POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

A Computational Geometry approach for Machine Learning based diagnosis of nasal breathing difficulties aided by CFD

Laurea Magistrale in Aeronautical Engineering - Ingegneria Aeronautica

Author: Riccardo Margheritti

Advisor: Prof. Maurizio Quadrio

Co-advisor: Ing. Andrea Schillaci

Academic year: 2021-2022

---

1. Introduction

During the last years, the use of Machine Learning (ML) in fluid mechanics has really grown, thanks to the improvement in terms of available computational power and, most importantly, to the potential in a big number of applications.
ML is usually exploited to represent the non-linear relation between input and output (both CFD quantities) embedded in the highly non-linear equations of motion, but connecting a CFD input to a non-computable output of more general and abstract nature is truly challenging, since the label of a non-computable output does not allow a simple mathematical definition. One such problem is the classification of geometries through their effect on the flow field, and an example is the classification of nasal pathologies through their effect on the internal flow. The ear and throat surgeons face an impressive anatomical variability and can only rely on visual analysis via CT scan, while they cannot take advantage of fluid dynamics information that certainly hides the effect of the patholgies. The rate of success of the diagnoses can be surely improved if fluid dynamics information would be made available to ENT doctors.
However, CFD data is not immediately interpretable, sometimes even for a CFD expert, hence a tool able to analyze these data and return useful information to a surgeon would be hugely interesting. In this respect, ML could be vital.
The combination of ML with CFD is really challenging as the latter produces a huge amount of data that also show an enormous variability with respect to small changes in the geometries. This has also to be matched to the immense variability in intensity and shape that a pathology can show.

2. Objective

This work presents a possible and valid pipeline aimed to combine CFD with Machine Learning, adopting a Computational Geometry approach in dealing with geometries. The combination relies on a dimensionality reduction applied to CFD data to extract informative and compact features which feed the ML model, and on the data augmentation procedure applied to the available dataset.

The ultimate goal is to design a method able to effectively diagnose nasal breathing difficulties based on CFD data.

3. Method

In this section the pipeline that creates the dataset of the ML model is briefly described while applied to a new patient *P1*. This patient will be used to assess the performance of the model, trained on a 7-patients dataset, for which this procedure has been repeated.

3.1. Anatomy from CT-Scan

The starting point of the work is the geometry of the nasal cavities of a healthy patient, namely, a patient that shows no pathologies. With the help of Dr. Antonio Bulfamante (Hospital San Paolo in Milan), a database of roughly 150 patients, provided by *Azienda Ospedaliera San Paolo*, has been revised with the aim of finding such a patient. Eventually, patient *P1* has been diagnosed to show no pathologies and chosen for the purpose.
Starting from the CT-Scan of patient *P1* (see Figure 1), a segmentation is applied using the *Slicer TresholdEffect* tool, which allows to set the Hounsfield Unit (HU), distinguishing between air and biological tissues.

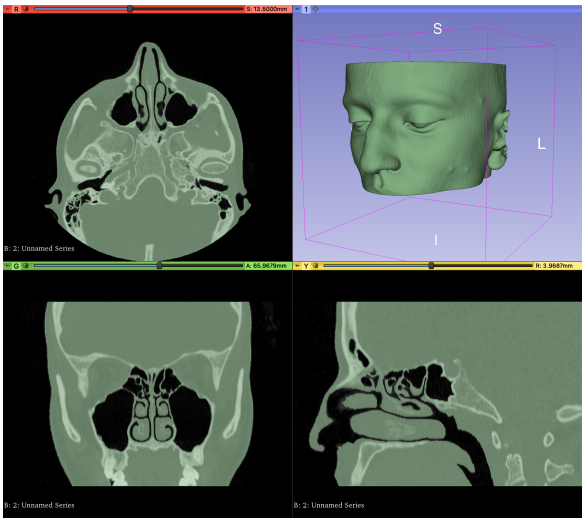


Figure 1: CT-Scan of patient *P1*.

The resulting STL geometry is then elaborated in *Blender* and *MeshLab* to correct small errors, such as non physical small edges and missing parts. The correct number of vertices and faces is set using *MeshLab*, taking into account the maximum amount of RAM memory available, resulting in a $\approx 25K$ vertices geometry.
An important intermediate step, is the alignment of this geometry with respect to a reference one, for which all the following steps were carried out by hand by experts. This reference geometry is the patient *P2* one. The alignment is performed thanks to the *Matlab procrustes* function, which implements the minimization of the sum of squared elements. This function minimizes the sum of the distances between landmarks, important and relevant points identified on the geometries. The alignment obtained using 17 landmarks is shown in Figure 2 and Figure 3.

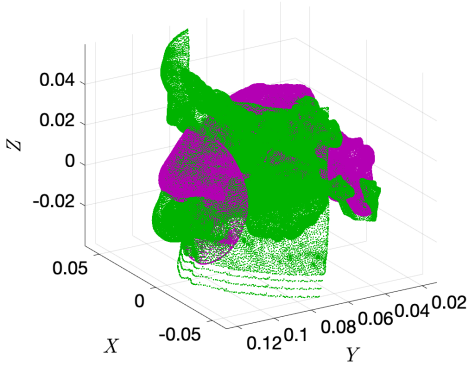


Figure 2: Geometries before the alignment. In green patient *P1*. In purple patient *P2*.

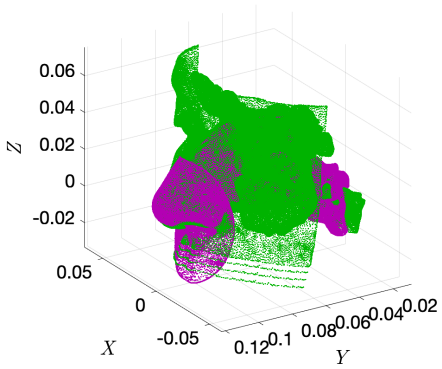


Figure 3: Geometries after the alignment. In green patient *P1*. In purple patient *P2*.

3.2. Shape registration and simplification

Fluid dynamics simulations are in general very computational expensive whereby the simplest possible geometry is required in order to re-

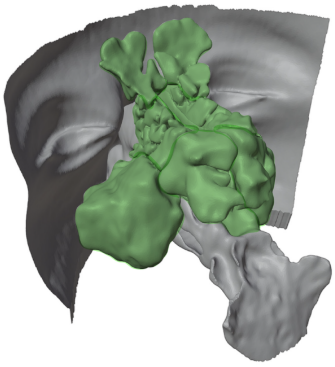


Figure 4: The paranasal sinuses are visible in green.

duce the running time and the computational resources needed. Patient *P1* still has unnecessary parts for a CFD simulation that are the paranasal sinuses (green parts in Figure 4), where the flow is so slow that can be neglected. A tool able to automatically identify and remove the sinuses would be of great help, avoiding doing that by hand for each patient, that would be incredibly time consuming. This tool is the functional correspondence and comes from the computational geometry.
The problem formulation is that of deformable object detection and dense correspondence in cluttered 3D scene presented in 2016 by Cosmo et al.[1] . The idea is to identify a linear operator (a functional map) between functional spaces defined over the shapes, enforcing descriptors preservation, along with landmarks and segment correspondences.
Input to the method are the *P2* shape (the model) and the *P1* shape (the scene) in which the model may appear up to deformation. The scene may contain additional clutter (here the sinuses); the model is clutter-free.
The goal is to determine a subset of the scene that is approximately isometric to some subregion of the model. The output of the method consists of the approximately isometric parts and a functional map encoding the correspondence between the parts. The two parts are represented as a binary indicator function on the respective shape, called segmentation function or segmentation mask. The result of the segmentation can bee seen in Figures 5 and 6, where the

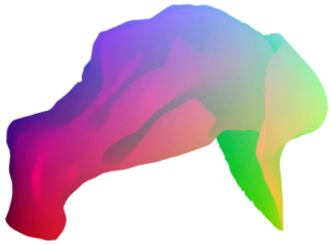


Figure 5: *P2* reference shape.

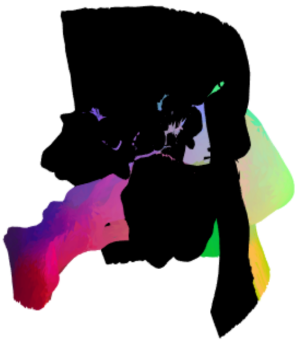


Figure 6: Registered *P1* shape. The segmentation masks contains zero values for the vertices depicted in black.

two nearly isometric parts are the colored ones.

3.3. ZoomOut

The next step is to compute a much higher resolution map between two shapes with respect to the one retrieved through shape registration. The map computed in Section 3.2 was in fact good for the matching of a shape into a cluttered one, well recognizing the model scene inside the target scene, but locally was not very accurate. The method considered is the one presented by Melzi et al.[2], based on spectral iterative up-sampling technique for non-rigid shape correspondence called ZoomOut method. The method receives as input a functional map $\mathcal{C}_0$ and at each step extends it to a new map $\mathcal{C}_1$, introducing additional frequencies and thus, intuitively adding samples for representing a map in the spectral domain.

Figure 7: The scalar map defined on the *P2* shape (top) is mapped on the *P1* shape (bottom) through the point to point map resulting from the ZoomOut method.

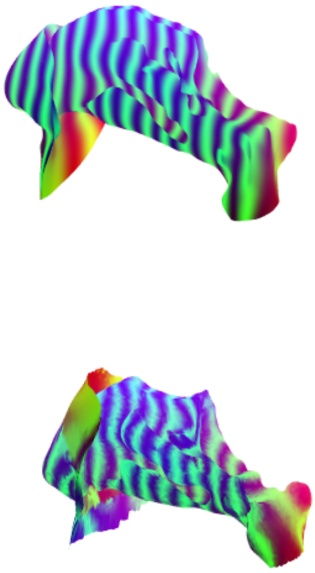


Figure 8: A high frequency sinusoid creates some bands that are mapped on the *P1* shape. The accuracy of the correspondence can be evaluated by looking at how these bands get deformed.

The results of the ZoomOut method can be seen in Figure 7 and Figure 8.

3.4. Pathologies insertion

The idea now is to use the "diseased" geometries of the patient *P2* for which the pathologies has been inserted by hand by ENT doctors, to map pathologies on the *P1* shape, aiming to extend the available "labeled" set of geometries, i.e. the diseased patients for which the pathologies are known. This procedure is called *Data Augmentation* and helps in improving the accuracy of a ML model, promoting variability in the training data. A deformation function is defined between the diseased and the healthy *P2* shapes, as the difference between the vertices coordinates (one should recall in fact, that a pathology is nothing more than a geometry deformation). This leads to a vectorial deformation function whose scalar component are mapped on the *P1* shape via the ZoomOut functional map, computed previously.
Repeating the mapping for 17 different pathologies (hypertrophies and septal deviations), new labeled diseased shapes of patient *P1* are obtained.

3.5. LES simulations

LES simulations are carried out on the resulting geometries, as accurate information of the internal is required. The simulations have been ran in OpenFOAM on a 0.65 seconds inspiration. The starting geometries are the *P1* STL files (with and without pathologies) and the STL of a cut sphere positioned to create a "closed mask" around the nostrils, necessary to enforce the inlet boundary conditions. 3 patches are generated, one for the sphere where the volumetric flow rate is imposed, one for the head representing the walls and one for the throat.
The mesh is created thanks to the *blockMesh* and *the snappyHexMesh* utilities. The first creates a background hexahedral mesh specifying the number of cells in all directions for each block generated; the two main blocks containing the background mesh can be seen in Figure 9. The second generates 3-dimensional meshes containing hexahedra that approximately conforms to the surface by iteratively refining. The snappyHexMesh *meshQualityControls* subdictionary helps in controlling the quality of the mesh, setting important values, as minimum determinant allowed, the maximum non orthogo-

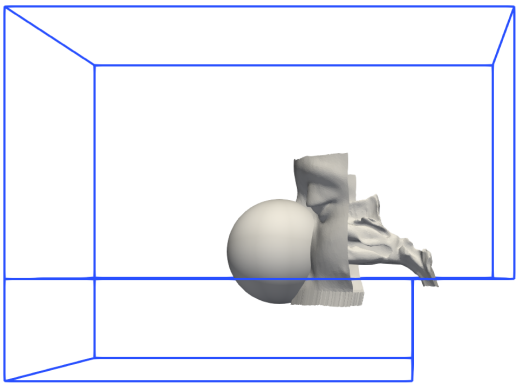


Figure 9: Main blocks exploited in the *blockMesh* utility.

Mesh characteristic values	
Parameter	*value*
Points	16788901
Faces	42114279
Cells	12954489
Min determinant cells	108

Table 1: Characteristic values for the "healthy" *P1* mesh. The Min determinant cells indicates the number of cell with determinant lower than the *minDeterminant* value of the *meshQualityControls* sub-dictionary.

nality, the minimum face area, etc... Some of the mesh characteristic values are reported in Table 1 for the *P1* "healthy" geometry.
The WALE turbulence model is set inside the *turbulenceProperties* dictionary, as requiring only local information, it is well-suited for LES in complex geometries; the solver is the *pimpleFoam* transient solver for incompressible, turbulent flow of Newtonian fluids.
The simulations are performed in remote ssh access to the server of the HPC infrastructure GALILEO100 of the CINECA system in Bologna on 2 out of the 528 total nodes for each simulation. Each node has 48 cores for a total of 96 cores and 160 GB's of RAM.

3.6. Features exctraction

Given the amount of data that CFD returns (around 40 GB's per simulation), feeding a neural network with such a huge dataset reveals currently an infeasible way to proceed, hence a selection and dimensionality reduction of the data is required. In particular, one can look for important and relevant values for the CFD, able to compactly encode the flow field properties, called *features* in a Machine Learning jargon.
The extraction of the features from the LES results is crucial. Features must contain information on the pathologies and at the same time should be easy to retrieve and to interpret. The features used within this work are inspired by engineering practice in the analysis of flow fields [3], namely *regional averages values of time-averaged quantities*. Six sections are identified on the *P1* shape (see Figure 10) on which such values are retrieved for the *enstrophy*, the *turbulent kinetic energy*, the *velocity magnitude* and the *magnitude of the pressure gradient*, distinguishing between left and right sub-region. The regional average is computed taking into account the uneven dimension of the cells, weighting the values on the cells area.
This results in 12 values per simulation for each CFD quantity previously mentioned. These values are used as input to a classification neural network.

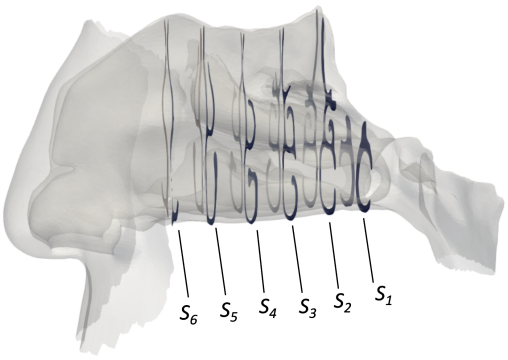


Figure 10: Sections extracted from LES results.

3.7. Neural network classification

A classifier is a particular Neural Network architecture able to predict if something belongs to one class or not. Specifically, the classifier built in *Matlab* for this project is trained to perform a binary classification, evaluating if a given set of features associates to a septal deviation or to a hypertrophy. The NN takes as input a 12-features vector, corresponding to the 12 regions extracted from each geometry, 2 regions per section. The output layer consists of one

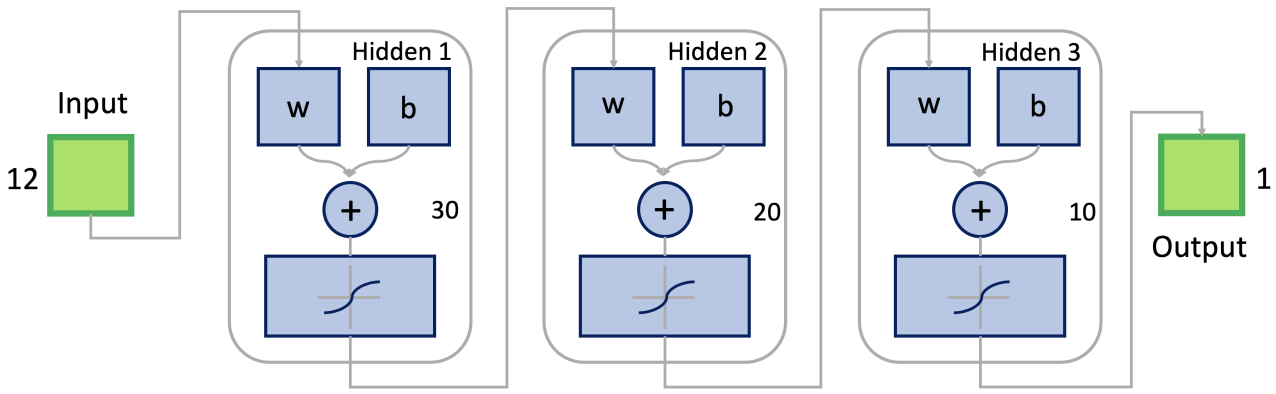


Figure 11: Classifier architecture.

Performance of the classifiers		
Network	*Score*	*Accuracy*
$\|\mathbf{U}\|$-based	13/17	77%
$\|\nabla p\|$-based	14/17	82%
E-based	12/17	71%
k-based	11/17	65%

Table 2: Performance of the classifiers. E indicates the enstrophy, k the turbulent kinetic energy.

neuron returning a value between 0 (associated to a hypertrophy) and 1 (associated to a septal deviation). Three fully connected hidden layers link the input to the output, having respectively 30, 20 and 10 neurons each (see Figure 11) for a total of 1231 trainable parameters.
The neurons implement the *hyperbolic tangent* activation function and the *Loss function* is the classical *binary cross-entropy*. The training is is managed by the *Scaled Conjugate Gradient Backpropagation* that updates the network parameters considering the sensitivity that the loss function shows with respect to each parameter. The tests are conducted on a *never-seen-before* patient, that is the *P1*, to evaluate the performance in the inference on never-seen-before data. 4 different networks are considered, each one trained on the values of the 4 CFD quantities previously cited. The results can be seen in Table 2.

4. Conclusions

Given the limited available dataset, the performance achieved with this pipeline is quite good, up to a $\sim 80\%$ of accuracy in the best case. This demonstrates that ML can effectively predict functional properties (here the type of pathology) from complex CFD data, even if the flow field does not provide direct high-level diagnostic information. The potential is huge, as with such a tool, ENT doctors could rely on new relevant information directly conveyed by the flow field, that can improve the rate of success of their diagnosis.

References

[1] L. Cosmo, E. Rodolà, J. Masci, A. Torsello, and M.M Bronstein. Matching deformable objects in clutter. *Fourth International Conference on 3D Vision*, 2016.

[2] S. Melzi, J. Ren, K. Rodolà, A. Sharma, M. Ovsjanikov, and P. Wonka. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics, Association for Computing Machinery*, 2019.

[3] M. Quadrio, G. Boracchi, A. Schillaci, M. Restelli, and C. Pipolo. Inferring functional properties from fluid dynamics features. *25th International Conference on Pattern Recognition (ICPR)*, 2020.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A Computational Geometry approach for Machine Learning based diagnosis of nasal breathing difficulties aided by CFD

Tesi di Laurea Magistrale in
Aeronaitical Engineering - Ingegneria Aeronautica

Author: **Riccardo Margheritti**

Student ID: 966594
Advisor: Prof. Maurizio Quadrio
Co-advisor: Ing. Andrea Schillaci
Academic Year: 2021-22

Abstract

The present Thesis is carried out within the framework of a standing collaboration, provisionally named *OpenNOSE*, led by PoliMi, which includes UNIMI with the *Santi Paolo e Carlo* University Hospitals.

The work presents the pipeline that has been developed during the recent years, aimed at diagnosing nasal breathing difficulties through a Machine Learning (ML) model. Specifically, the ML model is designed to exploit CFD information extracted from simulations of patient-specific anatomies with pathologies. In dealing with shapes, a Computational Geometry (CG) approach has been adopted to automate some complicated procedures, and to perform Data Augmentation (DA) of the available dataset while ensuring consistent and well defined pathologies with unique labels.

The work is structured in three parts. In the first, starting from the complete CT scan of a healthy patient, a simplified geometry of the nasal cavities is extracted thanks to a CG tool, known as functional maps. The ability of the functional maps to create functional correspondences helps in performing DA, by increasing the number of available anatomies for which LES simulations are carried out. The second part describes the computational procedure for the LES simulations; the third part describes how CFD results can be compacted into a handful of values (called features) and used as input on a (pre-trained) Neural Network (NN), which performs inference on the pathologies.

The pipeline has been previously trained on a database built with DA from 7 patients. The present work has provided data based on a 8th patient, used to test for the first time the accuracy of the classifier. Although the number of observations is still rather limited, the achieved accuracy of 80% is already satisfactory, and demonstrates the viability of the proposed approach.

Keywords: CFD, LES, Machine Learning, Computational Geometry, Functional Maps, Data Augmentation, Nasal Cavities Flow

Abstract in lingua italiana

Collocato all'interno del progetto *OpenNOSE*, una collaborazione aperta tra PoliMI, UNIMI e l'*Ospedale San Paolo polo universitario* di Milano, il lavoro che segue presenta la pipeline sviluppata negli ultimi anni con l'obbiettivo di diagnosticare difficoltà respiratorie attraverso l'utilizzo di un modello Machine Learning (ML). In particolare il modello utilizza dati CFD estratti da simulazioni condotte sulle geometrie delle cavità nasali dei pazienti. Nel trattare le geometrie, viene adottato un approccio basato sulla Geometria Computazionale (GC) per automatizzare delle procedure altrimenti complicate, e per fare Data Augmentation (DA) sul dataset disponibile.

Il lavoro è strutturato in 3 parti: nella prima, partendo dalla TAC di un paziente sano viene estratta una geometria semplificata delle vie nasali tramite l'utilizzo di mappe funzionali, uno strumento appartenente alla GC; la corrispondenza funzionale permette inoltre di fare DA, ovvero incrementare il numero di geometrie disponibili su cui lanciare simulazioni, promuovendo la variabilità anatomica, operazione fondamentale per migliorare l'accuratezza di una rete neurale, sopratutto in ambito CFD. Nella seconda parte, simulazioni LES sono lanciate sulle geometrie risultanti dalla DA. Nella parte finale, particolari valori chiamati features vengono estratti dai risultati delle LES e utilizzati per testare una rete neurale (NN) di classificazione, addestrata a fare inferenza sulle patologie associate alle geometrie dei pazienti.

La pipeline ha prodotto finora un database di 7 pazienti e con questo lavoro ne viene inserito un ottavo, dopo essere stato utilizzato per testare l'accuratezza del classificatore, che si assesta attorno ad un buon 80%.

Parole chiave: CFD, LES, Machine Learning, Geometria Computazionale, Mappe Funzionali, Data Augmentation, Corrente Nelle Cavità Nasali

Contents

Introduction

In the recent years, the use of Machine Learning (ML) in fluid mechanics has grown exponentially, thanks to the improvement in terms of available computational power and, most importantly, to the potential in a number of applications [1]. ML is usually exploited to represent the non-linear relation between input and output (both CFD quantities) embedded in the highly nonlinear equations of motion. Within this approach, a host of works have proposed: turbulence model improvement through ML [2]; reconstruction, estimation and super-resolution using Convolutional Neural Networks (CNN) [3]; physics-informed deep learning [4], and several others. The field is really blooming. In all the aforementioned applications, fluid dynamics quantities are used as input and output of the ML algorithm. Connecting a CFD input to a non-computable output of more general and abstract nature (that is a high functional property) is more challenging, since the label of a non-computable output does not allow a simple mathematical definition. One such problem is the classification of geometries through their effect on the flow field: an example is the classification of nasal pathologies through their effect on the internal flow, which is the original application where the approach was first conceived [5].

ML in rhinology is almost unexplored, being so far limited mainly to the classification of paranasal sinuses based on radiological information. In the last two decades, the number of patient-specific CFD studies has seen a sudden growth; however, very little is flowing back to clinical practice in terms of improved patient treatment, better surgery planning, reduced failure rate of surgeries. The ear and throat surgeons face an impressive anatomical variability and can only rely on visual analysis via CT scan, while they cannot take advantage of fluid dynamics information, and ground their diagnosis on experience and on the reported symptomatology. This entails a very subjective clinical path that often leads to unsatisfactory surgical maneuvers: as an example, for the surgical correction of the septal deviations the majority of patients reports an unsatisfactory outcome after surgery [6]. The rate of success of the diagnoses can be surely improved if fluid dynamics information would be made available to ENT doctors.

However, the native form of CFD data is not immediately interpretable [7], sometimes even for a CFD expert, hence a tool is required to analyze these data and return useful

information to a surgeon. In this respect, ML could be vital.

The combination of ML with CFD is really challenging as the latter produces a huge amount of data that also show an enormous variability with respect to small changes in the geometries. This has also to be matched to the immense variability in intensity and shape that a pathology can show. Thus, a valid Machine Learning model should be trained on a really big dataset of simulations, difficult to gather given the lack of annotated data. A possible walk around could be to perform data augmentation on the available dataset, namely, increasing the number of geometries on which a network is trained. These aspects, will be further deepened in the following pages.

Therefore, this work wants to present a possible and valid pipeline (schematized in Figure 1) aimed to combine CFD with Machine Learning, adopting a Computational Geometry approach in dealing with geometries. The combination relies on a dimensionality reduction applied to CFD data to extract informative and compact features which feed the ML model, and on the data augmentation procedure applied to the available dataset.

The ultimate goal is to design a method able to effectively diagnose nasal breathing difficulties based on CFD data.

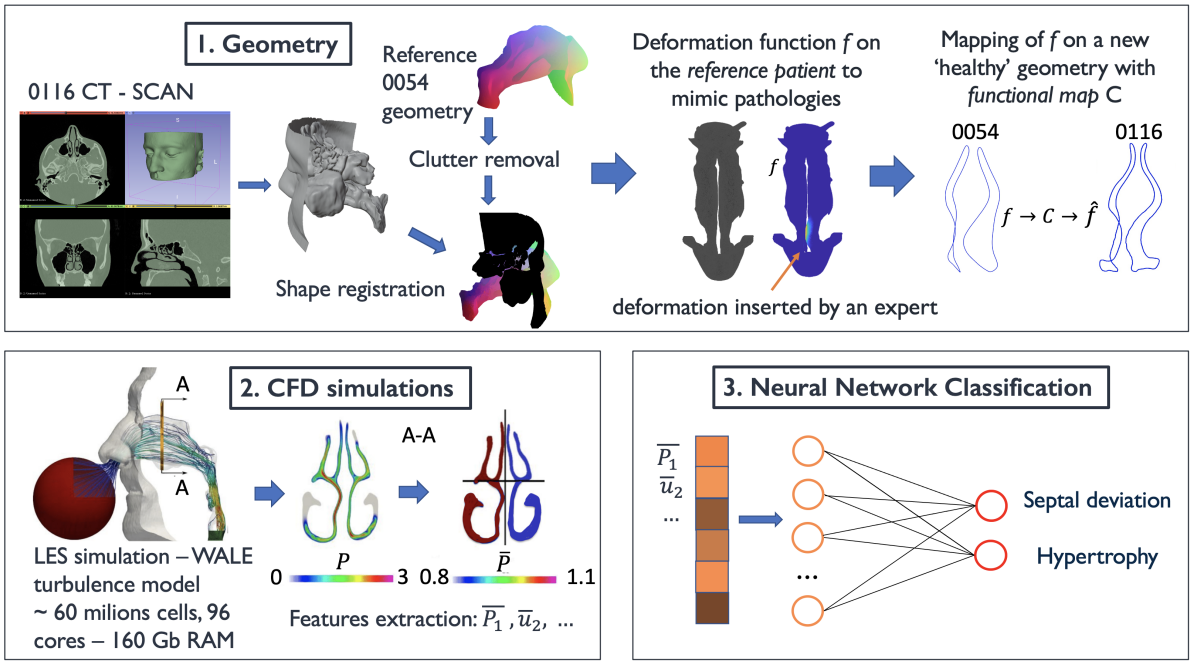


Figure 1: Scheme of the work.

1 | Anatomy from CT-Scan

1.1. CT-scan

The starting point of this work is the geometry of a healthy patient (i.e. with no pathologies), on which the method presented in this work will be applied; whereby a database of roughly 150 CT-scans provided by *Ospedale San Paolo polo universitario* in Milan has been revised with the aim of finding such a patient. Under the supervision of Dr. Antonio Bulfamante (hospital San Paolo), patient *P1* has been diagnosed to be healthy and chosen for the purpose (Figure 1.1).

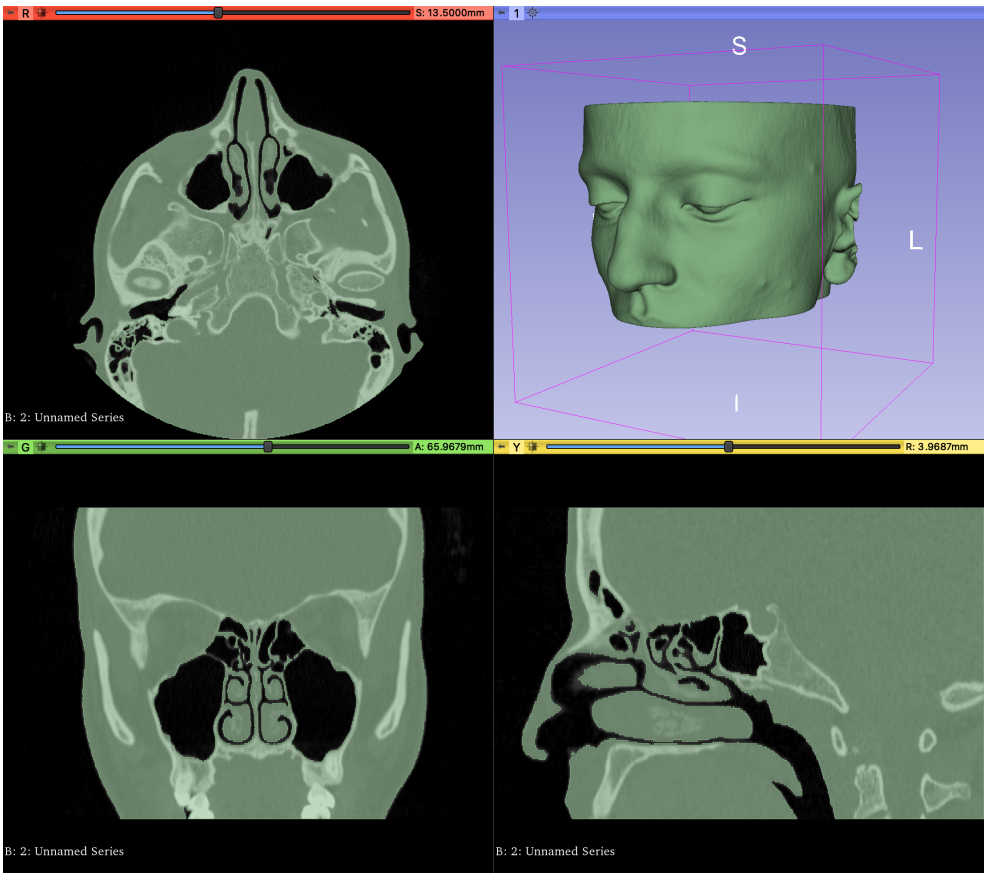


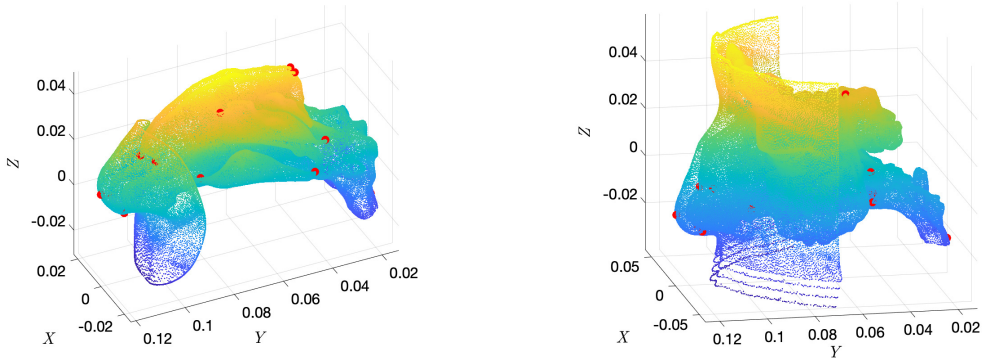
Figure 1.1: CT scan of patient *P1*. Top left: transversal plane; top right: 3D model; Bottom left: coronal view; Bottom right: sagittal view.

The generation of the preliminary STL geometry is achieved as follows: The DICOM data is imported in *Slicer* and a segmentation is applied with the help of the Treshold-Effect tool, which allows to set the Hounsfield Unit (HU), distinguishing between air and biological tissues. This value of threshold has been fixed to -220 HU [8]. After that, the segmentation created is checked looking for non physical small edges that are critical for the CFD analysis, for missing parts and for filled parts that needs to be excluded by hands. Once adjusted, the STL can be exported and it's ready for further corrections in *Blender* and *MeshLab*.

The *Blender* environment helps in removing unnecessary parts of the geometry, in removing small edges and adjusting errors. *MeshLab* allows to get a good number of faces and vertices for the following steps developed in *Matlab*. In particular, as 16 GB's of RAM were available, a surface mesh with a maximum of $\approx$ 25000 vertices were allowed to be stored in the *Matlab* workspace.

1.2. Alignment with reference patient

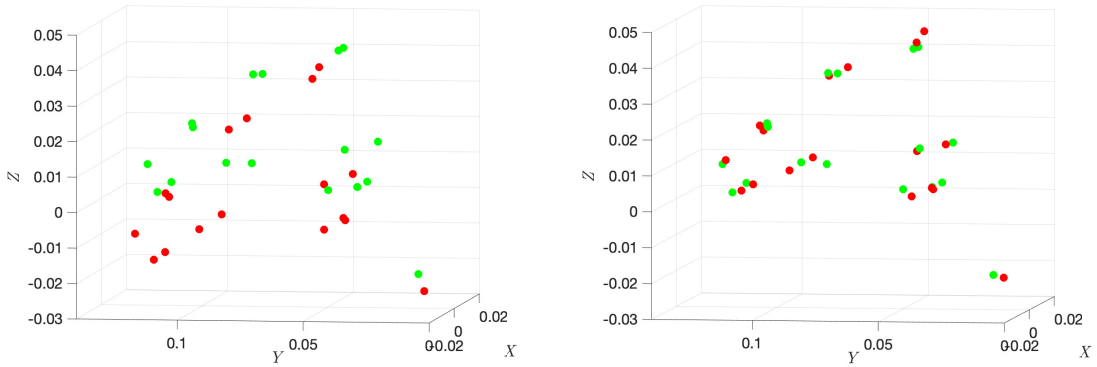
In order to "insert pathologies" and remove the paranasal sinuses with the help functional maps, the alignment of the patient's geometry with a reference one for which the procedure has already been applied is help full to reduce the probability of errors in the correspondence between geometries. The alignment is carried out in *Matlab* taking as reference the patient *P2*, patient with a particularly good CT-scan and to which all the pathologies have been carefully inserted by hand by surgeons that also removed the paranasal sinuses.



(a) Patient *P2*. **(b)** Patient *P1*.

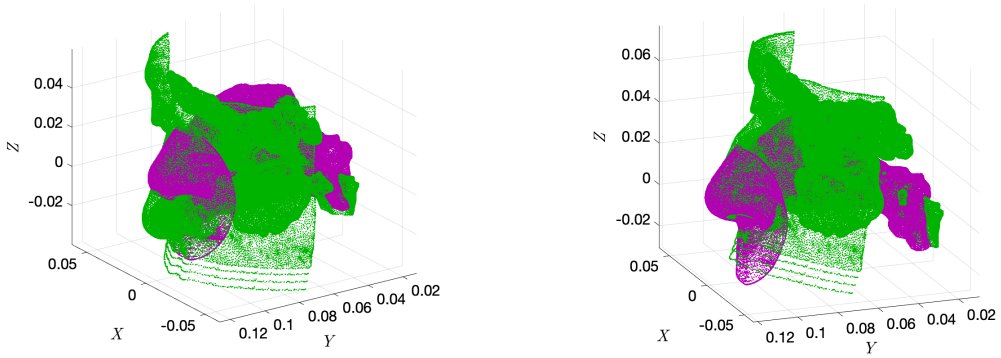
Figure 1.2: Some of the landmarks are visible in red.

A total of 17 Landmarks for the alignment have been selected as important and remarkable points for each geometry, such as the first vertex on the nose, the last on the throat, heads and tails of the turbinates, vertices referred to the nasal valve, to the olfactory region, to the septum and to the nostrils. Some of the landmarks are visible in Figure 1.2. The minimization of the sum of the distances between landmarks implemented in the *procrustes* function, returns a linear transformation (translation, reflection and orthogonal rotation) used to move the entire geometry. Figures 1.3 and 1.4 show the landmarks and the complete geometries before and after the alignment.



(a) Landmarks before alignment. **(b)** Landmarks after alignment.

Figure 1.3: The figure shows in green the landmarks of patient *P2* and in red patient *P1* ones.



(a) Geometries before alignment. **(b)** Geometries after alignment.

Figure 1.4: The figure shows in violet patient *P2*, in green patient *P1*.

2 | Shape registration and simplification

2.1. Removal of the paranasal sinuses through shape registration

Fluid dynamics simulations are in general very computational expensive whereby the simplest possible geometry is required in order to reduce the running time and the computational resources needed. Patient *P1* still has unnecessary parts for a CFD simulation that are the paranasal sinuses (visible in green in Figure 2.1), where the flow is so slow that can be neglected [9][10]. Since the whole procedure is thought to be applied to different patients, a tool able to automatically identify and remove the sinuses would be of great help, avoiding doing that by hand for each patient, that would be incredibly time consuming. This tool is the functional correspondence and comes from the computational geometry.

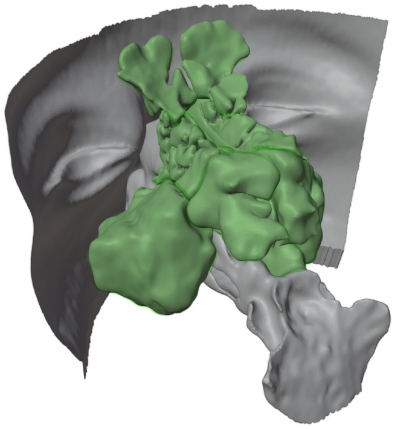


Figure 2.1: Paranasal sinuses.

The problem formulation is that of deformable object detection and dense correspondence in cluttered 3D scene presented in 2016 by Cosmo et al.[11]. The idea is to identify a linear operator (a functional map) between functional spaces defined over the shapes. Choosing appropriately the bases (such as the Laplace-Beltrami eigenfunctions proposed by Ovsjanikov et al. [12]), the operator admits a matrix representation that compactly encodes the map relating the two shapes. The shapes are modelled as 2D Riemannian manifolds $\mathcal{M}$ equipped with an intrinsic distance $d_{\mathcal{M}}$ and area element $d\mu$. In this framework, the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ admits eigen decomposition:

$$\Delta_{\mathcal{M}}\phi_i(x) = \lambda_i\phi_i(x) \qquad x \in int(\mathcal{M}), \tag{2.1}$$

$$\langle \nabla_{\mathcal{M}}\phi_i(x), \hat{n}(x)\rangle = 0 \qquad x \in \partial\mathcal{M}. \tag{2.2}$$

Equation 2.2 indicates the Neumann boundary conditions, where $\hat{n}(x)$ is the normal vector to the boundary, λ_i are the eigenvalues and ϕ_i the corresponding eigenfunctions. Since the eigenfunctions of the Laplacian form an orthonormal basis of $L^2(\mathcal{M}) = \{f : \mathcal{M} \to \mathbb{R} \mid \int_{\mathcal{M}} f^2 d\mu < \infty\}$, any function $f \in L^2(\mathcal{M})$ can be Fourier expanded

$$f(x) = \sum_{i\geq 1} \langle f, \phi_i\rangle_{\mathcal{M}}\phi_i(x), \tag{2.3}$$

where $\langle f, \phi_i(x)\rangle$ is the $L^2(\mathcal{M})$ inner product defined as $\langle f, g\rangle = \int_{\mathcal{M}} fg d\mu$.

So the idea is to identify a correspondence between shapes encoded in a linear operator $T : L^2(\mathcal{M}) \to L^2(\mathcal{N})$ that maps functions from $\mathcal{M}$ to $\mathcal{N}$. Furthermore, T holds matrix representation $\mathbf{C} = (c_{ij})$ such that

$$Tf(x) = T\sum_{i\geq 1} \langle f, \phi_i\rangle_{\mathcal{M}}\phi_i(x) = \sum_{i\geq 1} \langle f, \phi_i\rangle_{\mathcal{M}} T\phi_i(x) = \sum_{i,j\geq 1} \langle f, \phi_i\rangle_{\mathcal{M}} \underbrace{\langle T\phi_i, \psi_j\rangle_{\mathcal{N}}}_{c_{ji}} \psi_j, \tag{2.4}$$

where $\{\phi_i\}_{i>1}$ and $\{\psi_i\}_{i>1}$ are orthogonal bases on $L^2\mathcal{M}$ and $L^2\mathcal{N}$, and $f \in L^2(\mathcal{M})$ is an arbitrary function. The Fourier series is then truncated to the first k coefficients, leading to a $k \times k$ matrix $\mathbf{C}$. One can expect a near-isometry to have a matrix $\mathbf{C}$ close to the identity matrix, while Rodolà et al.[13] showed that for partial matching, matrix $\mathbf{C}$ presents a slanted-diagonal structure (Figure 2.2).

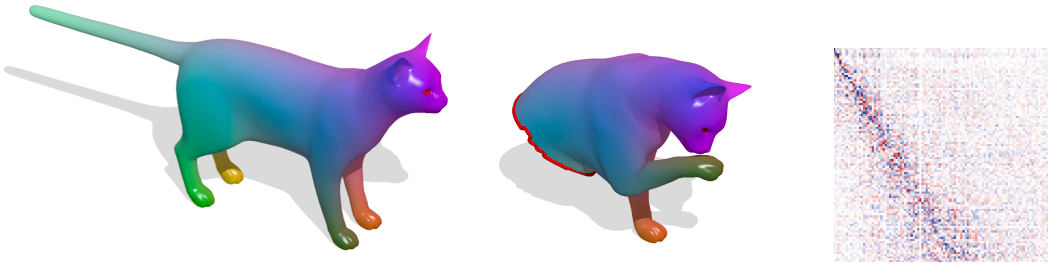


Figure 2.2: Slanted diagonal structure of the functional map **C**(last figure on the right) from $\mathcal{M}$(cat on the left) to $\mathcal{N}$(cat on the right) [13].

On the other hand, in case of partial correspondence in presence of clutter, only a sparse subset of eigenfunctions on the scene $\mathcal{N}$ have an approximately corresponding eigenfunction on the model $\mathcal{M}$. In the context of interest, the aim is firstly to identify the paranasal sinuses in a complete nose geometry, reflecting exactly the last of the aforementioned possibilities. In particular, the model here is represented by patient *P2*, the scene by patient *P1* and contains additional "clutter" that is the paranasal sinuses. Hence, the goal is to determine the part of patient *P1*'s geometry that is approximately isometric to patient *P2*'s one. The output of the method consists of the approximately isometric parts $\mathcal{M}' \subseteq \mathcal{M}, \mathcal{N}' \subseteq \mathcal{N}$ and the functional map T. Each part is represented as binary indicator function on the respective shape called segmentation function or segmentation mask.

In such a problem, descriptors preservation along with landmarks and segment correspondences are implemented [14]. The landmarks are the aforementioned ones and the segments are the paths linking these landmarks, computed as shortest geodesic path through the Dijkstra's algorithm [15] both on the *P2* and on the *P1* shapes (Figure 2.3).

Figure 2.3: Geodesic paths link the landmarks on the *P2* shape on the left hand side and on the *P1* shape on the right hand side.

Together with the 17 landmarks already cited, additional 50 descriptor points have been

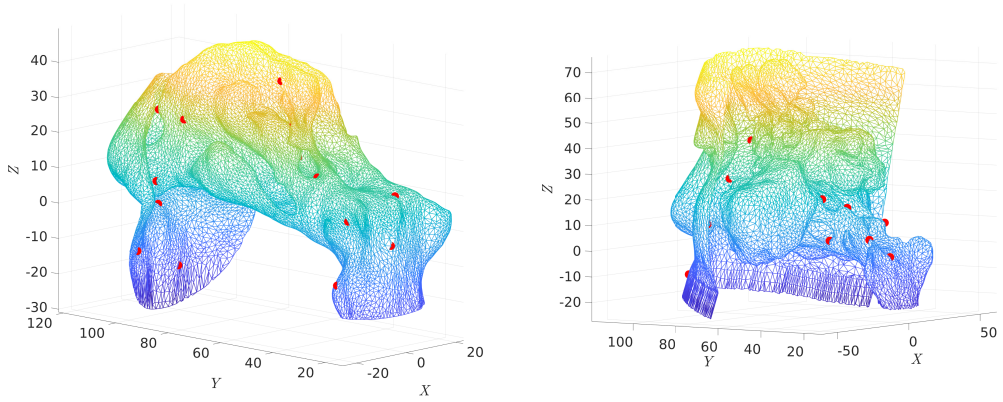


Figure 2.4: Additional descriptor points as K-means clustering centroids. On the left, patient *P2*; on the right patient *P1*.

added (Figure 2.4), considering the K-means clustering partitions on the *P2* shape, minimizing the sum over all clusters of the within-cluster sums of point-to-cluster-centroid Euclidean distances, and looking in shape *P1* for the nearest points to the *P2* ones.

The following unconstrained problem is therefore considered:

$$\min_{\mathbf{C},\theta,u,v} \| \mathbf{C}\mathbf{A}(\eta(u)) - \mathbf{B}(\eta(v)) \|_{2,1} + \| \mathbf{C}\Phi^T\eta(u) - \Psi^T\eta(v) \|_2^2 + \rho_{corr}(\mathbf{C},\theta) + \rho_{part}(u,v), \tag{2.5}$$

where $\Phi \in \mathbb{R}^{|\mathcal{M}|\times k}$ and $\Psi \in \mathbb{R}^{|\mathcal{S}|\times k}$ are two matrices containing as columns the first k Laplacian eigenfunctions of $\mathcal{M}$ and $\mathcal{S}$ respectively (computed as shown in Section 3.1); $\mathbf{A}(\eta(u))$ and $\mathbf{B}(\eta(v))$ contain the spectral coefficients of $\mathbf{F} \in \mathbb{R}^{|\mathcal{M}|\times d}$ and $\mathbf{G} \in \mathbb{R}^{|\mathcal{S}|\times d}$ (matrices holding the descriptors fields on model and scene), masked by the respective segmentations $u : \mathcal{M} \to [0,1]$ and $v : \mathcal{S} \to [0,1]$; $\eta(t) = \frac{1}{2}(tanh(2t-1)+1)$ is a saturation function that keeps u and v between 0 and 1.

ρ_{corr} is a regularizer for $\mathbf{C}$, explicitly:

$$\rho_{corr}(\mathbf{C},\theta) = \mu_1 \sum_{i\neq j}(\mathbf{C}^T\mathbf{C})_{ij}^2 + \mu_2 \sum_i |\mathbf{C}^T\mathbf{C}|_{ii} + \mu_3 \| \mathbf{C} \circ \mathbf{W}(\theta) \|_F^2, \tag{2.6}$$

where, $\| \, . \, \|_F$ is the Frobenius or Euclidean norm, defined as:

$$\| \mathcal{A} \|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{Tr(\mathcal{A}\mathcal{A}^H)}, \tag{2.7}$$

being $\mathcal{A}^H$ the conjugate transpose of $\mathcal{A}$.

The μ_1- and μ_2-terms require $\mathbf{C}^T\mathbf{C}$ to be as diagonal as possible and promote area preservation. The μ_3- terms requires $\mathbf{C}$ to have a slanted diagonal form with angle θ.

ρ_{part} is a regularizer for u and v, such that:

$$\begin{aligned} \rho_{part}(u,v) =& \mu_4 \left(\int_{\mathcal{M}} \| \nabla_{\mathcal{M}} \eta(u) \| \, dx + \int_{\mathcal{S}} \| \nabla_{\mathcal{S}} \eta(v) \| \, dx \right) \\ &+ \mu_5 \left(\int_{\mathcal{M}} \eta(u) dx - \int_{\mathcal{S}} \eta(v) dx \right)^2 \\ &- \mu_6 \left(\int_{\mathcal{M}} \eta(u) dx + \int_{\mathcal{S}} \eta(v) dx \right), \end{aligned} \tag{2.8}$$

where the μ_4-term derives from the Mumford-Shah functional [16] and promote the production of contiguous regions, penalizing boundary length; the μ_5-term requires the same area for $\mathcal{M}$ and $\mathcal{S}$ and the μ_6-term control the size of the parts.

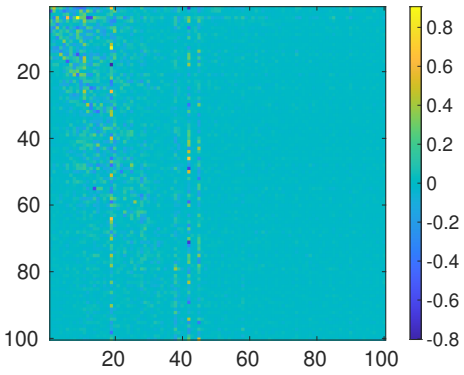
The values of the parameters used for this work are summarized in Table 2.1.

Values of the parameters	
Parameter	*value*
μ_1	5
μ_2	1
μ_3	0.1
μ_4	1
μ_5	2
μ_6	40

Table 2.1: Value of the parameters in Equation (2.5).

2.2. Results: match object in clutter

The result given by the functional correspondence is visible in Figure 2.5 and Figure 2.6, where vertices whose mask value is 0 are depicted in black and vertices whose mask value is 1 are colored creating a scalar map both on the *P2* shape and the *P1* one. While the functional map obtained can be seen in the figure on the right as scaled colored image: note the slightly visible slanted diagonal structure of the map.



The segmentation mask is very accurate and reduces the number of vertices from $\approx 25K$ to $\approx 12K$, leaving only small parts of the aimed shape, probably associated to parts of the surface far from both the geodesic path and the landmarks. Hence, some "holes" in the geometry are present and need to be corrected. A filter is applied checking for every vertex the "neighbour" vertices up to a given distance and changing the mask value to 1 if necessary.

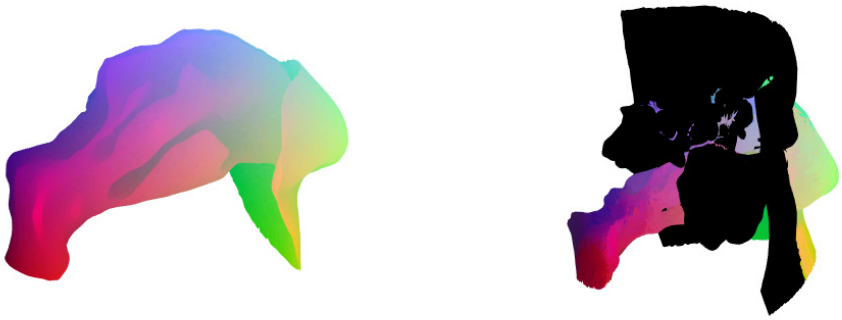


Figure 2.5: Result of the matching.

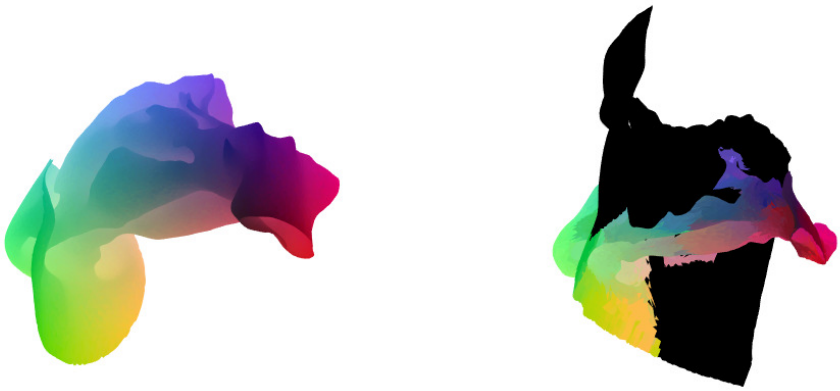


Figure 2.6: Result of the matching.

Once the final shape is obtained, an up-scaling to a finer mesh is required. Exploiting the FLANN (Fast Library for Approximate Nearest Neighbour, `https://github.com/flann-lib/flann`), the nearest neighbour of each vertex of a mesh of $\approx 50K$ vertices is retrieved in the $25K$ mesh, and is given the same value of the mask. In this way, the registered shapes will have again $\approx 25K$ vertices, exploiting all the available RAM.

Before exporting the very final geometry, a good idea could be to keep the whole face of the patient, useful to set boundary conditions in the CFD simulations. Whereby, the mask is converted in RGB code for black ([0 0 0]) and white ([255 255 255]) and exported as *.ply* file (Figure 2.7).

Blender is used to "paint" the face in white and the *.ply* is imported again in *Matlab*, where the RGB code is converted back to [0 1] values. Now the final geometry can be obtained, keeping only the vertices associated to a mask value of 1, modifying properly the triangulation matrix. The final result can be seen in Figure 2.8.

Figure 2.7: Blender view of the segmentation mask in black and white.

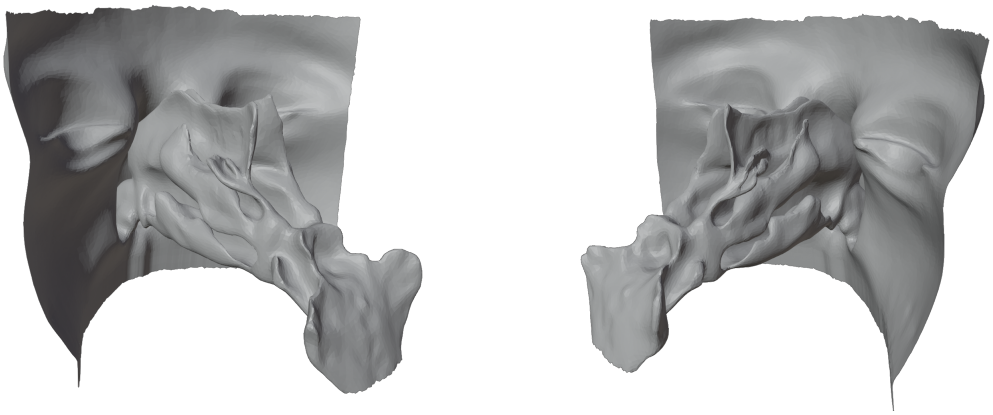


Figure 2.8: Final geometry obtained with the segmentation.

3 | Shape matching: ZoomOut method

3.1. ZoomOut

The geometry shown in Figure 2.8 is a healthy one, meaning that has been diagnosed to exhibit no pathologies. The goal is now to find an automatic way to precisely map a given pathology inserted by hand in the *P2* geometry onto the *P1* shape, recalling that a pathology can be seen as a geometry deformation of a healthy shape. The map computed in Section 2.1 was in fact good for the matching of a shape into a cluttered one, well recognizing the model scene inside the target one, but locally was not very accurate [11].

The method exploited in this section to compute a much higher resolution map with respect to the Section 2.1's one is the one presented by Melzi et al.[17] in 2019, which is based on spectral iterative up-sampling technique for non-rigid shape correspondence called ZoomOut method.

Given a pair of shapes $\mathcal{M}$ and $\mathcal{N}$ represented as triangular meshes, one can associate the positive semi-definite Laplacian matrices $\mathcal{L}_{\mathcal{M}}$ and $\mathcal{L}_{\mathcal{N}}$, discretized via the standard cotangent weight scheme [13] recalled here below.

Let $\mathcal{M}$ a manifold sampled in n points $x_1, ..., x_n$ which are connected by interior and boundary edges $E = E_i \cup E_b$ and faces F, forming a manifold triangular mesh, the discretization of the Lapalcian takes the form $n \times n$ sparse matrix

$$\mathcal{L}_{\mathcal{M}} = \mathcal{S}_{\mathcal{M}}^{-1} \mathcal{W}_{\mathcal{M}}, \tag{3.1}$$

where

$$w_{ij} = \begin{cases} -\dfrac{(cot\alpha_{ij} + cot\beta_{ij})}{2} & ij \in E_i; \\ -\dfrac{cot\alpha_{ij}}{2} & ij \in E_b; \\ \sum\limits_{k \neq i} w_{ik} & i = j; \\ 0 & else, \end{cases} \tag{3.2}$$

is the cotangent weight matrix, $\mathcal{S}_{\mathcal{M}} = diag(s_i, ...s_n)$, $s_i = \frac{1}{3}\sum_{jk:ijk \in F} s_{ijk}$ is the local area element at vertex i, s_{ijk} denotes the area of the triangle ijk, and α_{ij},β_{ij} denote the angles $\angle ikj$, $\angle jhi$ of the triangles sharing the edge ij (see Figure 3.1).

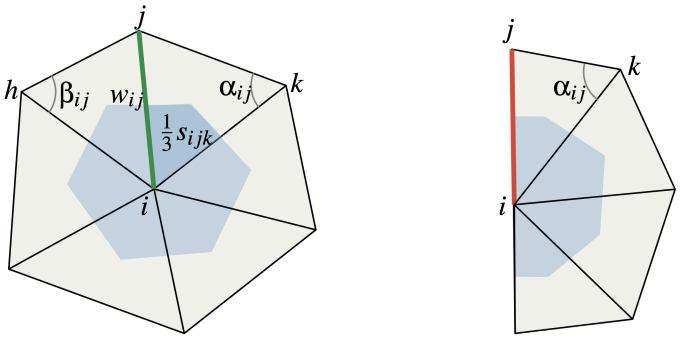


Figure 3.1: Discretization of the Laplace-Beltrami operator on a triangular mesh for interior edges (green, left) and boundary edges (red, right).

Figure 3.2 shows the first 4 Laplace-Beltrami eigenfunctions on the *P1* and on the *P2* shapes.

The ZoomOut method takes as input a $k_{\mathcal{M}} \times k_{\mathcal{N}}$ functional map $\mathcal{C}_0$ (see Section 2.1) or a point to point correspondence $T : \mathcal{M} \to \mathcal{N}$ and extends it to a new map $\mathcal{C}_i$ of size $(k_{\mathcal{M}} + i) \times (k_{\mathcal{N}} + i)$ by the following two-steps procedure applied $n-$times:

For i from 0 to n do

(1) Compute a point-to-point map T via Equation 3.4 and encodes it as matrix Π such that:

$$\mathcal{C} = (\Phi_{\mathcal{M}}^{k_{\mathcal{M}}})^{+} \Pi \Phi_{\mathcal{N}}^{k_{\mathcal{N}}}, \tag{3.3}$$

where $\Phi_{\mathcal{M}}^{k_{\mathcal{M}}} = [\phi_1, \phi_2, ..., \phi_{K_{\mathcal{M}}}]$ and $\Phi_{\mathcal{N}}^{k_{\mathcal{N}}} = [\phi_1, \phi_2, ..., \phi_{K_{\mathcal{N}}}]$ are the matrices having the eigenfunctions of $\mathcal{M}$ and $\mathcal{N}$ as columns, and $^{+}$ denotes the Moore-Penrose pseudo-inverse [18].

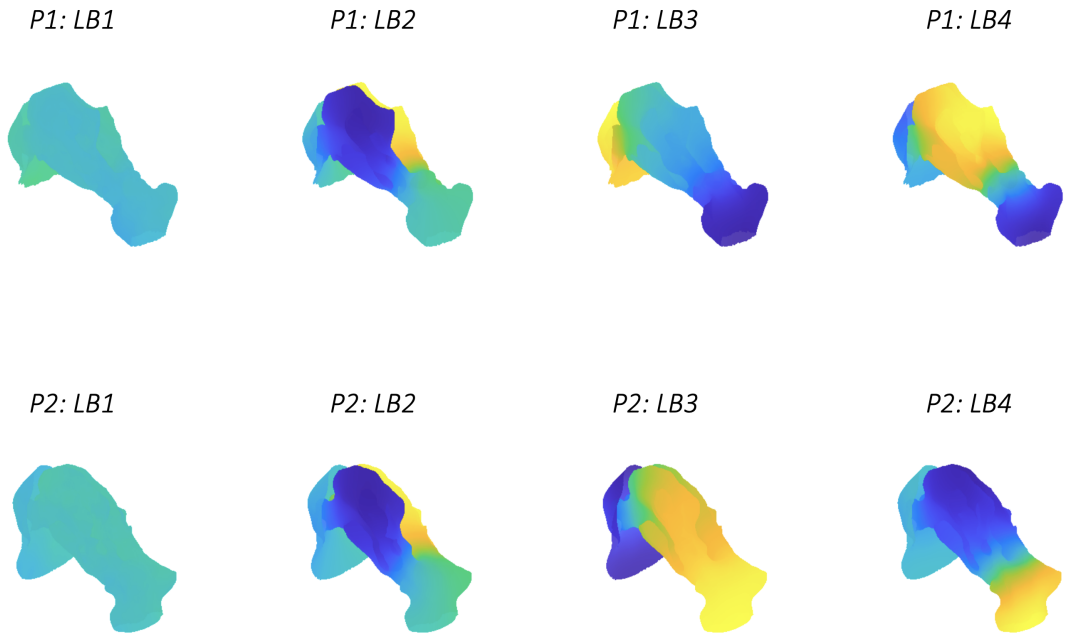


Figure 3.2: first 4 Laplace-Beltrami eigenfunctions computed with the cotangent method. Top row: *P1* shape. Bottom row: *P2* shape.

(2) Set $\mathcal{C}_1 = (\Phi_{\mathcal{M}}^{k_{\mathcal{M}}+1})^{+}\mathcal{S}_{\mathcal{M}}\Pi\Phi_{\mathcal{N}}^{k_{\mathcal{N}}+1}$.

To compute a pointwise map T from a given $\mathcal{C}$ in step (1) the following problem is solved:

$$T(p) = \arg\min_{q} \| \mathcal{C}(\Phi_{\mathcal{N}}(q))^T - \Phi_{\mathcal{M}}(p))^T \|_2, \quad \forall p \in \mathcal{M}, \tag{3.4}$$

where $\Phi_{\mathcal{M}}(p)$ denotes the p^{th} of the matrix of eigenvectors $\Phi_{\mathcal{M}}$. This procedure return a point-to-point map $T : \mathcal{M} \to \mathcal{N}$ and is implemented using the FLANN library already cited in Section 2.2.

The term "up-sampling" in the description of this method is related to the fact that at every iteration ZoomOut introduces additional frequencies and thus intuitively adds samples in the spectral domain for representing a map.

The pipeline is initialized with a 2×2 identity functional map $\mathcal{C}_0$, as one can notice in Figure 3.2 that the first two Laplace-Beltrami eigenfunctions align with each other. The map is then refined up to a 120×120 with a step size of 1.

3.2. Results: ZoomOut

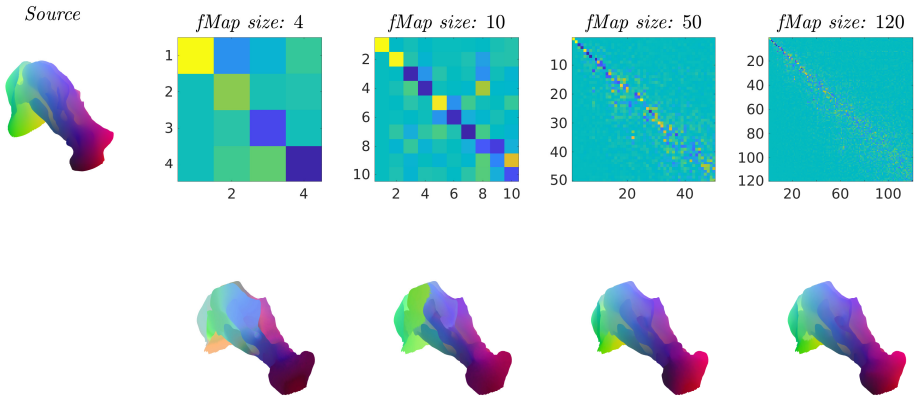


Figure 3.3: Some of the ZoomOut method results, starting from a 4×4 functional map, to a 120×120 one. In the top row: matrix representations of the map $\mathcal{C}$. In the bottom row: the mapping of the scalar map created on *source* though the correspondent functional map in the first row.

The results of the ZoomOut method are visible in Figure 3.3, Figure 3.4, Figure 3.5 and Figure 3.6.

One can notice from Figure 3.3 that the correspondence between the 2 shapes is quite accurate starting from a functional map size of 50 eigenfunctions; moreover, the functional map matrix representation becomes almost diagonal, reflecting the fact that, despite the two shapes are actually different, they look alike and the structure of the functional map is similar to a near-isometric one. Figure 3.4 and Figure 3.5 show the results in terms of scalar map on the surface: a scalar map is created on the reference shape of *P2*, and then mapped using the point-to-point form of the map $\mathcal{C}$ to the *P1* shape. The correspondence is very precise and a possible way to evaluate how good is the mapping is shown in Figure 3.5: a high frequency sinusoidal is plotted in the $y-$direction, creating some visible bands that are transported on the *P1* shape by the mapping. Therefore, the accuracy of the correspondence can be evaluated by looking at how these bands get deformed on the *P1* shape with respect to the *P2* ones. The bands keep very regular while the slight lack of definition at the borders may be due to the small number of vertices of the *P1* shape with respect to the *P2* one.

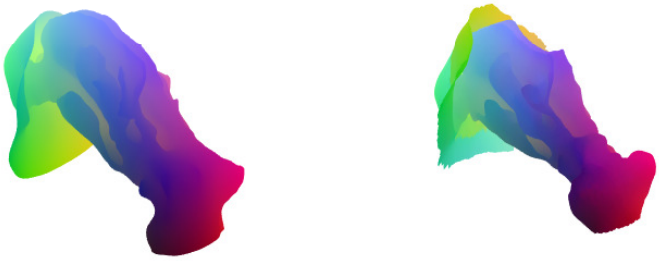


Figure 3.4: The scalar map defined on the *P2* shape (left) is mapped on the *P1* shape (right) through the final 120×120 map.

Figure 3.5: The high frequency sinusoidal plotted in the $y-$direction shows how good is the correspondence between the two shapes.

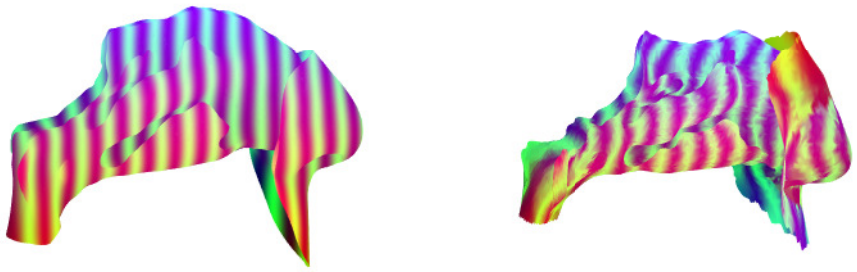


Figure 3.6: The high frequency sinusoidal plotted in the $y-$direction shows how good is the correspondence between the two shapes.

3.3. Pahologies insertion

Taking advantage of the correspondence computed in Section 3.1 through the ZoomOut method a function defined on the *P2* reference shape can be mapped on *P1* preserving its well definition. Hence, the idea now is to use the "diseased" geometries of the patient *P2* for which the pathologies has been inserted by hand by ENT doctors, to map pathologies on the *P1* shape, aiming to extend the available "labeled" set of geometries, i.e. the diseased patients for which the pathologies are known. This is a fundamental step for the training of a Machine Learning model, as it will be described in Chapter 5.

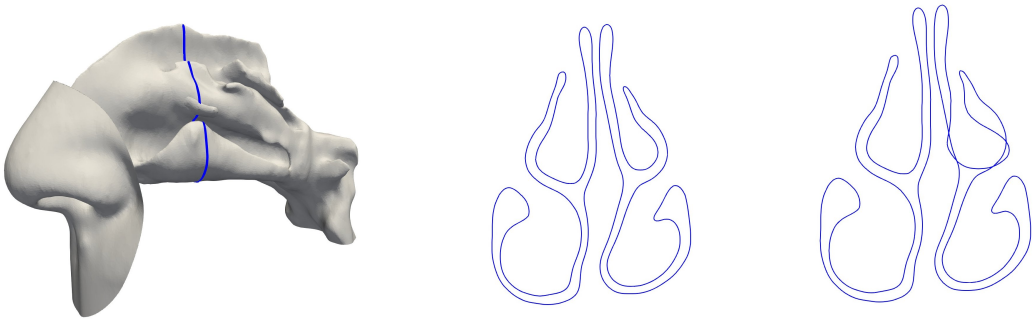


Figure 3.7: Left: section of the *P2* shape used to visualize the pathology. Center: P*1* "healthy" shape. Right: *P2* "diseased" shape.

The procedure is the following: the *P2* "healthy" shape together with the "diseased" geometries are considered and their "shape difference" is computed, being the difference between the coordinates of all the "diseased" vertices and the "healthy" ones; a vectorial difference function is thus obtained. Each of the three scalar maps for $\Delta x, \Delta y$ and Δz, can be now mapped on the *P1* shape using the point-to-point form T of the 120×120 functional map $\mathcal{C}$ assessed through the ZoomOut method in the previous section. The point-to-point map is retrieved solving Problem 3.4. Knowing the point-to-point correspondence between the two shapes, one can associate to each point of the *P1* shape the deformation of the corresponding point in the *P2* "diseased" shape. Finally, the mapped Δ-functions are added to the *P1* vertices coordinates, leading to the *P1* "diseased" geometries. An example can be seen in Figure 3.8 and Figure 3.9 with the hypertrophy of the head of the middle turbinate shown in Figure 3.7.

The procedure is repeated for different shapes of septal deviations and hypertrophies, occurring in different locations and having different levels of severity, for a total of 17 different cases shown in Appendix A.

Figure 3.8: The hypertrophy is visible on the *P2* shape as deformation function on the surface.

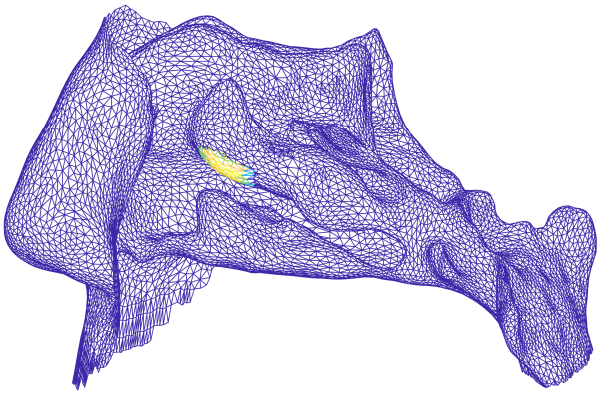


Figure 3.9: The mapped hypertrophy is visible on the *P1* shape as mapped deformation function on the surface.

4 | LES simulations

Accurate information on the internal flow in the nasal cavity is now required, whereby Computational Fluid Dynamics (CFD) plays a central role in this section. First of all, the choice of the CFD model is fundamental. Given the complexity of the geometry, a Direct Numerical Simulation (DNS) would be hugely expensive and practically infeasible. On the other hand, a RANS simulation, despite the immensely lower computational cost, would return not sufficiently accurate results. For these reasons a LES model has been chosen and performed on each geometry, being LES simulations a good compromise between RANS and DNS.

4.1. Background

The Large Eddy Simulation approach [19] relies on the concept of large and small scales, and in practice, the LES technique consists in solving the set of ad-hoc governing equations on a computational grid which is too coarse to represent the smallest physical scales, that are instead modelled as follows.

Starting from the incompressible unsteady Navier-Stokes equations,

$$\begin{cases} \dfrac{\partial u_j}{\partial x_j} = 0 \\ \dfrac{\partial u_j}{\partial t} + \dfrac{\partial u_i u_j}{\partial x_i} = -\dfrac{\partial P}{\partial x_j} + \nu \dfrac{\partial^2 u_j}{\partial x_i \partial x_i} \end{cases} \tag{4.1}$$

a decomposition is applied such that

$$\mathbf{u}(\mathbf{x}, t) = \tilde{\mathbf{u}}(\mathbf{x}, t) + \mathbf{u}'(\mathbf{x}, t) \tag{4.2}$$

where $\tilde{\cdot}$ is a low-pass filter operator or bandwidth Δ of cutoff wavenumber $\kappa_c = \frac{\pi}{\Delta}$, with

$$\tilde{\mathbf{u}}(\mathbf{x}, t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \mathcal{G}(\mathbf{r}, \mathbf{x}) \mathbf{u}(\mathbf{x} - \mathbf{r}, t) d\mathbf{r}, \tag{4.3}$$

being $\mathcal{G}(\mathbf{r}, \mathbf{x})$ the filter kernel with $\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\mathcal{G}(\mathbf{r}, \mathbf{x})d\mathbf{r} = 1$, usually a box, gaussian or sharp spectral filter.

Applying the filter in Equation 4.3 to Equations 4.1 one can retrieve the filtered Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial \tilde{u}_j}{\partial x_j} = 0 \\ \dfrac{\partial \tilde{u}_j}{\partial t} + \dfrac{\partial \tilde{u}_i \tilde{u}_j}{\partial x_i} = -\dfrac{\partial \tilde{P}}{\partial x_j} + \nu \dfrac{\partial^2 \tilde{u}_j}{\partial x_i \partial x_i} - \dfrac{\partial \tau_{ij}}{\partial x_i}, \end{cases} \tag{4.4}$$

where the effect of the small scales upon the resolved scales of the turbulence appears in the sub-grid scales stress tensor $\tau_{ij} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j$, which needs to be modelled to close the equations (see Section 4.2).

4.2. Turbulence Model

The turbulence approach proposed in this project is the WALE one [20][21][22]. The sub-grid scales stress tensor is modelled by eddy-viscosity approach, where the turbulence effect expressed by the latter is embedded in the eddy-viscosity ν_t, such that

$$\nu = \nu_0 + \nu_t. \tag{4.5}$$

The WALE model (Wall-Adapting Local Eddy-viscosity) provides a particular expression for ν_t as

$$\nu_t = (\mathcal{C}_W \Delta\mathcal{X})^2 \overline{\mathcal{OP}}, \tag{4.6}$$

being $\mathcal{C}_W$ the WALE model constant, $\Delta\mathcal{X}$ the lattice spacing and $\overline{\mathcal{OP}}$ the WALE model operator defined as

$$\overline{\mathcal{OP}} = \frac{(\mathcal{S}_{ij}^d \mathcal{S}_{ij}^d)^{\frac{3}{2}}}{(\tilde{\mathcal{S}}_{ij}\tilde{\mathcal{S}}_{ij})^{\frac{5}{2}} + (\mathcal{S}_{ij}^d \mathcal{S}_{ij}^d)^{\frac{5}{4}}}, \tag{4.7}$$

where

$$\begin{aligned}
\mathcal{S}_{ij}^{d} &= \tilde{\mathcal{S}}_{ik}\tilde{\mathcal{S}}_{kj} + \tilde{\Omega}_{ik}\tilde{\Omega}_{kj} - \frac{1}{3}\delta_{ij}\left(\tilde{\mathcal{S}}_{mn}\tilde{\mathcal{S}}_{mn} - \tilde{\Omega}_{mn}\tilde{\Omega}_{mn}\right) \\
\tilde{\Omega}_{ij} &= \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} - \frac{\partial \tilde{u}_j}{\partial x_i}\right) \\
\tilde{\mathcal{S}}_{ij} &= \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right)
\end{aligned} \tag{4.8}$$

The WALE approach is based on tensor invariant, meaning that it is not affected by any coordinate translation or rotation, and reproduces the proper scaling at the wall ($\nu_t = \mathcal{O}(y^3)$). It is well-suited for LES in complex geometries as only local information is required to build the eddy-viscosity. Finally, it is sensitive to both the strain and the rotation rate of the small turbulent structures, improving the performance of the Smagorinsky model [23], which takes in account only for the strain rate and leaves a ν_t of $\mathcal{O}(1)$ at the wall.

4.3. OpenFOAM setup

The simulations have been performed in OpenFOAM on a 0.65 seconds inspiration [24]. The starting geometries are the *P1* STL files (with and without pathologies) and the STL of a cut sphere positioned to create a "closed mask" around the nostrils (see Figure 4.1), necessary to enforce the inlet boundary conditions.

The patches of the problem are 3:

- **clownose**: it simulates the external air. The flow rate is imposed here.
- **testa**: patch representing the walls of the problem, hence the nasal cavities.
- **throat**: outlet patch, situated at the height of laryngopharynx.

The patches can be seen in Figure 4.2.

4.3.1. The Mesh

The mesh generation requires different steps to be completed. The two main phases are the *blockMesh* backgound mesh generation and the refinement of the latter by the *snappyHexMesh* utility.

blockMesh The *blockMesh* utility creates a background hexahedral mesh specifying the number of cells in all directions for each block generated. Each block of the geometry is defined by 8 vertices, one at each corner of a hexahedron. A glimpse of the *blockMeshDict*

Figure 4.1: Position of the sphere on the patient *P1*.

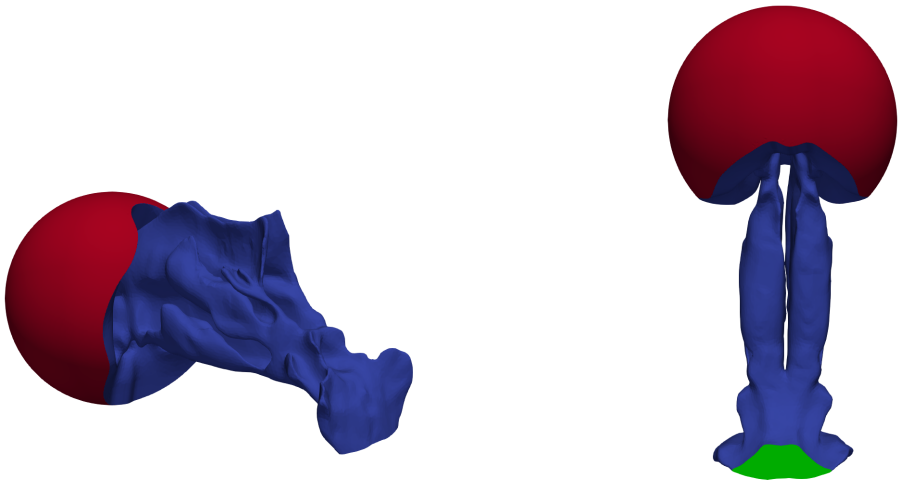


Figure 4.2: Patches of the problem. In red the "clownose", in blue the "testa" patch, in green the "throat".

```
x1  -0.1;
x2   0.1;

y1 -0.03;
y2  0.03;
y3  0.27;

z1 -0.0799;//-0.08;
z2 -0.0199;//-0.02;
z3  0.1401;// 0.14;


vertices
(
    ($x1 $y2 $z1)
    ($x2 $y2 $z1)
    ($x2 $y3 $z1)
    ($x1 $y3 $z1)
    //
    ($x1 $y2 $z2)
    ($x2 $y2 $z2)
    ($x2 $y3 $z2)
    ($x1 $y3 $z2)

    ($x1 $y1 $z2)
    ($x2 $y1 $z2)
    //

    ($x1 $y1 $z3)
    ($x2 $y1 $z3)
    ($x2 $y3 $z3)
    ($x1 $y3 $z3)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (300 360 90) simpleGrading (1 1 1)
    hex (8 9 6 7 10 11 12 13) (300 450 240) simpleGrading (1 1 1)
);
```

Figure 4.3: Section of the blockMeshDict file in the *system*−folder.

file can be seen in figure 4.3. The two main blocks containing the background mesh can be seen in Figure 4.4.

snappyHexMesh The *snappyHexMesh* utility generates 3-dimensional meshes containing hexahedra that approximately conforms to the surface by iteratively refining. It consists of different sub-phases: the *Castelleted* phase, the *Snappy* phase and the mesh quality controls phase. To run *snappyHexMesh* the background mesh is required from the *blockMesh* utility as well as the *snappyHexMeshDict* dictionary located in the *system* directory.

- ***Castellated***: splits the background cells in smaller cells. The control parameters are:
  - *maxLocalCells*: maximum number of cells per processor.
  - *maxGlobalCells*: maximum number of cells.

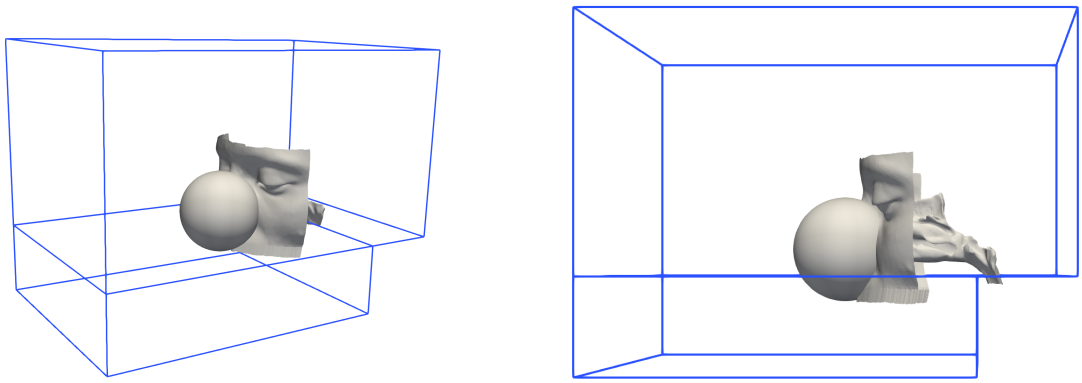


Figure 4.4: Main blocks exploited in the *blockMesh* utility. Note that the final part of the laryngopharynx must be outside of the block to ensure that the mesh will be created inside the nasal cavities.

- *minRefinementCells*: minimum number of cells to be refined.
- *nCellsBetweenLevels*: number of buffer cells between different refinement levels.
- *refinementSurfaces*: how to refine surfaces.
- *resolveFeatureAngle*: intersection angle between cells.
- *locationInMesh*: specify a point inside the mesh.

The values used for these parameters are shown in figure 4.5.

- ***Snappy***: brings the vertices of the exahedra on the surface and iteratively reduces the displacement of the verticed that do not meet the mesh quality parameters required. The control parameters are:
  - *nSmoothPatch*: number of patch smoothing iterations before finding correspondence to surface.
  - *tolerance*: ratio of distance for points to be attracted by surface feature point or edge, to local maximum edge length.
  - *nSolveIter*: number of mesh displacement relaxation iterations.
  - *nRelaxIter*: maximum number of snapping relaxation iterations.

  The values used for these parameters can be seen in Figure 4.6.
- ***meshQualityControls***: subdictionary that controls the quality of the mesh. The entries are:

```
castellatedMeshControls
{
        features ();
        maxLocalCells 90000000;
        maxGlobalCells 90000000;
        minRefinementCells 1;
        nCellsBetweenLevels 2;
        refinementSurfaces
        {
                testa
                {
                        level (3 3);//(3 3)
                        gapLevelIncrement 2;

                }

                clownose
                {
                        level (1 1);
                }

        }

        resolveFeatureAngle 80;

planarAngle 30;

        locationInMesh (0 0.13 0);
        allowFreeStandingZoneFaces false;

}
```

Figure 4.5: Section of the snappyHexMeshDict containing the *Castelleted* parameteres.

```
snapControls//

        nSmoothPatch 2;

        tolerance 2.0;

        nSolveIter 50;

        nRelaxIter 10;
}
```

Figure 4.6: Section of the snappyHexMeshDict containing the *Snappy* parameteres.

```
meshQualityControls
{
        maxNonOrtho 60;

        maxBoundarySkewness 6;

        maxInternalSkewness 4;

        maxConcave 80;

        minVol -1e-13;

        minTetQuality 1e-15;

        minArea 1e-13;

        minTwist 0.05;

        minDeterminant 0.001;

        minFaceWeight 0.05;

        minVolRatio 0.01;

        minTriangleTwist -1;

        nSmoothScale 4;

        errorReduction 0.75;
}
```

Figure 4.7: Section of the snappyHexMeshDict containing the *meshQualityControls* parameteres.

- *maxNonOrtho*: maximum non-orthogonality allowed.
- *maxConcave*: max concaveness allowed.
- *minTetQuality*: minimum quality of the tetrahedron formed by the face-centre and variable base point minimum decomposition triangles and the cell centre.
- *minArea*: minimum face area.
- *minDeterminant*: minimum normalised cell determinant.

The values used for these parameters can be seen in Figure 4.7.

The resulting mesh for *P1* "healthy" geometry can be seen in Figure 4.8 and Figure 4.9. Some of the mesh characteristic values are reported in Table 4.1.

4.3.2. Simulation controls

The *fvSchemes* dictionary in the *system*−directory sets the numerical schemes for terms, such as derivatives in equations, that appear in applications being run. It contains different sub-dictionaries for each different term and in particular:

- *ddtSchemes*: first time derivative numerical scheme, for this setup the *backward*

Mesh characteristic values	
Parameter	*value*
Points	16788901
Faces	42114279
Cells	12954489
Min determinant cells	108

Table 4.1: Characteristic values for the "healthy" *P1* mesh. The Min determinant cells indicates the number of cell with determinant lower than the *minDeterminant* value of the *meshQualityControls* sub-dictionary.

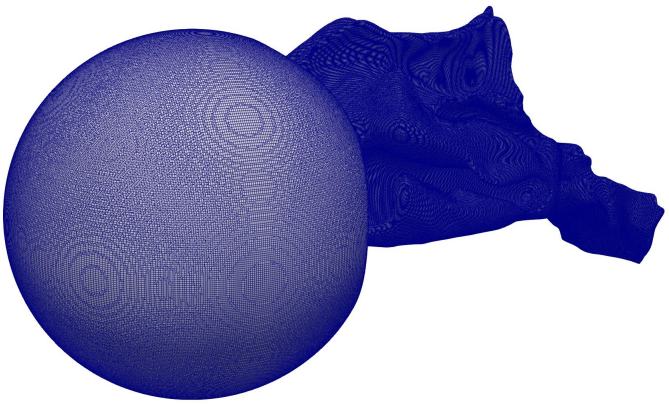


Figure 4.8: Resulting mesh for the *P1* "healthy" geometry. On the right a coronal section.

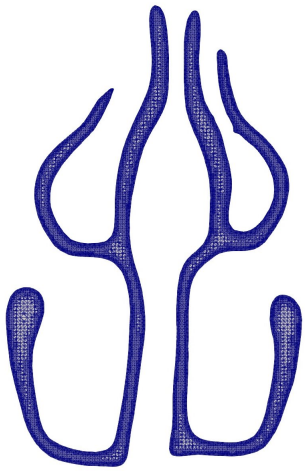


Figure 4.9: Resulting mesh for the *P1* 'healthy' geometry. On the right a coronal section.

second order, implicit scheme has been used.

- *gradSchemes*: numerical scheme for the gradient. In particular here the second order *Gauss* scheme has been used.
- *divSchemes*: numerical scheme for the divergence. Here the *Gauss linear* second order scheme has been used.
- *laplacianSchemes*: numerical scheme for the laplacian. Here the second order *Gauss linear corrected* has been applied.
- *interpolationSchemes*: Point-to-point interpolations of values.
- *snGradSchemes*: component of gradient normal to a cell face.
- *fluxRequired*: sets the information about fields for which is required to calculate a flux.

The *fvSchemes* dictionary can be seen in Figure 4.10.

The *turbulenceProperties* dictionary in the *constant*-directory is read by any solver that includes turbulence modelling. Within that file is the *simulationType* keyword that controls the type of turbulence modelling, in this case LES simulations. Alongside the simulations type, the dictionary requires the specification of the turbulence model and the cutoff wavenumber Δ (see Section 4.1 and Section 4.2), respectively *WALE* turbulence model and *cubeRootVol*, for which

$$\Delta = c(V_c)^{\frac{1}{3}}, \tag{4.9}$$

```
ddtSchemes
{
    default         backward;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
    div(phi,R)      Gauss linear;
    div(R)          Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}
```

Figure 4.10: *fvSchemes* dictionary in the *system*-directory.

```
simulationType LES;

LES
{
    LESModel        WALE;

    turbulence      on;

    printCoeffs     on;

    delta           cubeRootVol;

    cubeRootVolCoeffs
    {
        deltaCoeff      1;
    }

```

Figure 4.11: *turbulenceProperties* dictionary in the *constant*-directory.

where V_c is the cell volume and c the *deltaCoeff*. The *turbulenceProperties* dictionary can be seen in Figure 4.11.

The *controlDict* dictionary in the *system*-directory is used to specify the main case controls as the timing information, write format, and optional libraries that can be loaded at run time. In particular here the OpenFOAM application is set, in this case *pimpleFoam*, transient solver for incompressible, turbulent flow of Newtonian fluids. A section of the *controlDict* dictionary can be seen in Figure 4.12.

```
application     pimpleFoam;

startFrom       latestTime;

startTime       0.0;

stopAt          endTime;

endTime         0.6502;

deltaT          0.00005;

writeControl    adjustableRunTime; //timeStep; //runTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     ascii;

writePrecision  10;

writeCompression on;

timeFormat      general;

timePrecision   10;

runTimeModifiable true;

maxCo           1.0;

adjustTimeStep  yes;
```

Figure 4.12: Section of the *controlDict* dictionary in the *system*-directory.

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    testa
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    throat
    {
        type            zeroGradient; //inletOutlet;

    }

    clownose
    {
        type                    flowRateInletVelocity;
        volumetricFlowRate      constant 0.0002666667;
        rhoInlet                1.225;
    }
 }

// ************************************************************************* //
```

Figure 4.13: Section of the velocity boundary conditions file in the OpenFOAM 0-folder.

4.3.3. Boundary Conditions

Initial and boundary conditions are needed for all the physical variables of the problem:

Velocity Non-slip and non-penetration boundary conditions are imposed on the head patch "testa", whereby the velocity is here set to zero. On the throat patch the *zeroGradient* condition has been applied wherease on the clownose patch the *flowRateInletVelocity* has been fixed to 16 l/min corresponding to $\approx 2.6 \cdot 10^{-4} m^3/s$ (in accordance with the work by Wexler et al. [25]). (See Figure 4.13).

Pressure As regards the pressure, a *zeroGradient* condtion is enforced on the head patch "testa" and on the sphere "clownose", as the boundary conditions that sets the flow into motion is the fixed flow rate. A reference value for the pressure is required, whereby a zero value for the total pressure is imposed on the throat patch. (See figure 4.14).

Eddy viscosity The eddy viscosity is set to zero on the wall patch "testa", while for the inlet and for the outlet is set the *zeroGradient* condition. (see Figure 4.15).

```
17 dimensions      [0 2 -2 0 0 0 0];
18
19
20 internalField   uniform 0;
21
22 boundaryField
23 {
24     testa
25     {
26         type            zeroGradient;
27     }
28
29     throat
30     {
31
32         type            totalPressure;
33         p0              uniform 0.0;
34         gamma           1.4;
35     }
36     clownose
37     {
38         type            zeroGradient;
39
40     }
41 }
42
43
44 // ************************************************************************* //
```

Figure 4.14: Section of the pressure boundary conditions file in the OpenFOAM 0-folder.

```
17 dimensions      [0 2 -1 0 0 0 0];
18
19
20 internalField   uniform 0;
21
22 boundaryField
23 {
24     testa
25     {
26         type            fixedValue;
27         value           uniform 0;
28     }
29
30
31     clownose
32     {
33         type            zeroGradient;
34     }
35     throat
36     {
37         type            zeroGradient;
38     }
39 }
40
41
42 // ************************************************************************* //
```

Figure 4.15: Section of the eddy viscosity boundary conditions file in the OpenFOAM 0-folder.

4.3.4. Running the simulations

The simulations have been performed in remote *ssh* access to the server of the HPC infrastructure GALILEO100 of the CINECA system in Bologna. As the the meshes presented a huge number of cells and the flow is turbulent, important computational resources were needed. For each simulation, 2 out of the 528 nodes have been used, having 48 cores each, for a total of 96 cores and 160 GB's of RAM. The *decomposeParDict* dictionary in the *system*-directory handles the domain decomposition for the parallel running, setting the number of subdomains to 96 (number of cores).

4.4. LES results

In this section the results of the *P1* "healthy" geometry are firstly qualitative described and compared to other works with the aim of validating the simulations. Then, as example the results of the LES simulation of the *P1* "healthy" shape and the "diseased" shape associated to the opened anterior septal deviation of level 1 (see Figure 4.21) are briefly compared.

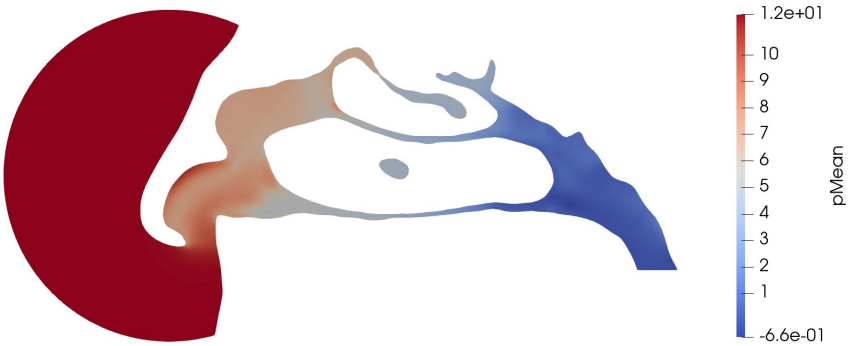


Figure 4.16: Mean pressure field on a sagittal view of the *P1* "healthy" geometry.

Figure 4.16 shows the mean pressure field in a sagittal section of the *P1* "healthy" geometry. One can notice that the pressure reduces as the flow reaches the throat with a mean pressure difference between the inlet and the throat of about $13Pa$. This behaviour is due to the progressive section restriction caused by the presence of the turbinates. The pressure drops up to a negative value as the reference *zero* value of the boundary condition imposed is associated to the total pressure on the throat patch.

Figures 4.17, 4.18, 4.19, 4.20 show the mean velocity in the same sagittal view as in Figure 4.16. It is important to stress the fact that the y-axis points towards the inlet of the nose (this is the reason why in Figure 4.19 velocity has negative values) and that the sagittal plane has the normal directed as the x-axis. The mean velocity magnitude shows a peak around $3.2m/s$ given by the acceleration of the flow inside the nostrils. The y-component of the velocity reaches a peak value of $2.9m/s$ in the negative direction of the y-axis, while the z-component shows a peak at the height of the nostrils of about $2.5m/s$.

These values are in accordance with the work by Biondi [9] and Riazuddin et al.[24], with some small discrepancies probably due to slight differences in the definition of the boundary conditions, in particular for the volumetric flow rate imposed.

Figures 4.22, 4.23, 4.24 show the comparison between the "healthy" *P1* geometry and the "diseased" one, for which the opened anterior septal deviation of level 1 has been inserted (see Figure 4.21). In particular, Figure 4.22 reveals how a local deformation can change abruptly the flow field, resulting in a completely different pressure distribution up to the end of the olfactory region. The local effect of the deformation can be seen in Figure 4.23 and Figure 4.24: the nasal cavity restriction due to the septal deviation increases the local velocity magnitude, and the pressure suddenly has a drop visible in the right hand side of Figure 4.22; thus, the pressure keeps very low since the anterior part of the nasal cavity.

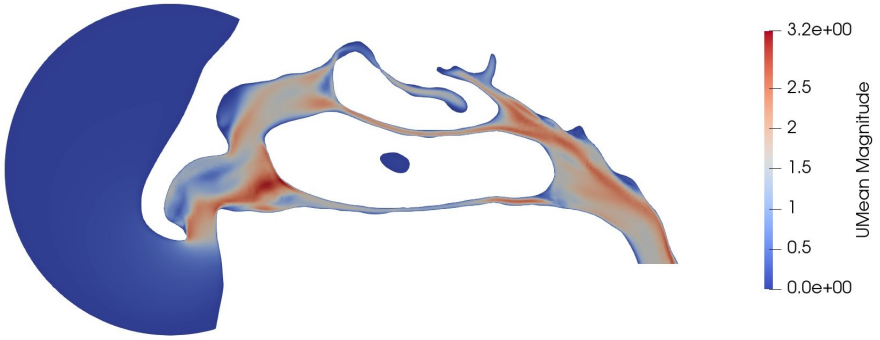


Figure 4.17: Mean velocity magnitude field on a sagittal view of the *P1* "healthy" geometry.

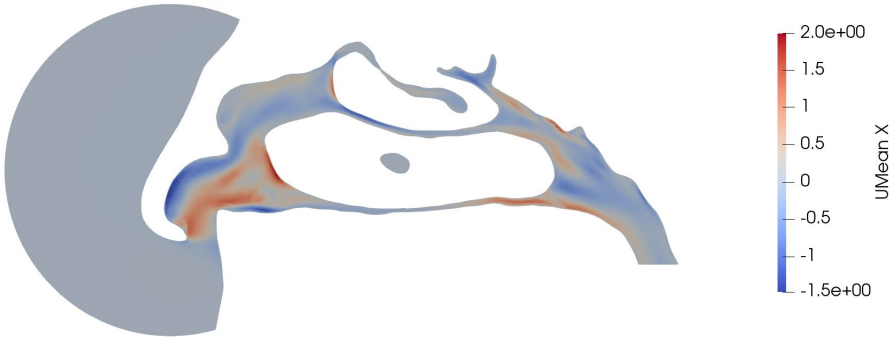


Figure 4.18: Mean x-velocity field on a sagittal view of the *P1* "healthy" geometry.

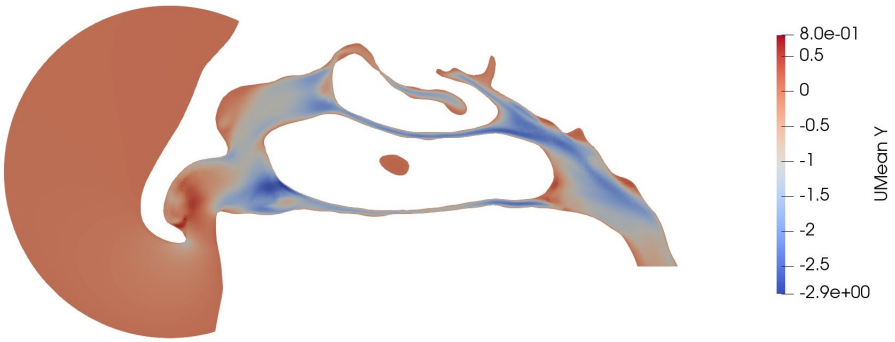


Figure 4.19: Mean y-velocity field on a sagittal view of the *P1* "healthy" geometry.

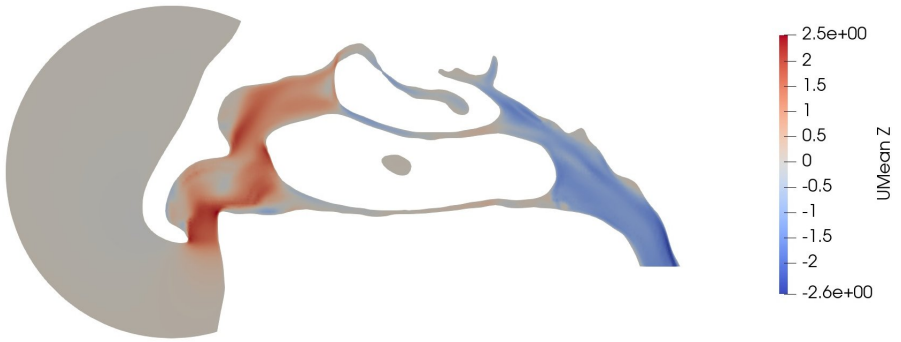


Figure 4.20: Mean z-velocity field on a sagittal view of the *P1* "healthy" geometry.

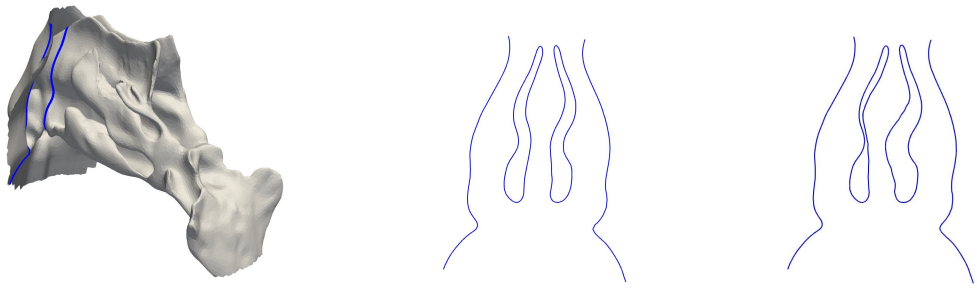


Figure 4.21: Left: section of the *P1* shape used to visualize the pathology. Center: *P2* "healthy" shape. Right: *P1* "diseased" shape.

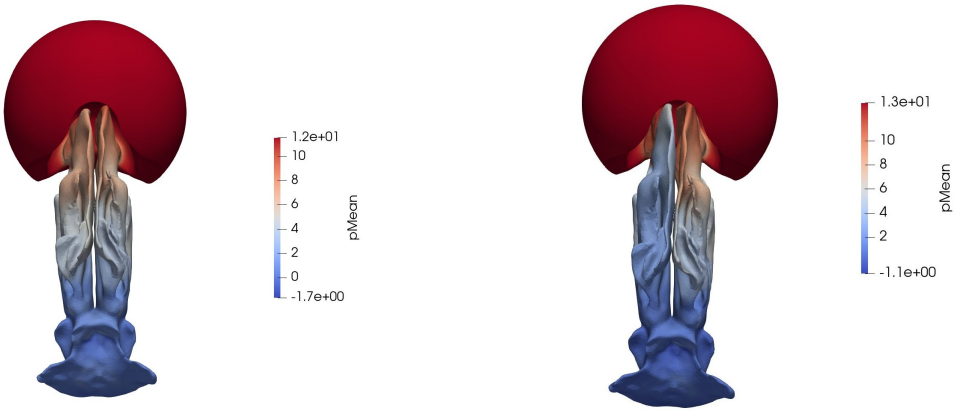


Figure 4.22: Comparison between the mean pressure field on the surface. LHS: "healthy" geometry. RHS: "diseased" geometry. View from above.

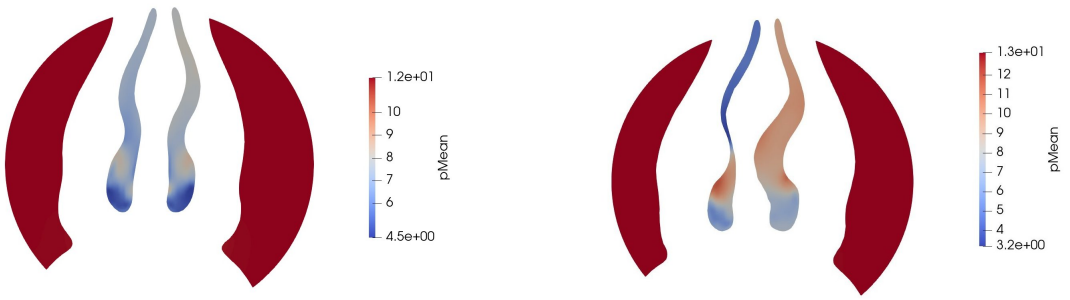


Figure 4.23: Comparison between the mean pressure field in the section shown in Figure 4.21. LHS: "healthy" geometry. RHS: "diseased" geometry.

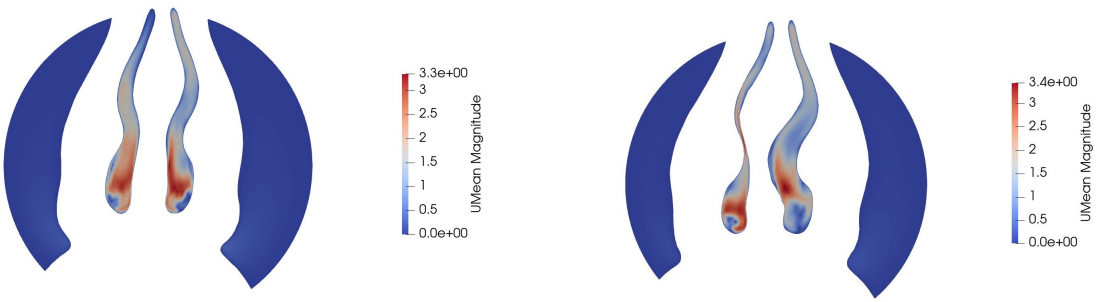


Figure 4.24: Comparison between the mean velocity magnitude field in the section shown in Figure 4.21. LHS: "healthy" geometry. RHS: "diseased" geometry.

The collected LES results are the basement of a classification network described in the next chapter, trained to infer pathologies on new patients.

5 | Machine Learning model: classification network

Artificial Neural Networks (ANNs) [26] are mathematical models that have been motivated by the functioning of the brain. They are also called *data driven models* as their ultimate goal is to *analyze data*, taking advantage of the massive amounts of data that the "big data era" produces every day.

5.1. General structure of a neural network

The basic entity of any neural network is a model of a neuron (see Figure 5.1) [27]. The idea is that a vectorial input $\mathbf{x}$ is weighted by a vector $\mathbf{w}$ and added of a bias b. The result is the argument of an activation function Φ that performs a non linear transformation and returns the output y of the neuron.

$$y = \Phi(\mathbf{w}^T\mathbf{x} + b) \tag{5.1}$$

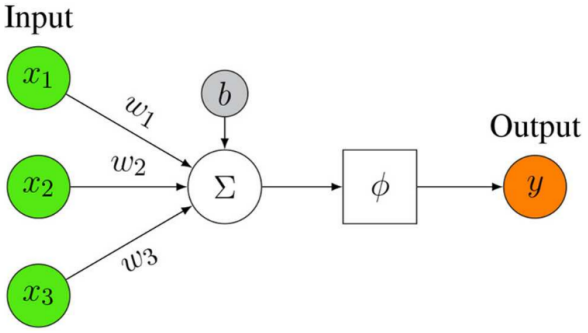


Figure 5.1: Representation of a mathematical artificial neuron model.

In order to build a neural network (NN), several neurons need to be connected with each other and structured in layers as shown in Figure 5.2. The depth of a network denotes the number of non linear transformations between the separating layers, whereas the

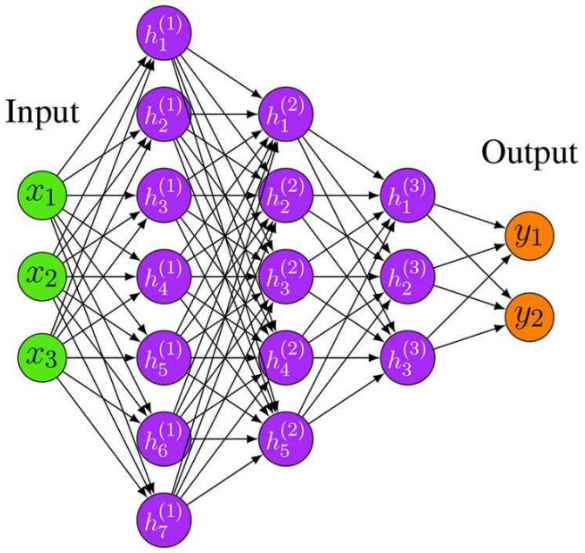


Figure 5.2: Representation of a simple neural network with 3 hidden layers denoted by $h^{(i)}$.

dimensionality of a hidden layer, i.e., the number of hidden neurons, is called width.

The output of each neuron is processed by the connected neurons in the following layer as new input value x (see Figure 5.1), whereby, in a network, the output of each neuron is the composition of all the activation functions of the sequentially backward connected neurons.

All the parameters that characterize a network (i.e. weights and biases) need to be optimized in a procedure called *training*: the NN is given input values (training set of values) for which the expected output is known; a cost function is defined on the outputs y_i and an optimization algorithm computes the optimal parameters by minimizing the error for training data.

The capability of neural networks in "predicting" something is owed to this procedure; thanks to annotated data, the model is tuned to produce expected results, and if the training set is sufficiently wide and complete in describing a given problem, the network will be probably good in describing inputs for which the output is unknown.

This is a key aspect in a CFD framework. CFD produces a huge amount of data, easily in the order of tens or hundreds of Gigabytes; this data presents a huge variability and sensitivity to small changes in the geometry. In rhinology, this has also to be combined to the immense variability that every single pathology shows and to the difficulty in gathering such data. For these reasons a ML approach to CFD is very challenging and the choice of the input values of a network is fundamental. These are also the reasons why in Section

3.3 pathologies have been inserted in a healthy patient: data augmentations allows to promote the anatomical variability, extending the available set of labeled shapes, namely, the diseased patients for which the pathologies are known, trying to make up for the lack of annotated data.

Given the amount of data that CFD returns, following a deep learning approach [28] and feeding a NN with such a huge dataset reveals currently an infeasible way to proceed, hence a selection and dimensionality reduction of the data is required. In particular, one can look for important and relevant values for the CFD, able to compactly encode the flow field properties, called *features* in a Machine Learning jargon. These features will be the input values of the neural network described in the next section.

5.2. Features extraction

The extraction of the features from the LES results is a crucial step. Features must contain information on the pathologies inserted in Section 3.3 and at the same time should be easy to retrieve and to interpret. The features used within this work are inspired by engineering practice in the analysis of flow fields [5], namely regional averages values of *time-averaged* quantities (all the quantities that will appear from here on are to be intended as time-averaged). Regional averages mimic procedures often used in fluid dynamics experiments, where probes like hot-wire anemometers or Pitot tubes are placed in the flow field to retrieve measurements.

In particular, 6 sections are identified on the *P1* shape taking the first one and the last one as the end and the beginning of the olfactory region respectively (see landmarks in Chapter 1), and the other 4 equally spaced between the previous two (see Figure 5.3). The CFD quantities taken into account on these section are:

$$\begin{aligned} |\mathbf{U}| &= \sqrt{(U_x^2 + U_y^2 + U_z^2)} \\ |\boldsymbol{\nabla} p| &= \sqrt{(\nabla p_x^2 + \nabla p_y^2 + \nabla p_z^2)} \\ E &= \frac{1}{2}(\omega_x^2 + \omega_y^2 + \omega_z^2) \\ k &= \frac{1}{2}(u_x'^2 + u_y'^2 + u_z'^2), \end{aligned} \tag{5.2}$$

respectively the velocity magnitude, the module of the pressure gradient, the enstrophy (module of the vorticity $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{U}$) and the turbulent kinetic energy. Extracting these values on the 6 surfaces has been possible thanks to the OpenFOAM *surfaces* function

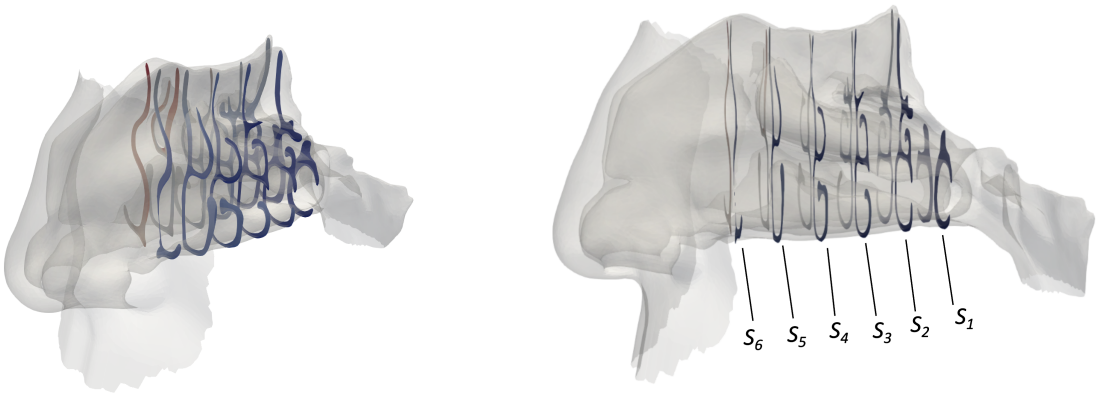


Figure 5.3: Sections extracted from LES results.

object.

Given the quantities in Equation 5.2, the regional averages on the k-region $\mathcal{R}_k$ are computed taking into account the uneven cells dimension as follows

$$\overline{q_k} = \frac{\sum_i q_i A_i}{\sum_i A_i}, \tag{5.3}$$

where index i includes all the cells $(x_i, y_i, z_i) \in \mathcal{R}_k$, and A_i denotes their areas. These values are computed separately for the left and for the right regions of each section.

An example of regional average values can be seen in Table 5.1.

5.3. Classifier

A classifier is a particular Neural Network architecture able to predict if something belongs to one class or not. Specifically, the classifier built in *Matlab* for this project is trained to perform a binary classification, evaluating if a given set of features associates to a septal deviation or to a hypertrophy. The NN takes as input a 12-features vector, corresponding to the 12 regions extracted from each geometry, 2 regions per section. The output layer consists of one neuron returning a value between 0 (associated to a hypertrophy) and 1 (associated to a septal deviation). Three fully connected hidden layers link the input to the output, having respectively 30, 20 and 10 neurons each (see Figure 5.4) for a total of 1231 trainable parameters.

Sections		*Quantities*			
		$\|\mathbf{U}\|$ $[m/s]$	$\|\boldsymbol{\nabla} p\|$ $[Pa/m]$	E $[m^2/s^2]$	k $[m^2/s^2]$
1	*Right*	1.769	268.11	6.23e+06	1.56e-04
	Left	1.709	201.60	5.26e+06	2.52e-05
2	*Right*	1.014	129.38	3.96e+06	1.62e-04
	Left	1.050	164.69	5.05e+06	2.08e-05
3	*Right*	1.063	127.64	5.01e+06	4.11e-05
	Left	1.113	109.97	5.42e+06	3.11e-05
4	*Right*	1.055	162.79	6.21e+06	1.41e-04
	Left	1.007	147.04	5.85e+06	8.98e-05
5	*Right*	1.495	177.89	7.68e+06	6.44e-04
	Left	1.183	367.68	6.92e+06	6.73e-04
6	*Right*	1.527	389.71	8.75e+06	4.10e-03
	Left	1.224	360.88	7.34e+06	9.67e-04

Table 5.1: Example of regional average values taken from one of the diseased geometry.

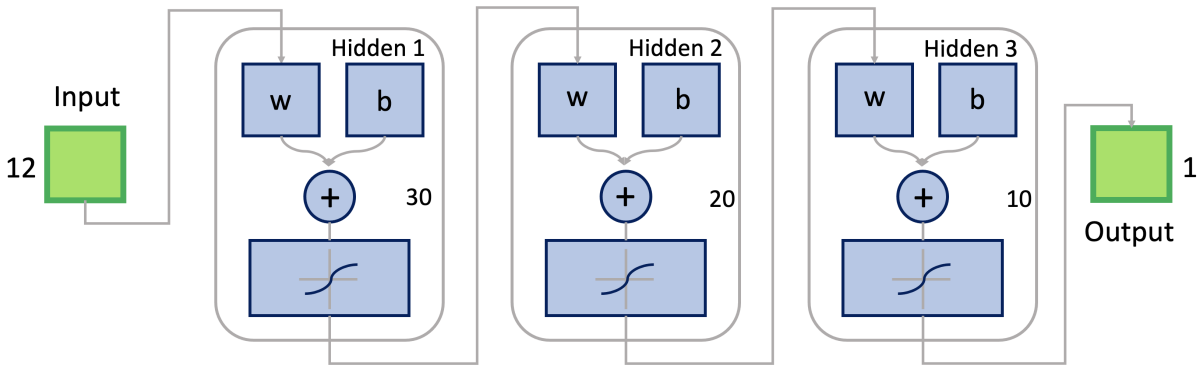


Figure 5.4: Architecture of the classifier.

All the neurons implements the *hyperbolic tangent sigmoid* [29] as activation function, that is

$$y = tansig(\tilde{x}) = \frac{2}{(1 + e^{-2\tilde{x}})} - 1, \tag{5.4}$$

where $\tilde{x}$ is the argument of Φ in Equation 5.1. The *tansig* activation function is commonly used for the simplicity in computing its derivatives.

The loss function considered is the classical loss function functions implemented in most of the classification neural networks, namely the *binary cross-entropy* [30] [31]. In the most general case, the binary cross-entropy returns the loss computing the following average:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \hat{y}_i + (1-y_i)\log\left(1-\hat{y}_i\right)\right), \tag{5.5}$$

where $\hat{y}_i$ is the model output, y_i is the actual target value and N is the output size (in this case 1). The *Loss* clearly diverges when the output value is far from the expected value, whereby a perfect model would have zero *Loss* value.

The binary cross-entropy loss function becomes the object function to be minimized in the optimization problem solved during the training. The algorithm that solves this optimization problem is the *Scaled Conjugate Gradient Backpropagation* (SCG) [32], which updates the network weights and biases considering the sensitivity that the loss function shows with respect to each parameter.

The classifier has been trained on a roughly 300-samples training dataset, fixing the number of complete passes through the whole dataset (epochs) to a max of 1000, until a best score of 90% is achieved. To evaluate the score, the training set is split in two parts during each epoch: the 85% of the samples are used for the training and the remaining 15% to assess the score.

The tests are conducted on a *never-seen-before* patient, that is the *P1*, to evaluate the performance in the inference on never-seen-before data. 4 different networks are considered, each one trained on the values of the 4 CFD quantities shown in Table 5.1.

5.4. Results

Performance of the classifiers		
Network	*Score*	*Accuracy*
$\lvert\mathbf{U}\rvert$-based	13/17	77%
$\lvert\boldsymbol{\nabla} p\rvert$-based	14/17	82%
E-based	12/17	71%
k-based	11/17	65%

Table 5.2: Performance of the classifiers.

Overall performance on the pathologies		
Network	*Hypertrophy*	*Septal Deviation*
$\|\mathbf{U}\|$-based	4/8	9/9
$\|\boldsymbol{\nabla} p\|$-based	6/8	8/9
E-based	7/8	5/9
k-based	4/8	7/9
Overall score	21/32	29/36
Overall accuracy	66%	81%

Table 5.3: Overall performance of the classifiers.

The performance assessed testing the classifiers are shown in Table 5.2 and Table 5.3.

In particular, one can figure out from Table 5.2 that the best classifier seems to be the pressure gradient based, followed by the velocity based with a similar score. Pressure gradient magnitude and velocity magnitude appear to be more informative than turbulent kinetic energy and enstrophy. This behaviour could perhaps be attributed to the fact that a modification in the geometry entails certainly a change in velocity and pressure, that is not always the case for enstrophy and turbulent kinetic energy, whose behaviour is much more difficult to be predicted.

Table 5.3 shows that a septal deviation is much easier to be predicted than a hypertrophy. The reason could be found in the type of deformations that they entail on the geometry. While a hypertrophy is always related to a restriction of a portion of the nasal cavities, a septal deviation not necessarily. This means that the hypertrophy in general influences a smaller fraction of the global flow rate, conveying less information to the flow field.

It is interesting to notice that the velocity magnitude based classifier achieves a 100% score in diagnosing septal deviations but the lowest score in predicting hypertrophies.

6 | Conclusions and further developments

The results of the experiments carried out by applying the pipeline demonstrate that ML can effectively predict functional properties (here the type of pathology) from complex and large CFD datasets, even when the flow field does not directly provide high-level diagnostic information. The potential of the approach is huge: by using such a tool, ENT doctors could get access to fundamental functional information directly conveyed by the flow field to inform their surgery decisions, thus improving the rate of success.

The training set required to achieve good results is relatively small: this is a fundamental result, that compensates for the significant cost of CFD and the difficulties in obtaining expert annotated data. In this process, a crucial role has been played by a CG tool, the functional maps, which enables increasing the number of geometries obtained from a small set of healthy patients, on which controlled, consistent and clearly defined pathologies are injected.

A set of features, based on the concept of regional averages, is extracted from the full CFD solution to reduce dimensionality. This is a crucial step, as feeding a ML model with the entire set of data produced by CFD would be unfeasible. Physical insight on the chosen features can be useful to conceive further, more informative values, and to improve the accuracy of the procedure. This can be indeed the first direction for developments. A alternate way to tackle the problem could be also a Deep Learning approach, namely feeding a Deep Learning model with as much data as possible, avoiding the procedure of extracting features, which will always loose some information in the process.

However, already at the present stage, the potential of an approach that puts together Machine Learning, CFD and CG has been demonstrated.

Bibliography

[1] S.L. Brunton, B.R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 2020.

[2] K. Duraisamy, G. Iaccarino, and H. Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51, 2019.

[3] K. Fukami, K. Fukagata, and K. Taira. Assessment of supervised machine learning methods for fluid flows. *Theor. Comput. Fluid Dyn.*, 2020.

[4] M. Raissi, A. Yazdani, and G. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367, 2020.

[5] M. Quadrio, G. Boracchi, A. Schillaci, M. Restelli, and C. Pipolo. Inferring functional properties from fluid dynamics features. *25th International Conference on Pattern Recognition (ICPR)*, 2020.

[6] C. Sundh and O. Sunnergren. Long-term symptom relief after septoplasty. *European Archives of Oto-Rhino-Laryngology*, 2014.

[7] M. Quadrio, C. Pipolo, S. Corti, R. Lenzi, F. Messina, C. Pesci, S. Saibene, and G. Felisati. Review of computational fluid dynamics in the assessment of nasal air flow and analysis of its limitations. *Eur Arch Otorhinolaryngol*, 271:2349–2354, 2014.

[8] M. Quadrio, C. Pipolo, S. Corti, F. Messina, C. Pesci, A. Saibene, S. Zampini, and G. Felisati. Effects of ct resolution and radiodensity threshold on the cfd evaluation of nasal airflow. *Med Biol Eng Comput*, 54:411–419, 2016.

[9] E. Biondi. Simulazione les del flusso nelle cavità nasali. *Master's thesis, Politecnico di Milano*, 2013-2014.

[10] J. Jin, J. Fan, M. Zeng, and K. Cen. Large eddy simulation of inhaled particle deposition within the human upper respiratory tract. *Aerosol Science*, 2006.

[11] L. Cosmo, E. Rodolà, J. Masci, A. Torsello, and M.M Bronstein. Matching deformable objects in clutter. *Fourth International Conference on 3D Vision*, 2016.

[12] M. Ovsjanikov, M. Ben Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics*, 2012.

[13] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers. Partial functional correspondence. *Computer Graphics Forum*, 2017.

[14] M. Ovsjanikov, E. Corman, M. Bronstein, E. Rodolà, M. Ben-Chen, L. Guibas, F. Chazal, and A. Bronstein. Computing and processing correspondences with functional maps. *SIGGRAPH ASIA*, 2016.

[15] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[16] D. Mumford and J. Shah. *Optimal approximations by piecewise smooth functions and associated variational problems.* Comm. Pure and Applied Math., 1989.

[17] S. Melzi, J. Ren, K. Rodolà, A. Sharma, M. Ovsjanikov, and P. Wonka. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics, Association for Computing Machinery*, 2019.

[18] R. Penrose. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, 1955.

[19] Pierre Sagaut. *Large Eddy Simulation For Incompressible Flow.* Springer, 2006.

[20] F. Ducros, F. Nicoud, and P. Thierry. Wall-adapting local eddy-viscosity models for simulations in complex geometries. *Numerical Methods For Fluid Dynamics VI*, 1998.

[21] F. Nicoud and F. Ducros. Subgrid-scale stress modelling based on the square ofthe velocity gradient tensor. *Flow, Turbulence and Combustion, Springer Verlag*, 1999.

[22] M. Weickert, G. Teike, O. Schmidt, and M. Sommerfeld. Investigation of the les wale turbulence model within the lattice boltzmann framework. *Computers and Mathematics with Applications*, 2010.

[23] J. Smagorinsky. General circulation experiments with the primitive equations. i. the basic experiment. *Month. Wea. Rev.*, 91:99–164, 1963.

[24] V. Riazuddin, M. Zubair, M. Abdullah, R. Ismail, I. Shuaib, S. Hamid, and K. Ahmad. Numerical study of inspiratory and expiratory flow in a human nasal cavity. *of Medical and Biological Engineering*, 31:201–206, 2010.

[25] D. Wexler, S. Rebecca, and J. Kimbell. Aerodynamic effects of inferior turbinate reduction computational fluid dynamics simulation. *Arch Otolaryngol Head Neck Surg*, 31:1102–1107, 2005.

[26] C.C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

[27] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer. An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, 2020.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[29] J. Ledere. Activation functions in artificial neural networks: A systematic overview. *Department of Mathematics, Ruhr-University Bochum*, 2021.

[30] V. Yendapalli. Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 2020.

[31] K. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT, 2012.

[32] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks: the official journal of the International Neural Network Society*, 1993.

A | Appendix A

In this appendix all the pathologies inserted by hand by an ENT doctor are shown in the coronal plane. Figures with four columns show in the right hand side the two levels of severity considered. The shape alongside the 3D geometry shows the 'healthy' section.

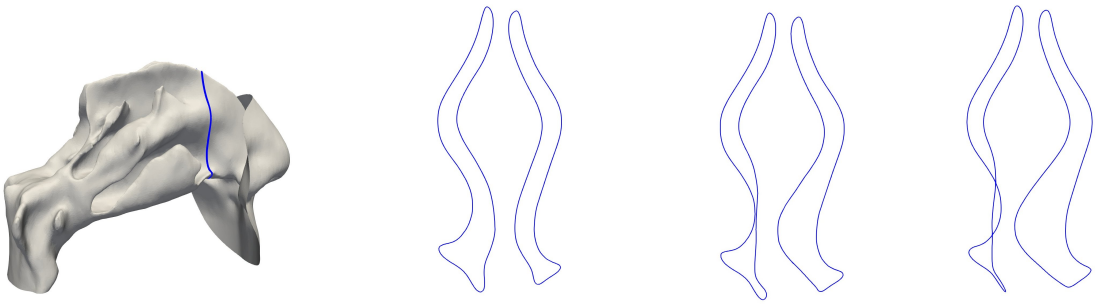


Figure 1: Septal deviation, opened, anterior, inferior.

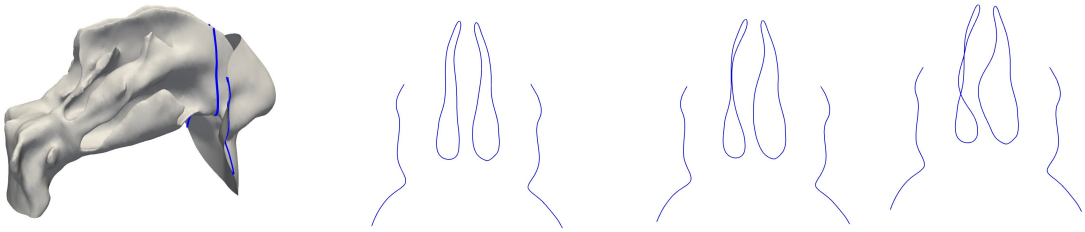


Figure 2: Septal deviation, opened, anterior, medium.

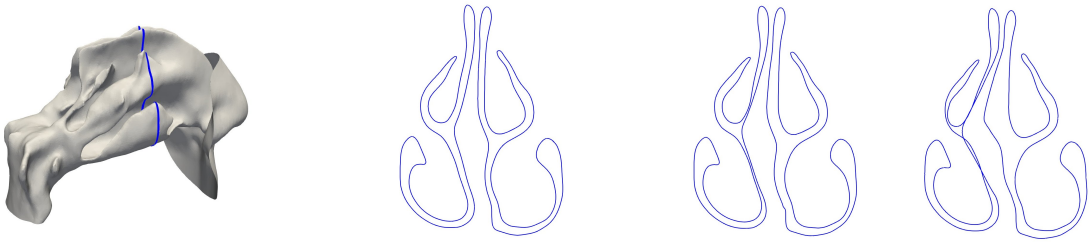


Figure 3: Septal deviation, opened, posterior, middle.

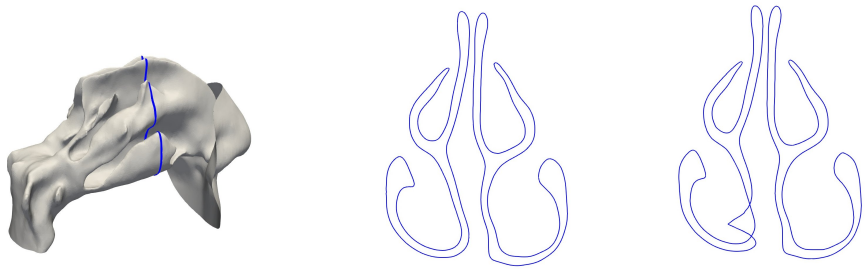


Figure 4: Endoscopic septal deviation, anterior, inferior

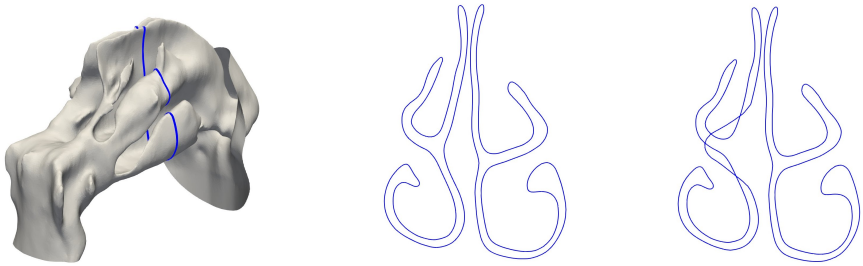


Figure 5: Endoscopic septal deviation, anterior, middle

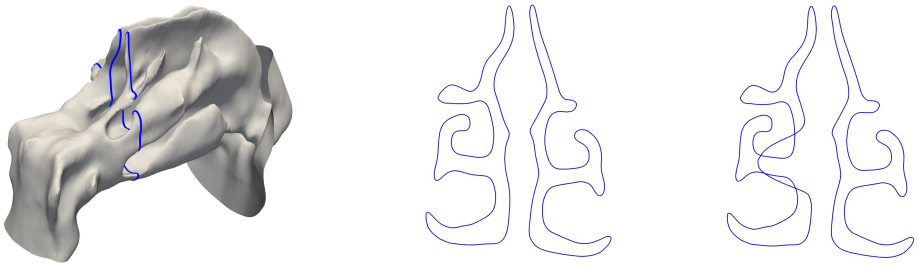


Figure 6: Endoscopic septal deviation, posterior, medium

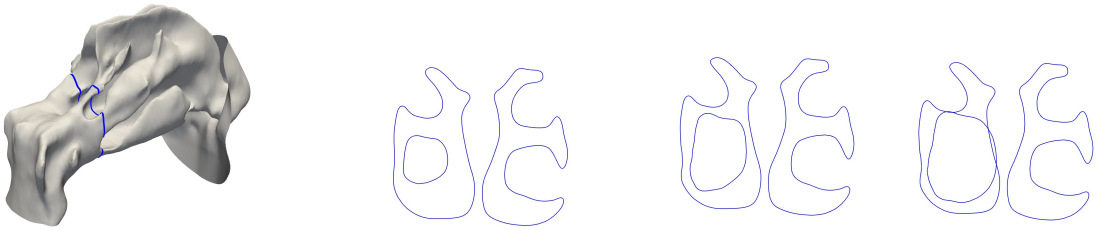


Figure 7: hypertrophy, inferior, tail

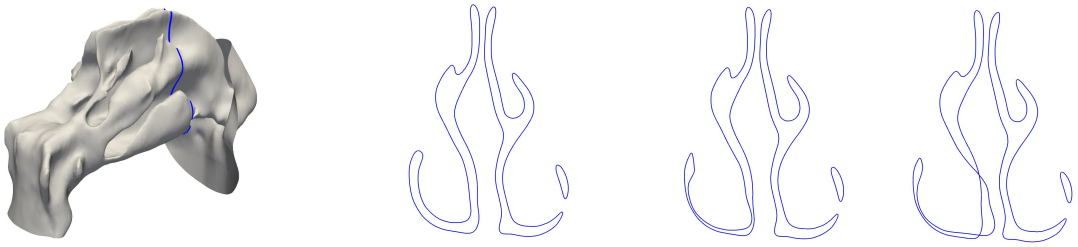


Figure 8: hypertrophy, inferior, head

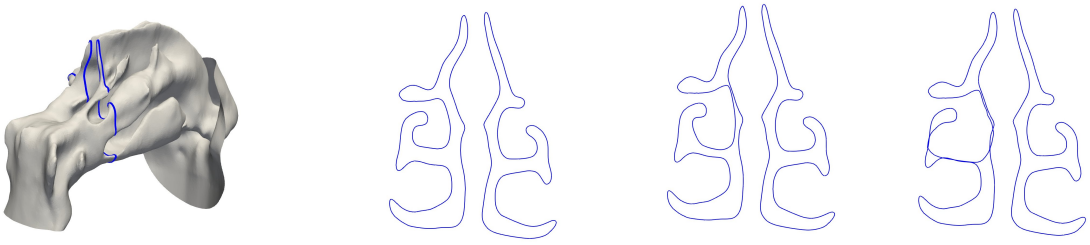


Figure 9: hypertrophy, medium, tail

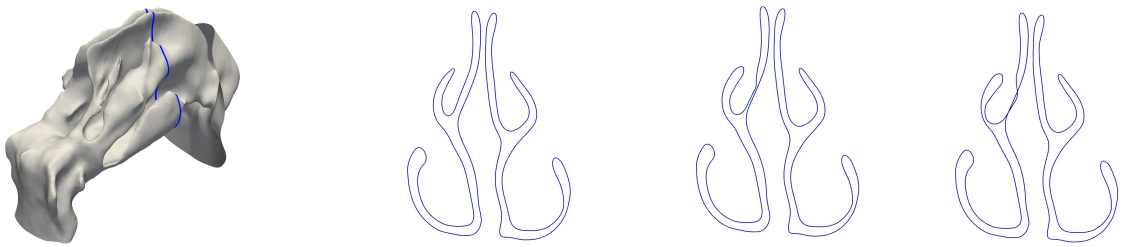


Figure 10: hypertrophy, medium, head

Acknowledgements

Colgo l'occasione per esprimere estrema gratitudine a tutta la mia famiglia che mi ha sempre ribadito quanto fosse speciale quello che ho fatto; spero di averli ripagati dandogli un motivo in più di orgoglio.

Un ringraziamento dal profondo del cuore ai miei amici e amiche, che nel corso degli anni sono sempre stati lì a sostenermi, spesso avendo più fiducia in me di quanta ne avessi io.

Ed infine, un grazie particolare al Prof. Maurizio Quadrio e all'Ing. Andrea Schillaci, senza il cui sostegno e aiuto questo lavoro non sarebbe stato possbile.

Riccardo