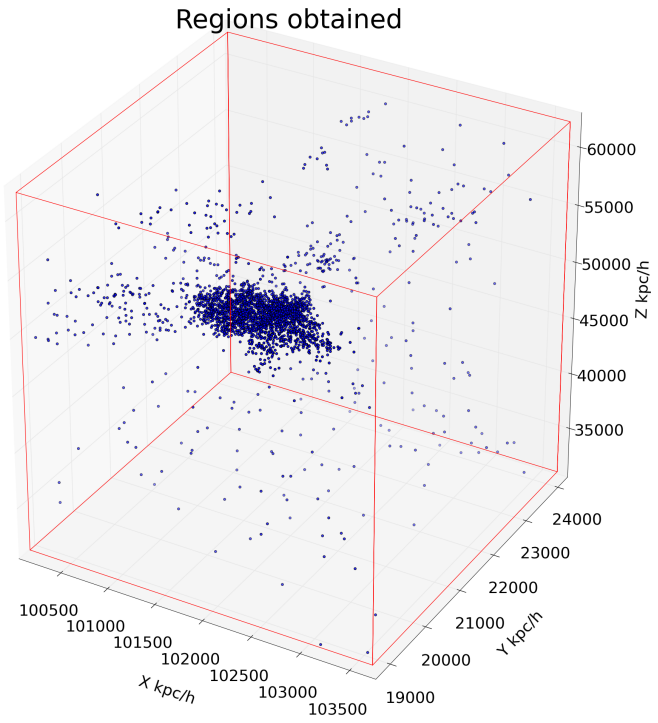

Figure 21: Region identified by the algorithm

And, changing the thresholds to: $th_{low} = \bar{v} + 2\sigma_v = 488.48$ and $th_{high} = \bar{v} + 8\sigma_v = 1195.0$
We obtain:

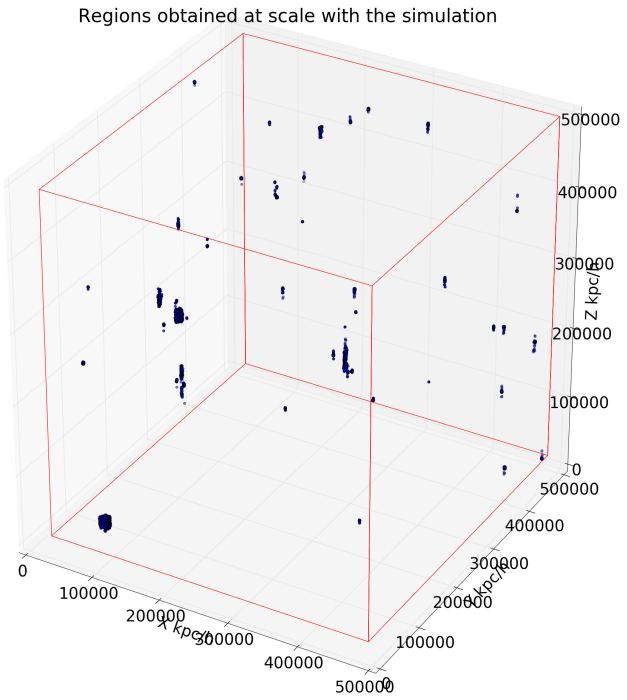

Figure 22: Region identified by the algorithm with the second thresholds

We have different observations:

1. The details of the regions in figures 21 and 22 seem accurate with what we would expect. It is more dense in the center and less dense in the borders.
2. The regions obtained have a dimension of only a few Mpc/h (Laniakea's dimension is two orders of magnitude higher). This could mean that Laniakea is atypical.
3. There is first only one region identified. This indicates than all the high velocity particles are congregated in only one region (the detected region). If we set lower th_{high} we can see we get more regions.
4. This is certainly a good naive approach to the problem. Nevertheless for more accurate results it needs to be drastically changed.

The script in Appendix A.1.3 was executed in a student's laptop and later on the HPC cluster. For a significant number of particles, the laptop limited memory (8GB) crashes. Running in the HPC solves this issue.

A.1.3 Code example for Region Growing Algorithm - Approach by the Speed

Listing 1: Region Growing Algorithm Code - Speed

```
from pygadgetreader import *
import matplotlib
matplotlib.use('Agg')
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import matplotlib.pyplot as plt
import numpy as np

snap_file_sim='/lustre/home/ciencias/fisica/pregrado/sd.hernandez204/Outputs/28_02_2016/snapshot_005'

requested='dmcount'
n_dm=readheader(snap_file_sim,requested)
print "Number of Dark Matter particles"
print n_dm

array_pos=readsnap(snap_file_sim,'pos','dm')
array_vel=readsnap(snap_file_sim,'vel','dm')

array_magnitud_vel=np.zeros(n_dm)
for i in range(0,n_dm):
    array_magnitud_vel[i]=np.linalg.norm(array_vel[i,:])
print "The speed array has been calculated"

longitud = (int) (array_pos.shape[0])
print "The lenght of the position array should be the same number of particles"
print str(longitud)

array_copy=array_pos.copy()
array_results=np.zeros((longitud,4))
array_results[:,:-1] = array_copy
print "The results array has been created."

vel_max = np.max(array_magnitud_vel)
vel_min = np.min(array_magnitud_vel)
vel_std = np.std(array_magnitud_vel)
vel_mean = np.mean(array_magnitud_vel)

print "Vel_max, Vel_min, std_vel, vel_mean"
print vel_max, vel_min, vel_std, vel_mean
thresh_v_hig = vel_mean + 8.0 * vel_std
thresh_v_low = vel_mean + 2.0 *  vel_std
print "Thresh_high, Thresh_low"
print thresh_v_hig, thresh_v_low

#Gets the indexes of the seeds
indexes = np.where(array_magnitud_vel>(thresh_v_hig))[0]
print 'There are: '+ str(indexes.shape[0]) +' seeds'

def region_accretion(l):
    #Resize the window if necesary
    size_window_up = n_window
    if(((l+size_window_up) > longitud)):
        size_window_up = longitud - l
    size_window_down = n_window
    if(((l-size_window_down) < 0)):
        size_window_down = 0 + l
    distancias_l_up = np.zeros(size_window_up)
    distancias_l_down = np.zeros(size_window_down)
    distancias_l_up[:] = 1000000
    distancias_l_down[:] = 1000000
    if(size_window_up == size_window_down):
        for k in range(0,size_window_up):
            if(((l+k) < longitud)):
                if((array_results[l+k,3] == 0)):
                    distancias_l_up[k]=np.sqrt(np.sum((array_results[l,0:2]-array_results[l+k,0:2])**(2.0)))
            if(((l-k) > 0)):
                if((array_results[l-k,3] == 0)):
                    distancias_l_down[k]=np.sqrt(np.sum((array_results[l,0:2]-array_results[l-k,0:2])**(2.0)))
    else:
        for k in range(0,size_window_up):
```

```
            if(((l+k) < longitud)):
                if((array_results[l+k,3] == 0)):
                    distancias_l_up[k]=np.sqrt(np.sum((array_results[l,0:2]-array_results[l+k,0:2])**(2.0)))
        for k in range(0,size_window_up):
            if(((l-k) > 0)):
                if((array_results[l-k,3] == 0)):
                    distancias_l_down[k]=np.sqrt(np.sum((array_results[l,0:2]-array_results[l-k,0:2])**(2.0)))
    distancias_l_up[0]=1000000
    distancias_l_down[0]=1000000
    closest_l_up = np.min(distancias_l_up)
    index_closest_up = np.argmin(distancias_l_up)
    closest_l_down = np.min(distancias_l_down)
    index_closest_down = np.argmin(distancias_l_down)
    #Now si if these validate the condition
    #It has to see if it got any new candidates (not marked particles)
    if ((closest_l_up != 1000000)):
        vel_l = array_magnitud_vel[l]
        vel_closest_up = array_magnitud_vel[l + index_closest_up]
        if( (vel_l>= vel_closest_up>=thresh_v_low)&(array_results[l+index_closest_up,3]==0)):
            #it is marked
            array_results[l + index_closest_up,3] = 1
            #Recursion
            l_up = l + index_closest_up
            region_accretion(l_up)
    if ((closest_l_down != 1000000)):
        vel_l = array_magnitud_vel[l]
        vel_closest_down = array_magnitud_vel[l + index_closest_down]
        if( (vel_l >= vel_closest_down>=thresh_v_low)&(array_results[l-index_closest_down,3]==0)):
            #it is marked
            array_results[l - index_closest_down,3] = 1
            #Recursion
            l_down = l - index_closest_down
            region_accretion(l_down)

#Define the default size of the search window
n_window=10000
#Now mark the seeds
for i in indexes:
    array_results[i,3] = 1
#Now do the recursion
for i in indexes:
    #Recursion
    region_accretion(i)

print 'The regions have: '+ str(len(array_results[array_results[:,3] == 1,:])) +' galaxies.'
print 'Starting with points with V> '+ str(thresh_v_hig)
print 'Ending at points with V< ' + str(thresh_v_low)

#Now it gets the points of the regions
array_pos_region=array_results[(array_results[:,3] == 1),:]
```

A.2 Simple implementation of the Region Growing Algorithm with the FA and VDH

A.2.1 Code example for Region Growing Algorithm - Approach by the FA and VDH

Listing 2: Region Growing Algorithm Code - FA & VDH

```
from ast import literal_eval
from struct import *
import numpy as np
import matplotlib.pyplot as plt
from pylab import *

# A Region Accretion Algorithm using the FA and the VDH

#Predicate to be part of the group
def predicate(i,j,k, seed_FA):
    if(results[i,j,k]>=1):
        return False #There is no need to re-evaluate the cell
    elif( (FA[i,j,k] <= thresh_FA) & (Trace[i,j,k] >= thresh_Trace)):
    #elif((Trace[i,j,k] >= thresh_Trace)):
        #It validates the predicate
        return True
    else:
        #It fails the predicate
        return False

#The algorithm, it evaluates the special cases
```

```
def region_accretion_4_neigboors(i,j,k):
    present_FA = FA[i,j,k]
    present_region = results[i,j,k]
    #Finds if it is on the border of the grid
    case = 0
    #Cases
    if(i==0):
        case+=1
    if(j==0):
        case+=2
    if(k==0):
        case+=4
    if(i==n_grid):
        case+=8
    if(j==n_grid):
        case+=16
    if(k==n_grid):
        case+=32
    #General case
    if(case==0):
        if(predicate(i+1,j,k,present_FA)):
            results[i+1,j,k] = present_region
            region_accretion_4_neigboors(i+1,j,k)
        if(predicate(i,j+1,k,present_FA)):
            results[i,j+1,k] = present_region
            region_accretion_4_neigboors(i,j+1,k)
        if(predicate(i,j,k+1,present_FA)):
            results[i,j,k+1] = present_region
            region_accretion_4_neigboors(i,j,k+1)
        if(predicate(i,j-1,k,present_FA)):
            results[i,j-1,k] = present_region
            region_accretion_4_neigboors(i,j-1,k)
        if(predicate(i,j,k-1,present_FA)):
            results[i,j,k-1] = present_region
            region_accretion_4_neigboors(i,j,k-1)
        if(predicate(i-1,j,k,present_FA)):
            results[i-1,j,k] = present_region
            region_accretion_4_neigboors(i-1,j,k)
    #Special cases
    #... Border cases

inputfolder = 'simulation_folder/'
inputfile_1 = 'snapshot_005.eigen_1'
inputfile_2 = 'snapshot_005.eigen_2'
inputfile_3 = 'snapshot_005.eigen_3'

# Formation of the Shear Tensor eigenvalues grids, FA and VDH
grid_1 = read_CIC_scalar(inputfolder+inputfile_1)
grid_2 = read_CIC_scalar(inputfolder+inputfile_2)
grid_3 = read_CIC_scalar(inputfolder+inputfile_3)

FA = (grid_1-grid_3)**2  + (grid_2-grid_3)**2  + (grid_1-grid_2)**2
FA = FA/(grid_1**2 + grid_2**2 + grid_3**2)
FA = np.sqrt(FA)/np.sqrt(3.0)

Trace = grid_1 + grid_2 + grid_3

#Seed selection
n_x,n_y,n_z = shape(FA)
n_grid = n_x-1 # the size of box -1, the maximum index
results = np.zeros(shape(FA)) #preps the results matrix
thresh_FA = 0.8
thresh_Trace = 0.0
thresh_FA_seeds = 0.6
thresh_Trace_seeds = 1.0

#Needs to pick the seeds.
seed_list = []

counter = 0
for i in range (n_x):
    for j in range (n_y):
        for k in range (n_z):
            if((FA[i,j,k]<thresh_FA_seeds ) & (Trace[i,j,k]>thresh_Trace_seeds)):
                counter +=1
                results[i,j,k] = counter
                seed_list.append([i,j,k])


#Growth thresholds selection
thresh_FA = 0.8
thresh_Trace = 0.0

#Runs the Algorithm
for (i,j,k) in seed_list:
    region_accretion_4_neigboors(i,j,k)
```

A.2.2 Region Growing Algorithm with the FA and VDH

The results from the simple implementation are good. Nevertheless, our implementation using the FoF is much more efficient. We show this simple implementation and its results in a way to illustrate the concepts behind the Region Growing Algorithm.

Using the data from our Small(Dense) simulation described in table 1, after the CIC (in a 256^3 grid) and V-Web have been applied, we run the implementation described in section 5.3.2 for some of the values described in tables 3 and 4.

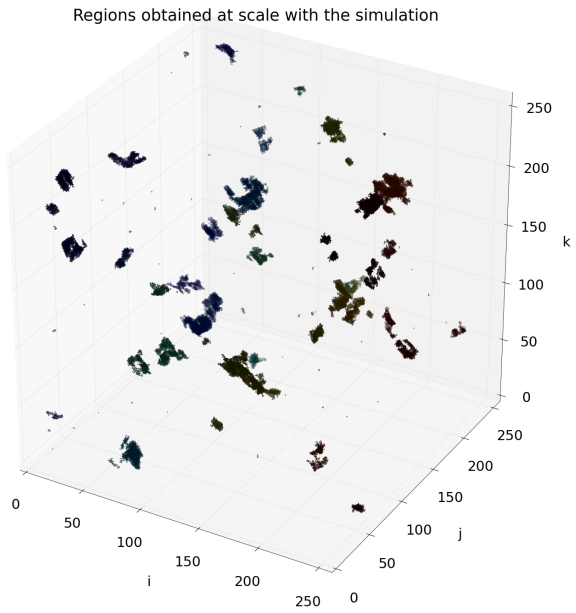

Figure 23: Regions detected for: Seeds (FA:0.6, VDH: 1.0) and Growth (FA:0.8, VDH: 0.0)

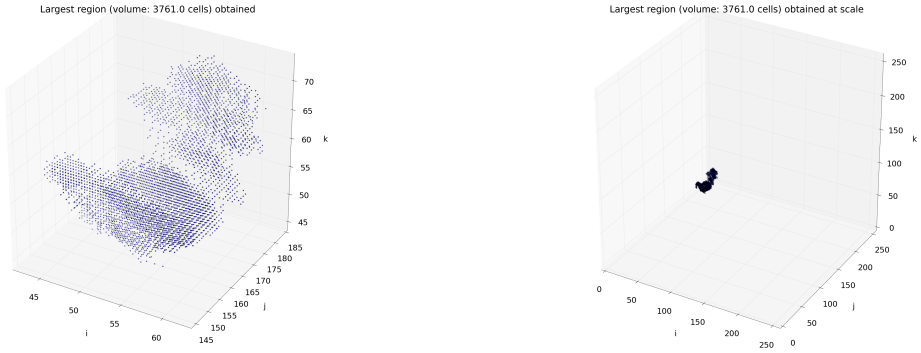

Figure 24: Largest volume detected for: Seeds (FA:0.6, VDH: 1.0) and Growth (FA:0.8, VDH: 0.0). The region is on the left and can be seen at scale with the simulation on the right.