Internship report

Émile Trotignon

September 1, 2020

Introduction

The problem

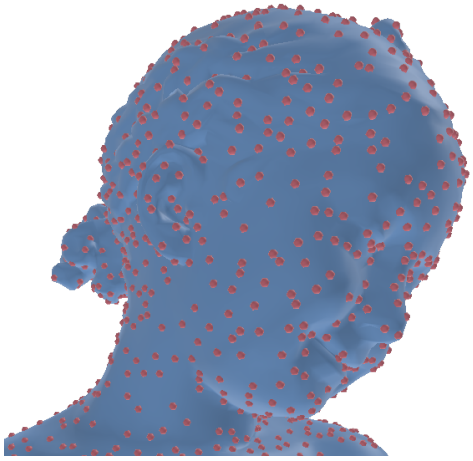


Figure 1: Sampled mesh of a woman's head

Sampling a surface with evenly spaced-out points is a very useful task in a lot of 3D modelling software, for instance physics simulations or rendering.

Tools to perform this task already exist, but they require a well-defined surface that is topologically correct (i.e. perfectly closed, and with no artifacts such as self-intersection). My goal during this internship was to explore more robust ways to perform sampling, that would have fewer requirements on the surface: we wanted to be able to sample a surface with defects such as self-intersection, or some holes in it, or even a triangle soup, with the constraint that the imperfect surface is an approximation of a topologically correct surface.

Our approach in this internship was to use winding numbers, a technique to robustly and efficiently determine if a point is inside or outside a surface.

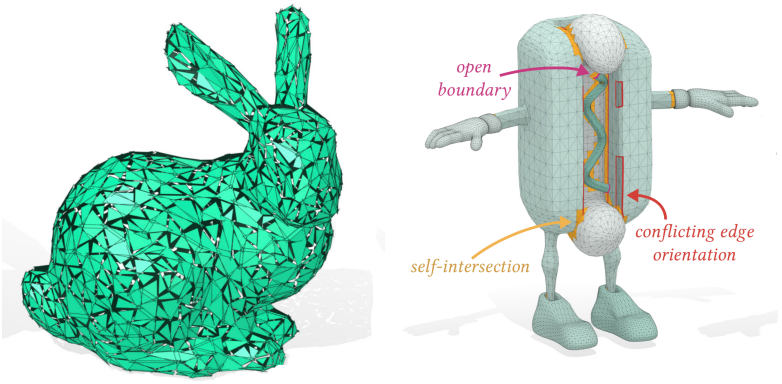


Figure 2: Meshes with defects. The rabbit on the left is a triangle soup, and the hotdog has the indicated defects.

Winding numbers

The principle of the winding number is to provide a way to mesure if a point is inside an oriented curve in the 2D plane. 3 It is an important and classic object of study in various fields such as algebraic topology, or vector calculus [Wik20b], but its use in geometrical processing date from a 2013 [JKS13] article generalising it for 3D meshes (i.e. a collection of vertices, edges and faces that define the boundary of a 3D shape).

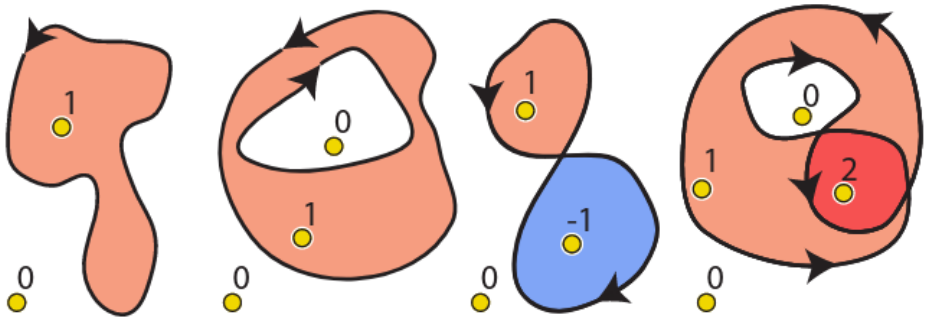


Figure 3: Winding numbers exactly segment inside and outside for concave, high-genus, inverted and overlapping curves. Multiple components are also naturally handled. Figure from [JKS13]

Definition [JKS13] :

> "The winding number is the signed length of the projection of a curve onto a circle at a given a point divided by 2π. Outside the curve, the projection cancels itself out. Inside, it measures one."

If the curve in 2D is perfecly closed, the winding number will be 1 inside the surface, 0.5 on the curve itself and 0 outside it.

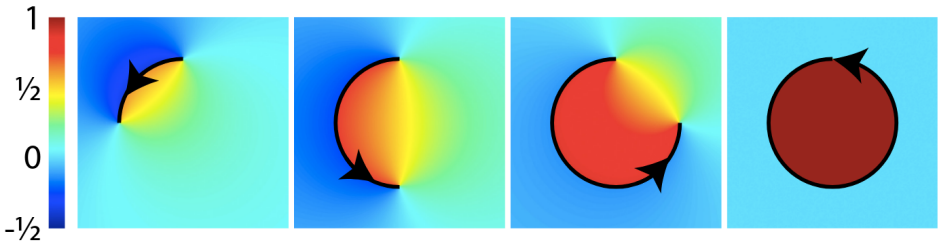


Figure 4: Winding number field of partial circles. Figure from [JKS13]

If the curve has holes, the winding number take values between 0 and 1 in the region around the hole. The winding number field is harmonic (i.e. twice continuously differentiable) everywhere except on the curve, as shown in figure 4.

It is possible to take an imperfect curve and repair it by defining the repaired curve as the isocurve at 0.5.

In 2013, an algorithm for efficient computation of the generalized winding number of a point with regards to a 3D mesh was published [JKS13].

A faster algorithm that works on arbitrary triangle soups and sets of dipoles (here, a dipole is a position and a normal) was published in 2018. [BDS+18]. This was made possible by noticing that the effect of a single triangle on the winding number field is similar to the effect of a dipole on an eletro-magnetic field: if there are a lot of small dipoles close to each other, the isosurface of the field will be very intricate, but from afar, the effect of the set of dipoles will be similar to the effect of a single, bigger one as shown in figure 5.

With a tree data structure that groups close elements together, this allows to compute the winding number of a triangle soup very efficiently.

The algorithms from these two papers are implemented in a library called libigl [JP+18], which I used to perform the computations.

Since the winding number has value 0.5 on the surface, what we want to sample is the isosurface at 0.5 of the winding number field.

1 Density sampling approach

Our first approach sampling a surface was to use an algorithm called Lloyd's relaxation.

1.1 Lloyd's relaxation

Lloyd's relaxation is an algorithm that distributes a set of points evenly inside a section of the plane, or a section of 3D space.

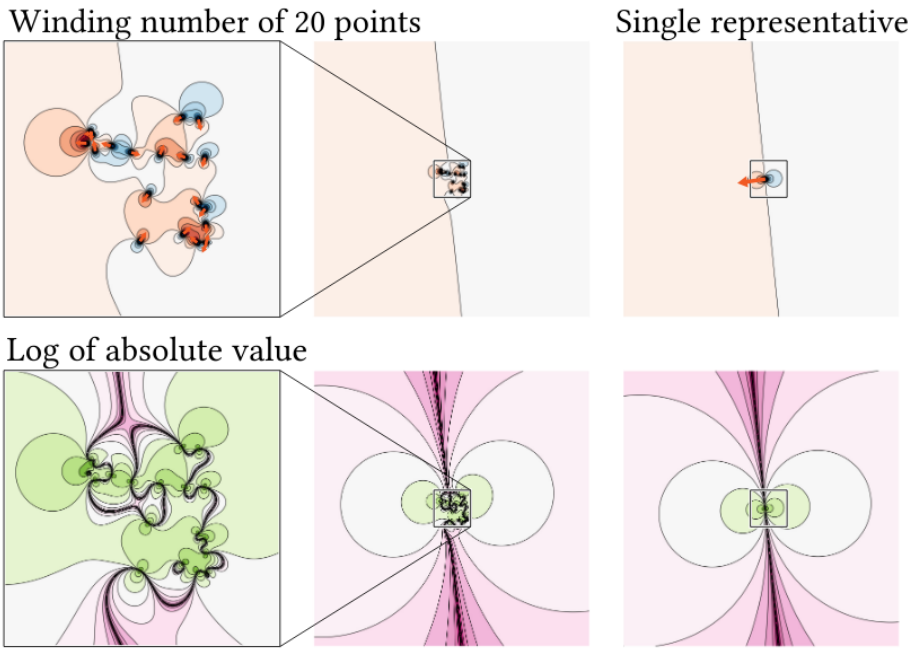


Figure 5: A cluster of 20 dipoles has an intricate winding number field nearby (left), but far away their function is quite tame (middle) and well approximated by a single, stronger dipole (right). Figure from [BDS+18]

It works by computing each point's Voronoi cell, and then advecting each point to its cell's centroid. After repeating this step multiple times, the set of points converges to an evenly spaced distribution.

Here, *e*venly spaced means that for V the point cloud and D the domain, Lloyd's relaxation gives you a new point cloud V' that minimizes the function

$$V \to \int_D d(x, V)^2 dx$$

where $d(x, V)$ is the distance between x and the closest point of V.[Llo82]

Its behavior can be tuned by considering that the cells do not have homogeneous density, and using a density function of your choice when computing the centroid.

If you use a density function δ the function minimized by Lloyd's relaxation is:

$$V \to \int_D d(x, V)^2 \delta(x) dx$$

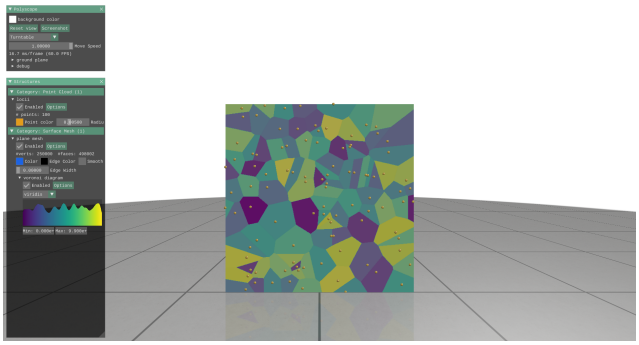


Figure 6: Voronoi cells in a unit square domain of a random set of points (the points are the yellow dots)

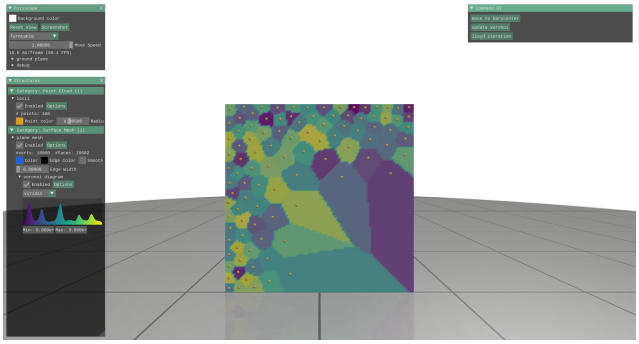


Figure 7: Lloyd's relaxion convergeance with density function $(x+y)^n$

Technical details

In order to get a proof of concept faster, we chose to use a discretization of the domain we worked on, and then compute the closest point for each point of the discretization. After this, the density function was computed on each point of the discretization, and then Lloyd's relaxation was performed.

If a final product was to exist, it would work by computing the geometry of the Voronoi cell directly, then compute the density function on the vertices of each cell, and computing the barycenter by interpolating the density.

1.2 Application to sampling

What we wanted to do was use a transformed winding number as a density function for Lloyd's relaxation, hoping that it could lead to uniform sampling.

First of all, we wanted to sample the surface of the mesh. The winding number has a value of 0.5 on the surface of the mesh. My first attempt at choosing a density function was picking a polynomial P such that $P(0) = 0$, $P(0.5) = 1$, $P(1) = 0$ and raising it to a high integer power.

This did not work because of the discrete winding numbers samples we use: almost no sample was of value 0.5, and if we lowered the power, the function would not be harsh enough to make the points converge to the isosurface.

A better way to do this was to use the norm of the gradient of the winding number. No matter the discrete sampling, the gradient would still be higher around the surface, since we computed it by comparing the winding number to the closest other samples.

After a few experiments, it was clear that while this was working, we needed to have some control over the distribution of the values, because a very harsh density function (here harsh means very high close to the isosurface and very low far from it) was needed and raising the gradient to a high power would not work past a certain point, because that would flatten values close to holes in the field, and that would lead to artifacts in the distribution, as shown in figure 9.

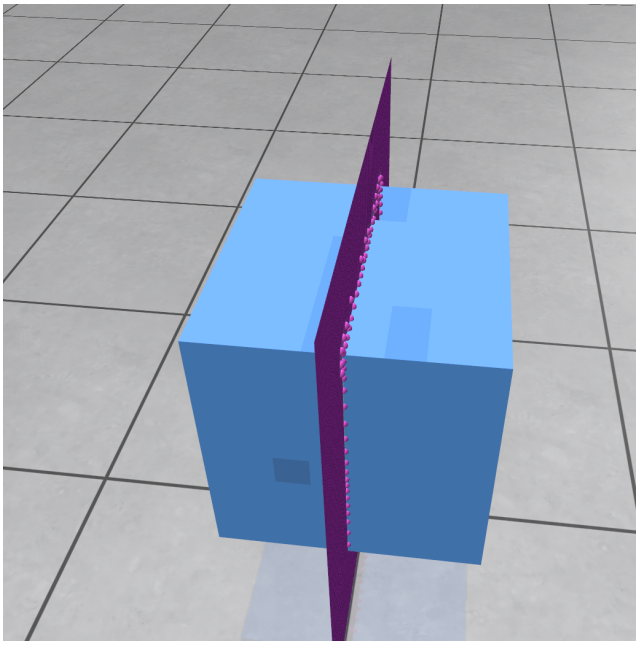
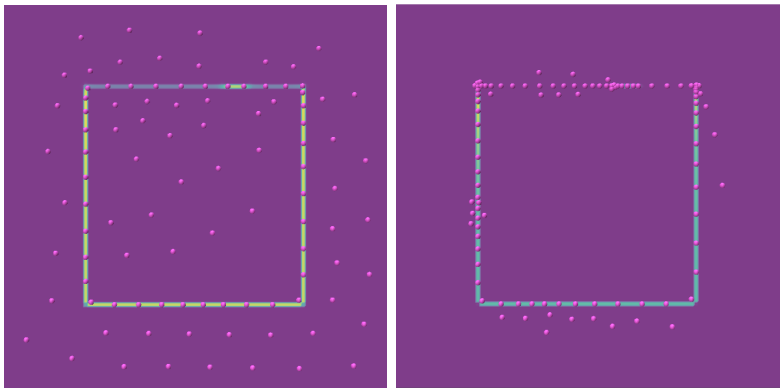


Figure 8: View of the mesh responsible for the winding number fields in figures 9, 10 and 11. Notice the two holes on the top of the cube that come in contact with the plane. They are the only ones relevant to the winding fields.

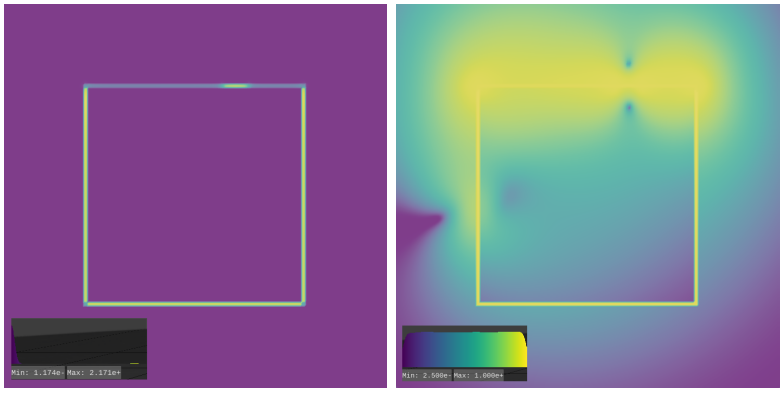
The solution I came up with was changing the values so that their order is maintained, but they are uniformly distributed in $[0; 1]$, as shown in figure 10. In order to do so I used an algorithm analogous to histogram equalization [Wik20a].

After this was done, we could raise every value to a very high inte-



(a) Power of 1 (b) Power of 5

Figure 9: Results after 20 iterations by raising the gradient norm to different powers. Notice that raised to power 1, the points are not on the surface at all, and raised to a power of 5, they are mostly on the surface or close to it, and there are big clusters at the beginning of the holes.



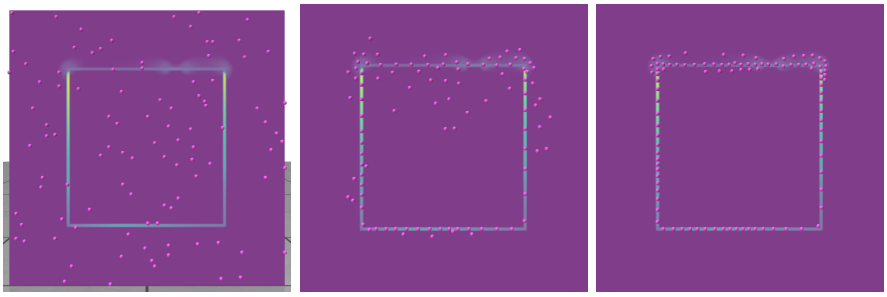
(a) Raw gradient (b) Uniform gradient

Figure 10: Comparison of the raw and the uniform winding number gradient norm. The distribution is in the corner.

ger power, without flattening certain regions, because the distribution was smoother. As shown in figure 11, the result is not perfect : the clean regions of the original surface are sampled, however the distribution is visibly not even. But more importantly, the points on the region of the hole are clustered and present in numerous layers, where we expected only one.

2 Gradient descent approach

After reaching the limits of the previous approach, we tried a completely different idea.



(a) Initial situation (b) After 5 steps (c) After 20 steps

Figure 11: Sampling result

An interesting property of the 2018 algorithm for fast winding numbers, is that it is possible to compute the winding number of a point with regards to a set of dipoles that represent points on the implied surface.

This gives us a way to measure how good a sampling is: use the difference between the winding number field of the surface and of the winding number field of the sampled points. This require to also have normals associated with the samples.

In order to get simpler formulas for dipoles, we shifted the winding number: The surface has winding 0, the inside 1, and the outside -1.

When this is the case, the winding number at x, with regards to point p and normal vector n is:

$$\frac{(x-p)\cdot n}{||x-p||^3}$$

In order to get the winding number with regards to a set of dipole, we just sum the winding number with regards to each dipole of the set. With D the set of dipoles :

$$\sum_{(p,n)\in D} \frac{(x-p)\cdot n}{||x-p||^3}$$

From here, we can define an error function e that we will try to minimize, with w_x the winding number implied by the mesh we try to sample: if D is a set of dipoles:

$$e(D) = \int_{\mathbb{R}^3} \left(w_x - \sum_{(p,n)\in D} \frac{(x-p)\cdot n}{||x-p||^3} \right)^2 dx$$

In order to minimize this function, we decided to derive it with regards to one dipole, which would allow us to perform a gradient descent. We chose to only try to make it work with one dipole at first on a very simple

field, because it would be simpler to program, and it would also make the tweaking of the paramaters easier.

If the discretized domain is Δ, the gradient of the error with regards to the position p of the current dipole is:

$$\frac{de(D)}{dp} = \frac{2e(D)}{||\Delta||} \sum_{x \in \Delta} \frac{1}{||x-p||^3} \left(n - 3\,(x-p) \cdot n \frac{x-p}{||x-p||^2} \right)$$

The gradient of the error with regards to the normal n of the current dipole is:

$$\frac{de(D)}{dn} = \frac{-2e(D)}{||\Delta||} \sum_{x \in \Delta} \frac{x-p}{||x-p||^3}$$

The error for one dipole is:

$$e'(p,n) = \int w_x - \left(\frac{(x-p) \cdot n}{||x-p||^3} \right)^2 dx$$

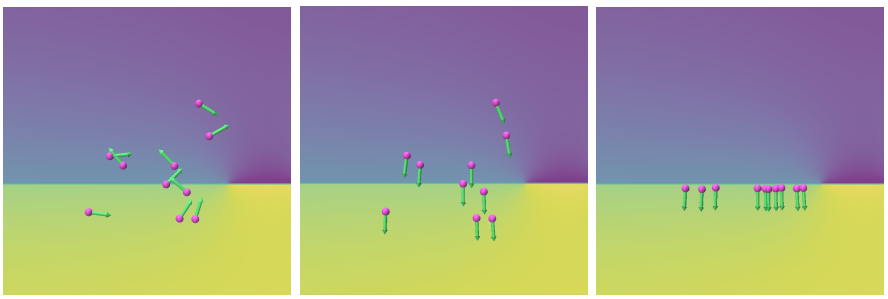
Its gradient relative to p is:

$$\frac{2e'(p,n)}{||\Delta||} \sum_{x \in \Delta} \frac{1}{||x-p||^3} \left(n - 3\,(x-p) \cdot n \times \frac{(x-p)}{||x-p||^2} \right)$$

and it gradient relative to n is:

$$\frac{de(D)}{dn} = \frac{-2e'(p,n)}{||\Delta||} \sum_{x \in \Delta} \frac{x-p}{||x-p||^3}$$

Using this formula, I was able to test if it would converge. After a lot of experiments, I was able to make the dipole converge on a very simple winding field as shown in 12. For the sake of simplicity, normals were normalized at each step of the algorithm. In order to get this result, it was very important to make the normal converge first. On a wrong normal, the gradient of the position is very unpredictable, and it often ejects the point from the domain.



(a) Initial situation (b) Converged normals (c) Converged positions

Figure 12: Independant optimisation of points

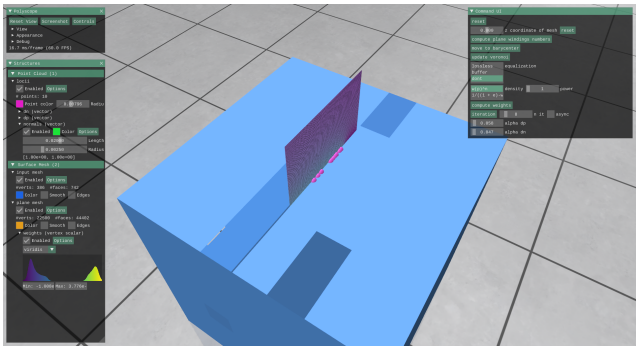


Figure 13: Origin of the winding field of figure 12

From here, we tried to optimize the points together. I made attempts with two points, but it was very difficult to make anything converge. At first, the movement was very erratic. After a lot of tweaking, only one point would converge, and the other would be "pushed" outside the domain.

The reason for that behavior is that I kept normalizing the normal vectors at each step of the algorithm, which was in fact very important: the length of the normal vector determines the strength of the dipole as shown in figure 14. If two strong dipoles are close to each other they will increase their winding field to a value higher than one, which will increase the error : in order to minimize the error, if you have two strong dipoles, it is better to place one dipole correctly and place the other one on the edge of the domain where its effect will be minimal.

After understanding this, I tried to add the norm back as a parameter, but that would prove way too volatile. I had minor success in optimizing two dipoles still having a fixed normal length, just smaller, but it was clear at that point that this approach was too volatile to be used as is.

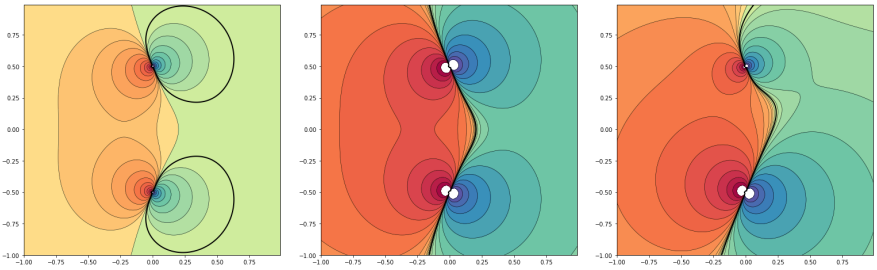


Figure 14: Each image shows in black the surface implied by the winding field generated by dipoles with different normal length

3 Gradient descent with Lloyd's relaxation

The idea to exploit the result of the previous part was to optimize each dipole separately, and once every dipole was on the surface, use Lloyd's relaxation to distribute them evenly.

In order to do so, we decided to switch the framework used for the computations. Until that point, the Voronoi cells used in the Lloyd's algorithm were computed according to a discretization of the domain: for each point of the discretization, we looked for the closest point, and stored the information.

However, there are faster algorithms that are able to give you a geometrical representation of the Voronoi cell. The boost of speed was going to be needed, because my code ran very slowly before that.

I was not able to complete this part of the internship: I ran into technical issues with the libraries I needed to use, which delayed my work, and then I did not have any time left.

4 Conclusion

The main result of this internship is that the two methods explored do not allow one to sample a mesh. However, I think the last approach, that I was not able to complete, is promising and deserves further investigation.

References

[BDS+18] Gavin Barill, Neil Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics*, 2018.

[JKS13] Alec Jacobson, Ladislav Kavan, and Olga Sorkine. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.*, 32(4), 2013.

[JP+18] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. `https://libigl.github.io/`.

[Llo82] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[Wik20a] Wikipedia contributors. Histogram equalization — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Histogram_equalization&oldid=974741741`, 2020. [Online; accessed 31-August-2020].

[Wik20b] Wikipedia contributors. Winding number — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Winding_number&oldid=966747580, 2020. [Online; accessed 30-August-2020].

Internship experience

My internship happened in the LIRIS laboratory that is attached to Université Claude-Bernard Lyon 1.

Due to the sanitary situation, I was not able to work on site. Instead, I had work remotely, which was made a lot easier by the tools my tutor David Cœurjolly had set up. We discussed my work daily in a Slack chatroom, and we had video conferences at least once a week.

I was supervised both by David and a colleague of his, Vincent Nivolier, and I want to thank them warmly for the work they put in in order to tutor me in these difficult conditions.