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 10, 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

1 Acknowledgements **2**

2 Abstract **3**

3 Introduction **5**

4 Background **5**
4.1 Cosmic flows and peculiar velocities 5
4.2 Boundaries of Laniakea - The V-Web Algorithm 5
4.3 Fractional Anisotropy and Overdensity 6
4.4 The Current Model: Cosmological Parameters 6

5 Methods **6**
5.1 Simulations 6
5.2 Approaches 7
5.3 Algorithms 9
5.3.1 Region Growing Algorithm: Basic Description 9
5.3.2 Region Growing Algorithm with the FA and the trace of the velocity shear tensor 10

6 Results **12**
6.1 General visualizations (FA, δ) of the Simulations 12
6.1.1 FA, Fractal Anisotropy 12
6.1.2 δ, Overdensity 13
6.2 Region Growing Algorithm with the FA and Overdensity 13
6.3 Region Growing Algorithm and FoF with the FA and Overdensity . . . 15
6.3.1 Influence of the FA in the seeds 15
6.3.2 Influence of the FA in the growth 15
6.3.3 Volume distribution 17
6.3.4 Shape of the regions 17
6.4 Comparisons with Laniakea 17

7 Discussions **17**

8 Conclusions **17**

9 Future Work **17**

A Approach by the Speed **18**
A.1 Region Growing Algorithm with the Speed 18
A.2 Results of the Speed Approach 19
A.3 Code example for Region Growing Algorithm - Approach by the Speed 26

3 Introduction

The Universe, at large scales, has a filamentary scructure. Large voids regions and galaxy superclusters are part of it. These structures can be identified visually. However, there are different algorithms to identify them based on physical criteria [2].

Due to gravitational attraction, velocity and density are related. It is possible to relate velocity fields with density fields. A proposal to find galaxy superclusters is to analyze the velocity field of galaxy. Matter tend to flow to the most dense region. In this way, spacial regions where the galaxy flow is convergent represent galaxy superclusters.

Recently a team build a galaxy velocities flow map of the local group in a scale of hundreds of light years [3]. In this map converging points were found and this team identified Laniakea, the galaxy supercluster which includes our galaxy, the milky way [1]. These observations are difficult to make for several reasons we will describe briefly. As a result of these difficulties, there is not a statistical quantity of galaxy superclusters obtained from the observational approach.

We propose to develop a method to detect a statistical significant number of galaxy supercluster in cosmological simulations. With this we aim to quantify if Laniakea can be considered as an atypical structure in the Universe.

4 Background

4.1 Cosmic flows and peculiar velocities

The peculiar velocity refers to the velocity relative to a rest frame.

In the case of the Cosmicflows-2 map, the rest frame is the earth.

A Wiener filtering method was apply to obtained a better Signal/Noise proportion and separate the peculiar velocities from the cosmic expansion.

4.2 Boundaries of Laniakea - The V-Web Algorithm

The contour of the region were reconstructed with the V-web Algorithm. This algorithm is deeply connected to the velocities as its core is the shear velocity tensor.

4.3 Fractional Anisotropy and Overdensity

4.4 The Current Model: Cosmological Parameters

5 Methods

Volker Springel's N-Body simulation software, Gadget-2 [4], is currently state-of-the-art in the field of cosmological simulations and is widely used in the scientific community. One example of use of Gadget is the Millennium Run [5], which has brought our understanding of our current model forward. We describe our numerical experiments in the section 5.1.

We devised different approaches for the detection of galaxy superclusters in the simulation data. One approach is based on the speed and is described in appendix A. We describe our main approach in section 5.2. It is based on the analysis of the eigenvalues of the local velocity shear tensor in the V-Web scheme. We use the Fractional Anisotropy and the trace of the shear tensor described in section 4.3.

We developed code in C and Python to treat the simulation data. The code is made available for the public in Github `https://github.com/sercharpak/Monografia/`

5.1 Simulations

We used the N-Body simulation software Gadget-2 [4] widely used in the scientific community to generate different cubic boxes of dark matter particles (DM). 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.

- Numerical Experiments

We are looking for structures of the same order of magnitude of Laniakea. Laniakea is, approximatively, a sphere of radius 80 Mpc/h. Our simulations need to be of at least the same order of magnitude. Next, we need to have a sufficient number of DM particles to be able to perform the Cloud in Cells described in section 5.2. Finally we want these simulations to run in a reasonable time. In table 1, we describe the 3 simulations we ran. We first ran the Medium simulation, of dimension 500 Mpc/h and 512^3 DM particles, as a first approach to this problem. To perform the CIC correctly we needed a more dense simulation. The result is the Small (Dense) simulation, of dimension 250 Mpc/h and the same 512^3 DM particles. Finally we ran a longer Medium (Dense), of dimension 500 Mpc/h and 1024^3 DM particles. This last simulation took several days to complete (80 hours). A bigger simulation would take several weeks and is to be performed in a future work.

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 Gadget-2 Simulations. We named them to easily refer to each one of them

By DM particle we mean a particle which only interacts through gravity interaction. In the simulation each particle represents a galaxy. The DM particles are placed in a 3D grid layout first and then are perturbed following a Poisson distribution before the simulation start, during the initial conditions generation. Once the simulation starts, the particles interact only with gravity

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

Ω_0	Ω_Λ	Ω	H
0.3	0.7	0	0.7

Table 2: Cosmological Constants in the Gadget-2 Simulations

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. For a geometrically flat universe, as ours, one has $\Omega_0 + \Omega_\Lambda = 1$. Ω is the baryon density in units of the critical density at red shift z = 0. In our case it is not relevant. Finally, H denotes the Hubble constant at red shift z=0 in units of 100 $km\ s^{-1}\ Mpc^{-1}$. It will be relevant in the discussions, section 7.

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

5.2 Approaches

In a first approach we analyse the distribution 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.

For our second approach (main one), we will use the following concepts: Cloud In Cell (CIC) method, the V- Web scheme (as described in section 4.2), the Fractional

Anisotropy (FA) (as described in section 4.3), the trace of the velocity shear tensor $tr\left(\sum_{\alpha\beta}\right)$ (as described in section 4.3) and the algorithms described in section 5.3.

- The Cloud In Cell (CIC) method

The Cloud In Cell (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 [6]. CIC is very popular among the scientific community [**?**]. 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). 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.

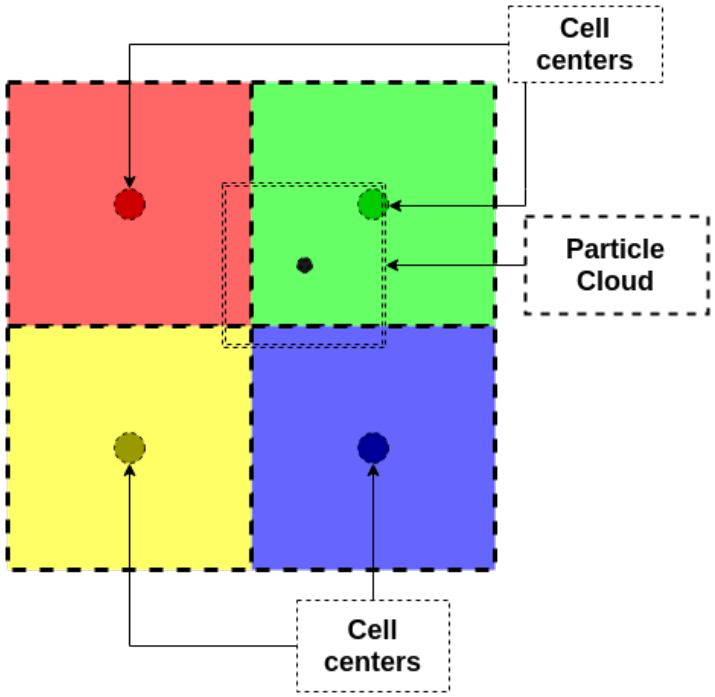

Figure 1: Explanation of the Cloud In Cell Method (CIC)

- The V-Web scheme

We described the V-Web scheme in section 4.2.

- The Fractional Anisotropy and the trace of $\sum_{\alpha\beta}$

As we saw in section 4.3 the FA and the trace of $\sum_{\alpha\beta}$ carry valuable information. The FA

The trace of $\sum_{\alpha\beta}$

In summary, our approach follows the steps:

1. Calculate the density field using CIC on the simulation data.
2. At the same time we calculate the velocity field using CIC on the simulation data.
3. Using finite differences we form the velocity shear tensor $\sum_{\alpha\beta}$.
4. We diagonalize it and find its eigenvalues λ_i and eigenvectors $\hat{u}_i$
5. We form the FA grid as described in section 4.3
6. We form the trace of $\sum_{\alpha\beta}$ grid.
7. We apply the algorithms described in section 5.3 to find the superclusters
8. We extract the properties (volume, shape, mass) from the superclusters
9. We compare the superclusters to Laniakea

Our approach is hence 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 Algorithms

Our main algorithm is a region growing algorithm as described by Gonzalez [7] mainly used in image segmentation. We now proceed to describe it generally.

5.3.1 Region Growing Algorithm: Basic Description

The region growing algorithm is used to identify different regions in a image. The aim is to identify regions which share 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 trace of the velocity shear tensor $tr\left(\sum_{\alpha\beta}\right)$. It is uses 4-connectivity as described before.

- Seeds

  We are looking for overdense structures. Therefore we will look for particles with $tr\sum_{\alpha\beta} \geq 1.0$. 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}	$tr\left(\sum_{\alpha\beta}\right)_{seed}$
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 $tr\left(\sum_{\alpha\beta}\right)$ 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 $tr\left(\sum_{\alpha\beta}\right)$. 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:

$$Q : (FA(x,y,z) \leq Thresh_{FA}) \text{ AND } \left(\delta(x,y,z) \geq Thresh_{tr\left(\sum_{\alpha\beta}\right)}\right)$$

  As we are looking for overdense regions (no voids) we define the threshold for $tr\left(\sum_{\alpha\beta}\right)$ as 0.0. As we are exploring the different values for the FA. each run will follow a row of the table 4.

FA_{growth}	$tr\left(\sum_{\alpha\beta}\right)_{growth}$
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 $tr\left(\sum_{\alpha\beta}\right)$ for the growth of the regions

A simplified version without the border cases management can be seen in appendix **??**. Results of this implementation can be seen in section 6.2.

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

6.1 General visualizations (FA, δ) of the Simulations

6.1.1 FA, Fractal Anisotropy

Using our small and dense simulation described in table 1, after the CIC (in a 256^3 grid) and V-Web have been applied, we can form the FA and visualize its histogram (Figure 2) and a 2-D cut (Figure 3).

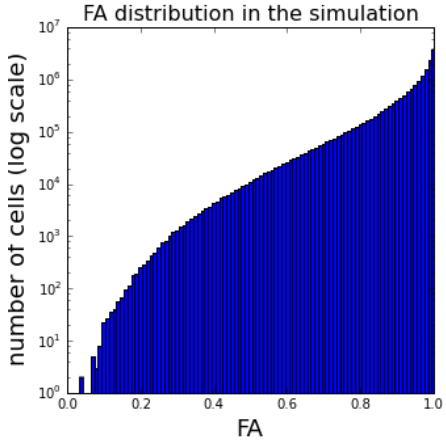

Figure 2: FA Histogram of the Small and Dense simulation

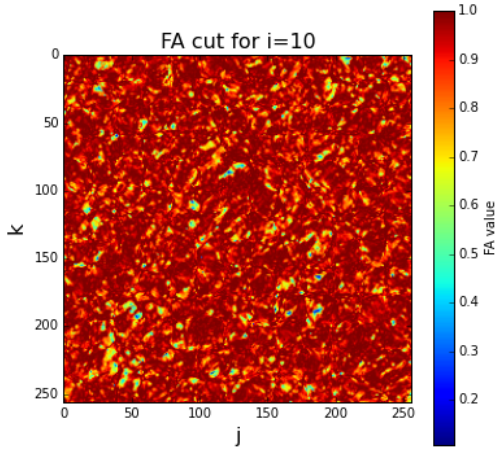

Figure 3: A cut of the FA in the x axis

As we saw in section 4.3, you would expect to have few highly isotropic particles. The distribution 2 we obtained agrees with it.
Observing a 2-D cut of the FA 3 we observe the filamentary structures we would expect

of the Cosmic Web. This cut is very similar to the ones presented by Bustamante et al. [8] in their work. This is a good indicator that our simulation, CIC procedure V-Web application and FA forming were correctly executed. Bustamante et. al explain that void regions should have a FA ¡ 0.95. We expect the similar FA values for superclusters and will make use of the Overdensity δ to discriminate the regions and make sure we do not find voids.

6.1.2 δ, Overdensity

As we did for the FA, we visualize the distribution (Figure 4) and a 2-D cut (Figure 5)of δ for the small and dense simulation of table 1.

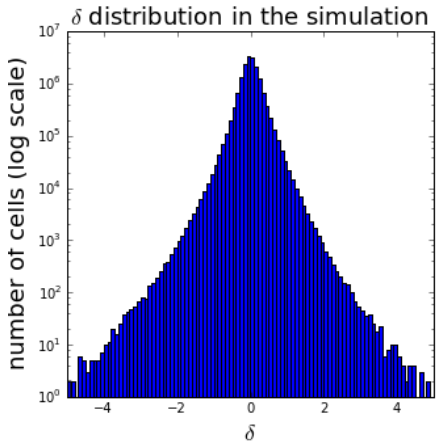

Figure 4: δ Histogram of the Small and Dense simulation

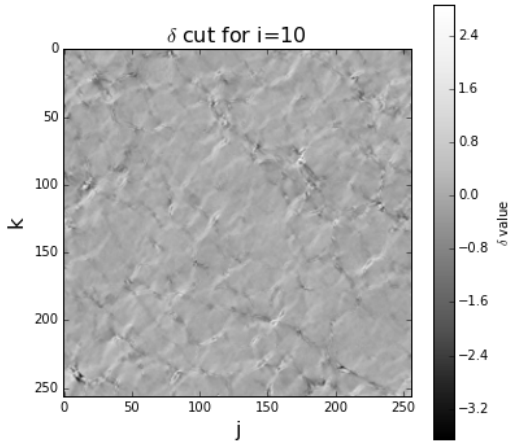

Figure 5: A cut of the δ in the x axis

The overdensity δ 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 $\delta \geq 0.0$.
Observing a 2D cut of the overdensity 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, $\delta \leq 1.0$) and highly overdense regions ($\delta \geq 1.0$) which are our candidates for seeds of supercluster structures.

6.2 Region Growing Algorithm with the FA and Overdensity

Using our small and dense simulation described in table 1, after the CIC (in a 256x256x256 grid) and V-Web have been applied, we ran the implementation described in section 5.3.2 for some of the values described in tables 3 and 4.

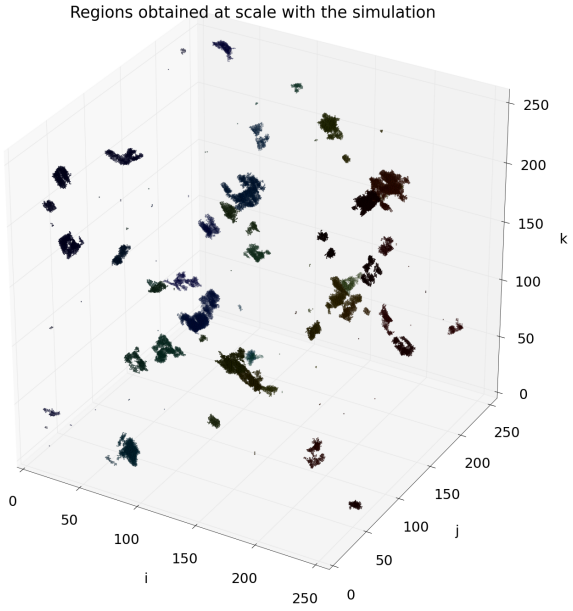

Figure 6: Regions detected for: Seeds (FA:0.6, δ: 1.0) and Growth (FA:0.8, δ: 0.0)

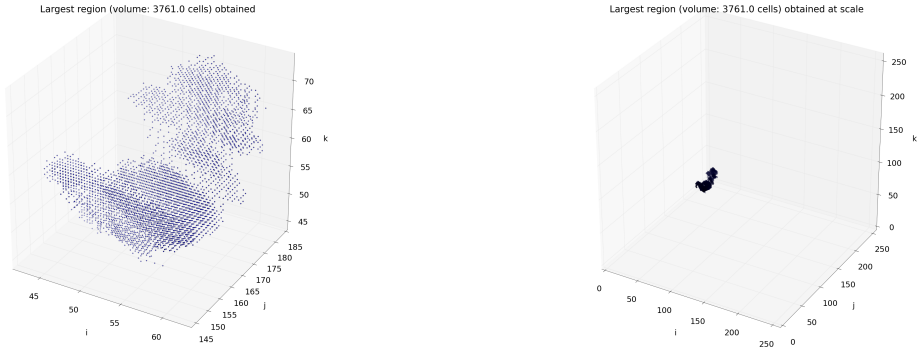

Figure 7: Largest volume detected for: Seeds (FA:0.6, δ: 1.0) and Growth (FA:0.8, δ: 0.0). The region is on the left and can be seen at scale with the simulation on the right.

6.3 Region Growing Algorithm and FoF with the FA and Overdensity

For the results of the small and dense simulation 1 we applied all the combinations for thresholds in tables 3 (seeds) and 4 (growth).

6.3.1 Influence of the FA in the seeds

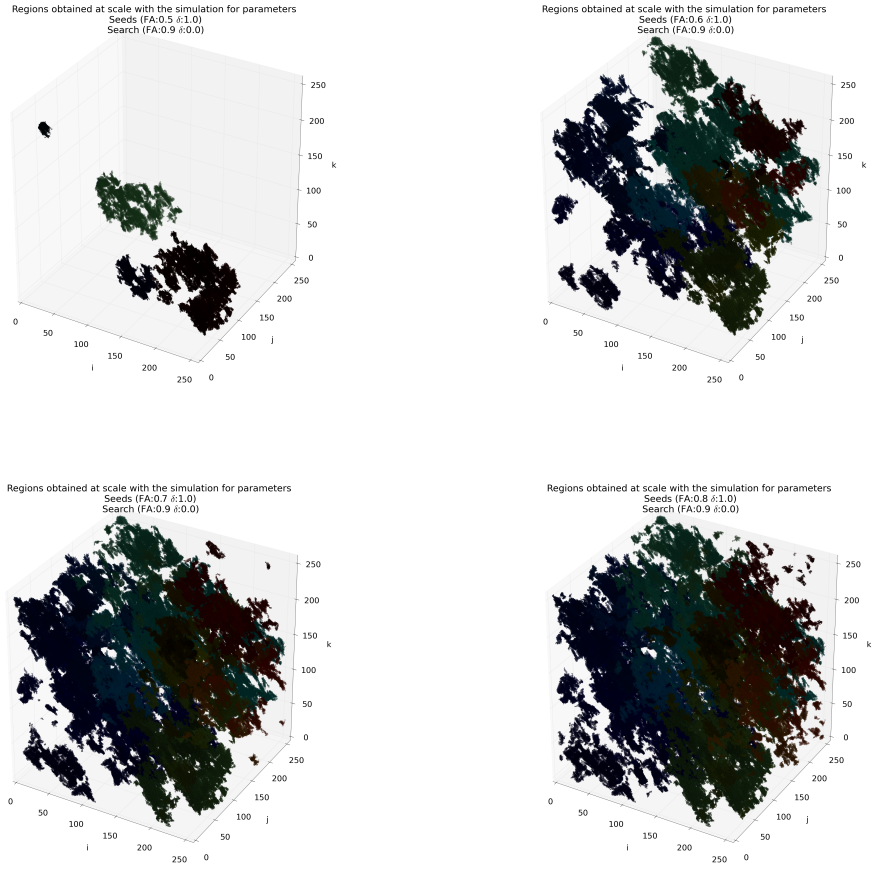

Figure 8: Efect of augmenting the FA threshold in the seed chosing. From upper left to down right the FA for the seeds are: 0.5, 0.6, 0.7 and 0.8. The growth FA is 0.9 and δ is 0.0. The δ for the seeds is 1.0.

6.3.2 Influence of the FA in the growth

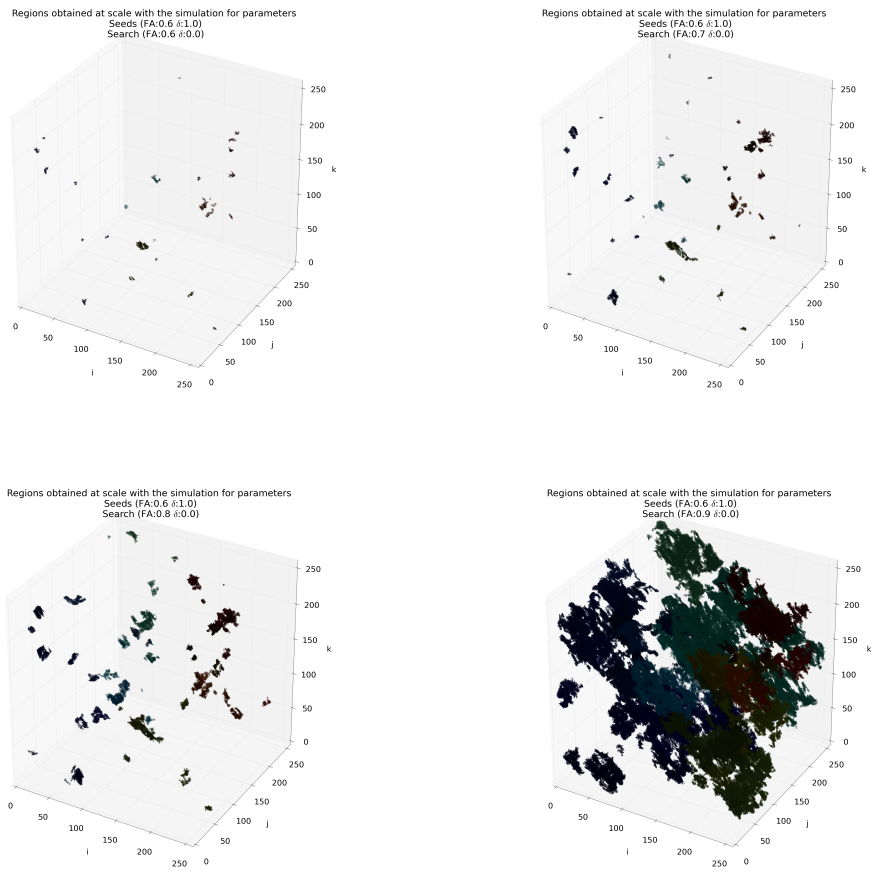

Figure 9: Efect of augmenting the FA threshold in the growth. From upper left to down right the FA for the growth is: 0.6, 0.7, 0.8 and 0.9. The seed FA is 0.6 and δ is 1.0. The δ for the growth is 0.0.

We can see in figure 9 how augmenting the FA thresholds for the growth changes drastically the volumes of the regions detected. This can also be seen comparing the possible growth space volume with the volume of the largest region detected ratio (figure 10).

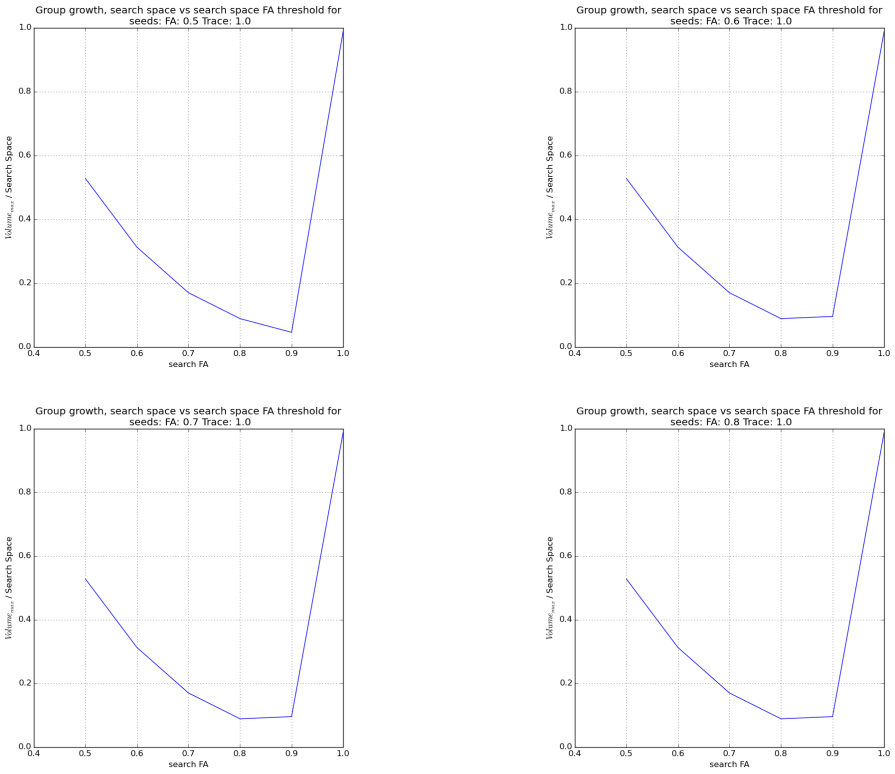

Figure 10: Volume of the largest structure and Volume of the growth space ratio augmenting the FA threshold in the growth. From upper left to down right the FA for the seeds is: 0.5, 0.6, 0.7 and 0.8. The growth δ is 1.0. The δ for the seed is 0.0.

6.3.3 Volume distribution

6.3.4 Shape of the regions

6.4 Comparisons with Laniakea

7 Discussions

8 Conclusions

9 Future Work

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

[5] 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.

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

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

[8] 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.

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

A Approach by the Speed

A.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 [7].

$f(x, y, z)$ 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 [9] to read the data and transform it to NumPy arrays in Python. In Appendix A.3 we attach the source code used.

A.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 11.

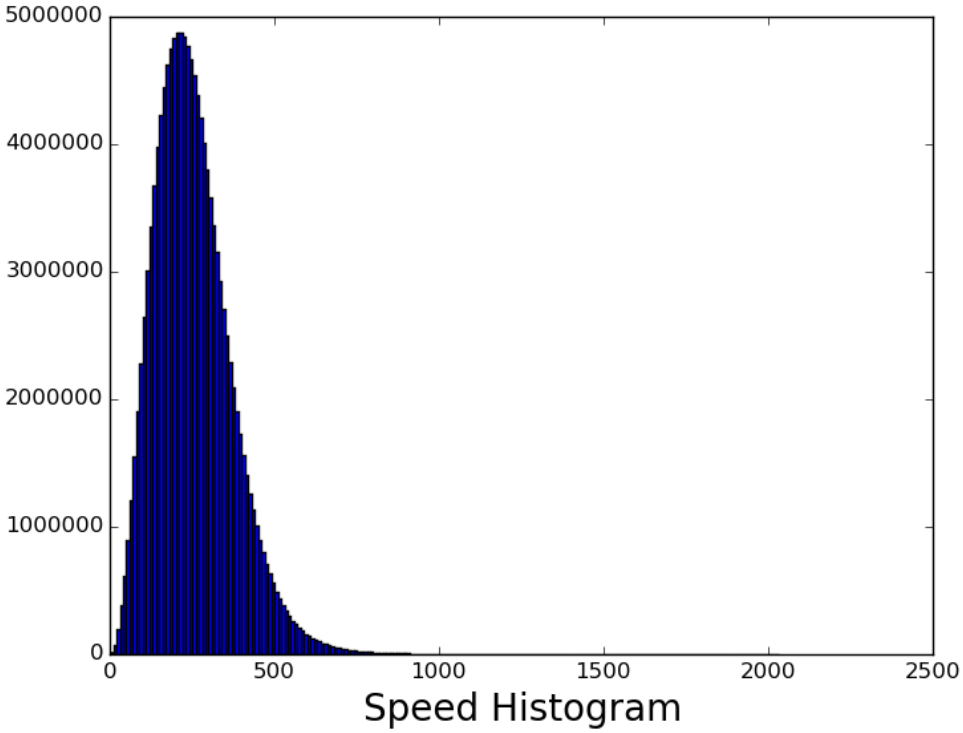

Figure 11: Speed Histogram of the simulation

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 5: 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.

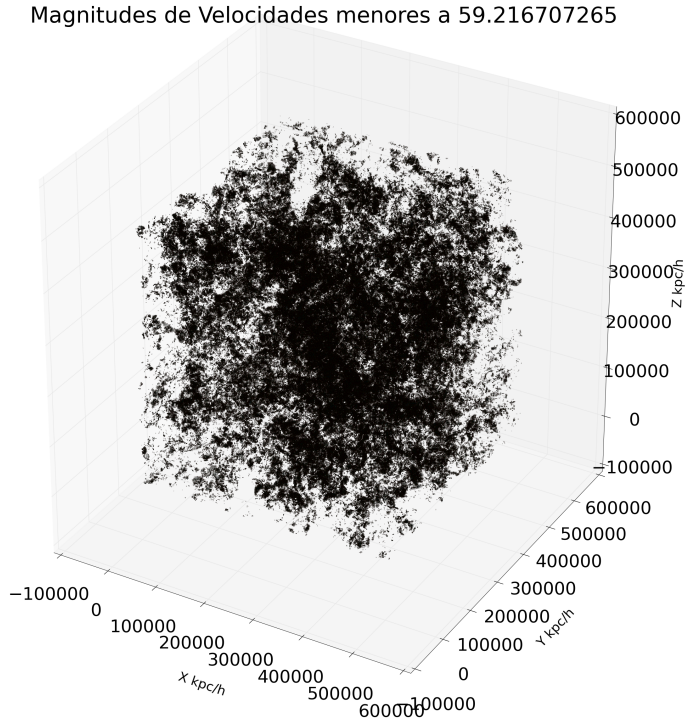

Figure 12: DM particles with $|v| < 59.2$

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.

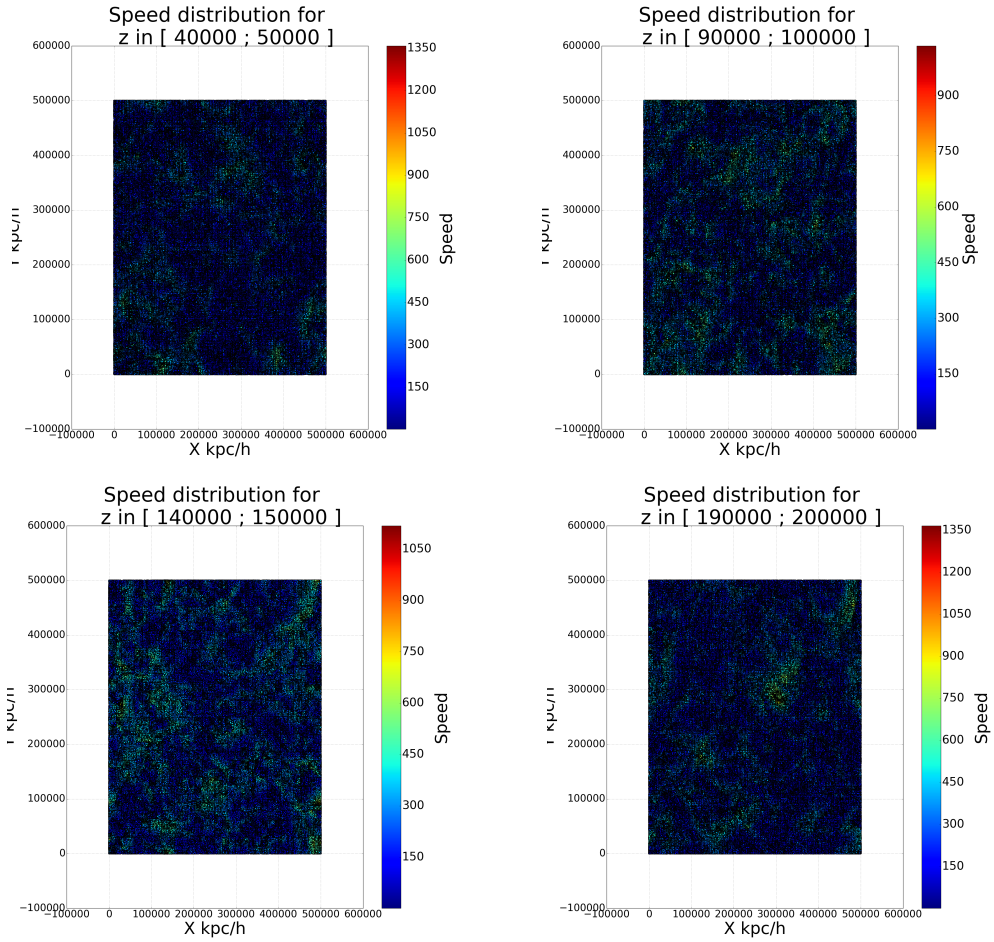

Figure 13: 2D slices of the speed distribution

We can observe in figure 13 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}.

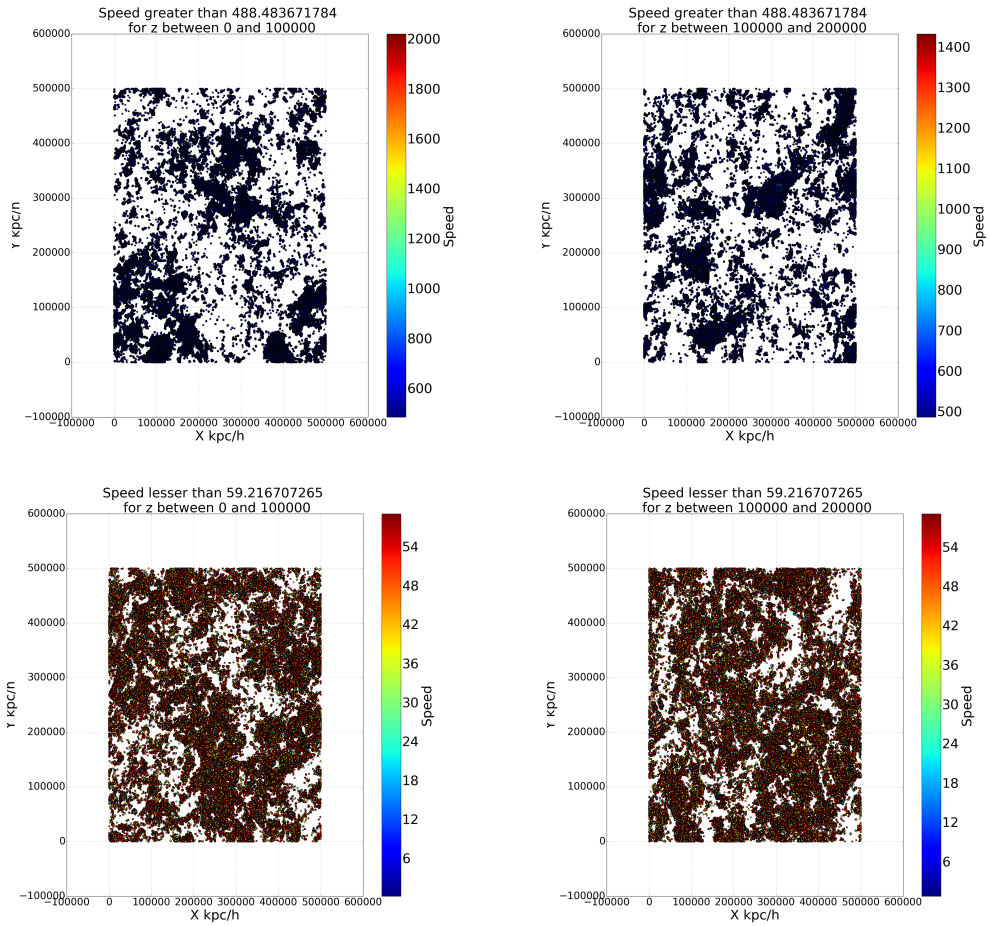

Figure 14: 2D slices of the speed distribution. The upper slices with $|\vec{v}| > th_{high} = 488$ and the two lower slices with $|\vec{v}| < th_{low} = 59$ for the same cuts, $z = \{100, 200\} Mpc/h$

As we can see in figure 14 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.

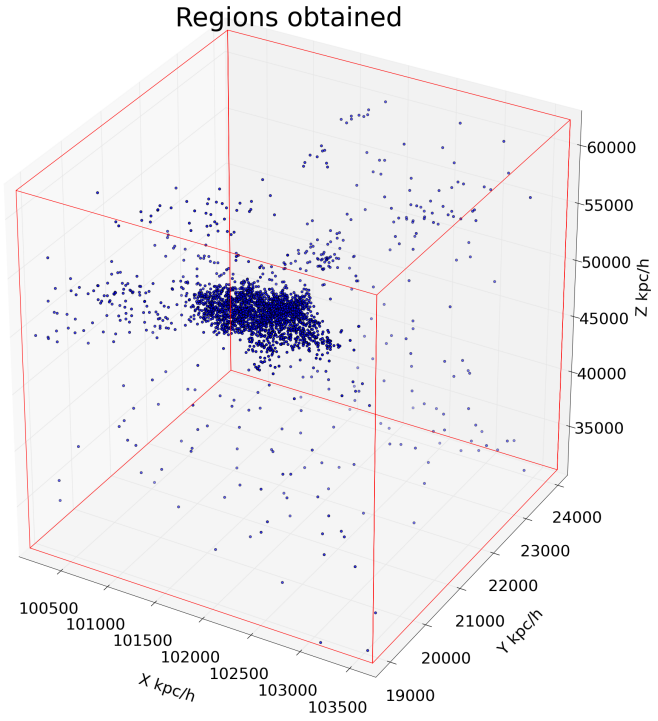

Figure 15: Region identified by the algorithm

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:

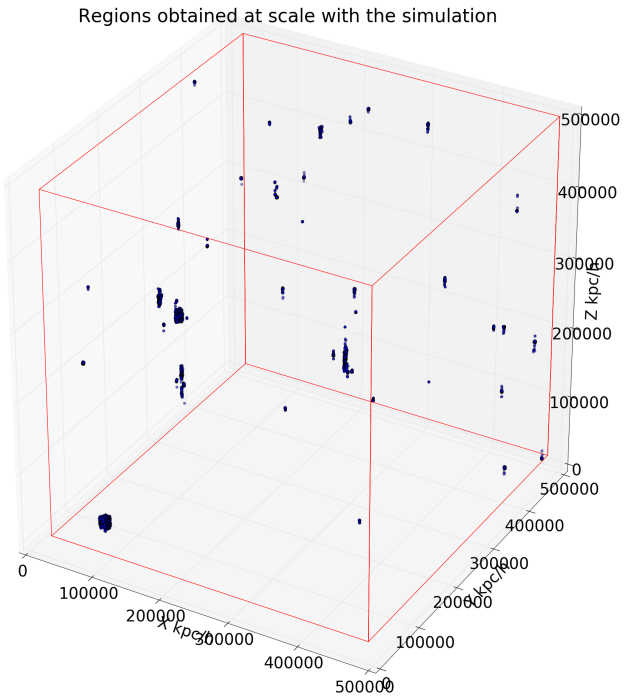

Figure 16: Region identified by the algorithm with the second thresholds

We have different observations:

1. The details of the regions in figures 15 and 16 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.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. results.

A.3 Code example for Region Growing Algorithm - Approach by the Speed

Listing 1: Region Growing Algorithm Code

```
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),:]
```