Final Master Project

Master in Neuroengineering and Rehabilitation

3D vessel reconstruction based on intra-operative intravascular ultrasound for robotic autonomous catheter navigation

THESIS

Author:	Maria Montserrat Ainchil Cayuela
Director:	**Prof. dr.** Jos Vander Sloten
Co-director:	**Prof. dr.** Emmanuel Vander Poorten
Supervisor:	Miguel Angel Mañanas Villanueva
Summoning:	April 2023

ETSEIB

FACULTEIT INGENIEURSWETENSCHAPPEN

UPC

KU LEUVEN

Abstract

In recent years, robotic technology has improved instrument navigation precision and accuracy, and helped decrease the complexity of minimally invasive surgery. Still, the inherent restricted access to the anatomy of the patients severely complicates many procedures. Interventionists frequently depend on external technologies for visual guidance, usually employing ionizing radiation, due to the limited view upon the surgical scene. In the case of endovascular procedures, fluoroscopy is the common imaging modality used for visualization. This modality is based on X-rays and only offers a two-dimensional (2D) view of the surgical scene.

Having a real-time, up-to-date understanding of the surrounding environment of the surgical instruments within the vasculature and not depending on using ionizing radiation would not only be very helpful for interventionists, but also paramount for the navigation of an intraluminal robot. Therefore, the aim of this thesis is to develop an algorithm able to do an intra-operative and real-time three-dimensional (3D) vessel reconstruction.

The algorithm is divided into two parts: the reconstruction and the merging. In the first one, it is obtained the 3D vessel reconstruction of a section of the vessel and in the second one, the different sections of 3D vessel reconstruction are combined. A real vessel mesh is used to calculate the fitting errors of the reconstructed vessel which are very small.

Resumen

En los últimos años, la tecnología robótica ha mejorado la precisión y fiabilidad de la navegación de instrumentos y ha ayudado a disminuir la complejidad de la cirugía mínimamente invasiva. Aún así, el acceso restringido inherente a la anatomía de los pacientes complica gravemente muchos procedimientos. Los intervencionistas dependen con frecuencia de tecnologías externas para la guía visual, generalmente empleando radiación ionizante, debido a la visión limitada de la escena quirúrgica. En el caso de los procedimientos endovasculares, la fluoroscopia es la modalidad de imagen común utilizada para la visualización. Esta modalidad se basa en rayos X y solo ofrece una vista bidimensional (2D) de la escena quirúrgica.

Poder saber en tiempo real y de forma actualizada como es el entorno alrededor de los instrumentos quirúrgicos que se encuentran dentro de la vasculatura y no depender del uso de radiación ionizante no solo sería muy útil para los intervencionistas, sino también fundamental para la navegación de un robot intraluminal. Por lo tanto, el objetivo de esta tesis es desarrollar un algoritmo capaz de realizar una reconstrucción tridimensional (3D) del vaso sanguíneo de forma intraoperatoria y en tiempo real.

El algoritmo se divide en dos partes: la reconstrucción y la unión. En la primera se obtiene la reconstrucción 3D de una sección del vaso sanguíneo y en el segundo se combinan las diferentes secciones obtenidas de vasos sanguíneos reconstruidos en 3D. Se utiliza una malla de un vaso sanguíneo real para calcular los errores de ajuste del vaso sanguíneo reconstruido, son errores muy pequeños.

Resum

En els últims anys, la tecnologia robòtica ha millorat la precisió i la fiabilitat de la navegació dels instruments i ha ajudat a disminuir la complexitat de la cirurgia mínimament invasiva. Tot i així, l'accés restringit inherent a l'anatomia dels pacients complica greument molts procediments. Els intervencionistes sovint depenen de tecnologies externes per a la guia visual, normalment emprant radiacions ionitzants, a causa de la visió limitada de l'escena quirúrgica. En el cas dels procediments endovasculars, la fluoroscòpia és la modalitat d'imatge comuna utilitzada per a la visualització. Aquesta modalitat es basa en raigs X i només ofereix una visió bidimensional (2D) de l'escena quirúrgica.

Poder saber en temps real i de forma actualitzada com és l'entorn al voltant dels instruments quirúrgics que es troben dins de la vasculatura i no depèn de l'ús de radiació ionitzant no només seria molt útil per als intervencionistes, sinó també fonamental per a la navegació d'un robot intraluminal. Per tant, l'objectiu d'aquesta tesi és desenvolupar un algorisme capaç de fer una reconstrucció tridimensional (3D) del vas sanguini de forma intraoperatòria i en temps real.

L'algorisme es divideix en dues parts: la reconstrucció i la fusió. En la primera s'obté la reconstrucció en 3D d'una secció del vas sanguini i en la segona, es combinen les diferents seccions obtingudes de vasos sanguinis reconstruïts en 3D. S'utilitza una malla d'un vas sanguini real per calcular els errors d'ajust del vas sanguini reconstruït, els errors son molt petits.

Contents

List of figures

1. Glossary

General notations

{●} Coordinate frame ●

${}_{b}\bullet^{a}$ ● of frame {a} expressed in frame {b}

${}_{b}T^{a}$ Homogeneous transformation matrix expressing frame $\{a\}$ with respect to frame $\{b\}$

${}_{b}R^{a}$ Rotation matrix expressing the orientation of frame {a} with respect to frame {b}

Abbreviations

3D: Three-dimensional

2D: Two-dimensional

CVD: Cardiovascular disease

US: Ultrasound

IVUS: Intravascular ultrasound

EM: Electromagnetic

DOFs: Degrees of freedom

CT: Computed Tomographic

SCEM: Simultaneous Catheter and Environment Modelling

MR: Magnetic Resonance

GNNs: Graph Neural Networks

CFST: Concrete-filled steel tube (CFST)

MVS: Multi-View Stereo

SfM: Structure from Motion

UKF: Unscented Kalman Filter

IQR: Interquartile Range

TCM: Triangular Connectivity Matrix

2. Introduction

2.1. Endovascular catheterization

Globally, cardiovascular disease (CVD) is the leading cause of death. According to estimates, 17.9 million deaths worldwide in 2019 were attributable to CVDs, or 32% of all fatalities. Heart attacks and strokes were to blame for 85% of these fatalities [1].

Endovascular procedures have been increasingly embraced for diagnosis and treatment within the cerebral vasculature (interventional neuroradiology), peripheral vascular system and cardiac interventions (interventional cardiology), areas primarily affected in patients with CVDs [2]. The technique involves performing a challenging minimally invasive procedure in which guidewires and catheters are guided through the patient's blood vessels.

Nowadays, many CVD, such as aneurysms, that traditionally needed open surgery for treatment may be handled through endovascular interventions. It offers many advantages, including a quicker recovery time, less pain, smaller incisions, local or regional anesthesia instead of general anesthesia, less strain on the heart and fewer risks for patients with other medical conditions [3], [4]. Most contemporary endovascular procedures are carried out under fluoroscopic guidance and the progress of the catheter is controlled by manual manipulation at its proximal end.

However, there are some drawbacks, such as exposure to high doses of X-ray radiation and the lack of contact force sensing from the endovascular instruments and the vascular architecture X-ray images. Furthermore, because of their intrinsic compliance and typical manual manipulation from outside the patient, traditional catheters are challenging to maneuver. Therefore, adverse outcomes brought on by sudden contact between the catheter tip and the vascular wall might occur as a result of the clinician's general poor situational awareness. Examples of such occurrences include the perforation of vessels and the embolization of calcified plaques, both of which may have severe neurological effects [5].

Robotic systems with embedded actuators and/or sensors have been developed to decrease the complexity of endovascular surgery. These robotic systems are formed by a

catheter and a remote-control system which allow doctors to control the robot while being distant from the patient, thereby preventing the surgeon from being exposed to X-ray radiation. By using a robotic system, the operator may take highly accurate measurements of the length of the lesion, which may result in using fewer stents per lesion and decreasing the likelihood of longitudinal geographic miss, in which the stent is positioned incorrectly, reducing the incidence of late recurrent events. With embedded actuators, robotic catheters intend to reduce the physical and mental load of the clinician by providing superior controllability over their distal tip. In particular, robotic systems with steerable catheters make possible to perform procedures or portions of procedures that would be challenging or impossible to complete with conventional catheters [5]–[8]. In observational studies and non-randomized studies [6], the basic feasibility and benefits of robotic interventions in cardiovascular applications have been reported: increased procedural speed, accuracy, and reproducibility; decreased exposure to radiation and contrast agent; reduced surgical trauma and shorter patient recovery time.

Despite the fact that robotic catheters provide several benefits over traditional catheters, visual feedback is still mostly relying on fluoroscopy. Therefore, the situational awareness of the clinician is still impaired. Clinicians need to correlate the 2D fluoroscopic images with the 3D geometry of the anatomy of the patient they have knowledge of and map the steering commands to the catheter motion in these 2D and 3D representations. This continuous mental mapping prevents the clinicians to take full benefit from the additional mobility provided by robotic systems.

Alternative radiation free-sensing modalities such as 3D ultrasound (US) imaging and intravascular ultrasound (IVUS) imaging, have been considered in the literature to better visualize the surroundings of blood vessels, localize instruments in a 3D space and/or monitor catheter-vessel interactions without resorting to ionizing X-rays [7]. These modalities use EM pose sensing as it enables the spatial localization of the catheter tip with respect to the surrounding vessel. EM sensors are tracked and localized relative to an external coordinate system, while posing in six degrees of freedom (DOFs).

Since an IVUS probe only produces a single cross-sectional US image from inside the vessel, it only gives limited information about the vessel. Such characteristic largely complicates the direct use of this imaging modality as a path planning tool. Pre-operative imaging-derived 3D vascular models may provide insight into the anatomy of the patient. However, intra-operative local 3D vessel representations have the potential to significantly

decrease the use of contrast agents and exposure to ionizing radiation during endovascular procedures, while overcoming the 2D visualization limitation of fluoroscopic guidance [9]. These representations should be updated in real-time, intra-operatively, to reflect the deformation of the vasculature.

2.2. Three-dimensional vessel reconstruction

The aim of reconstruction is to find the optimum approximation surface for the data acquired in the acquisition phase. This issue is characterized by the type of information gathered during the acquisition phase or the approach used to reconstruct [10]. Fluoroscopic images, the gold standard guiding surgeons during endovascular catheterization are used as input data. Nevertheless, the generalized use of fluoroscopic images has been reduced and only used for complementary actions since they rely heavily on the use of contrast material which exposes to radiation the doctor and patient. Moreover, the images obtained from it have poor visibility of anatomical structures and there is an absence of depth information. Alternatively, using IVUS and EM data can provide radiation-free data with pose sensing. Deep learning and deformable models are others approaches to reconstruct. On one hand, deep learning-based methods can handle enormous amounts of data and learn complex patterns, in contrast to conventional methods. On the other hand, deformable models provide an abstract model of an object class by modeling separately the variability in shape, texture, or imaging conditions of the objects in the class. It is a very powerful segmentation technique to reconstruct the walls of the arteries. From the camara, two types of information can be acquired: photos or point cloud data, one of the most popular digital representation formats for 3D content.

There are different methods to reconstruct using fluoroscopic images. G. Hichem et al. suggest obtaining a 2D image of the vascular tree of the human retina using image segmentation and create a 3D reconstruction model by combining the approach of the vessel tracking algorithm with the use of a continuous model of generalized cylinder [11]. PT. Tran et al. propose to combine the dense information in 2D extracted from fluoroscopic images with the discrete pose information of EM sensors to provide a reliable reconstruction of a full 3D catheter shape [12].

In 2016, various papers proposed the use of IVUS and EM data. The first method, published in February 2016, proposes a new image processing method based on sensor

fusion between IVUS imaging and EM tracking to realize fully automatic processing of IVUS imaging and 3D reconstruction in real time for endovascular aortic stent grafting, as well as branch detection for alignment and deployment of the stent grafting with the vasculature branches. In this case, Computed Tomographic (CT) data is used for complementary navigation allowing an efficient catheter advancement and assistant clinical judgment [13]. In June 2016 was published a second method that proposes a technique for endovascular navigation based on IVUS imaging and EM sensing called Simultaneous Catheter and Environment Modelling (SCEM). Vessel structure information from pre-operative CT/Magnetic Resonance (MR) imaging is used to avoid radiation exposure and contrast agents. This method relies on precise registration between EM and pre-operative data to recover the 3D structure of the vasculature together with the position of the catheter tip intraoperatively, allowing for the provision of knowledge about the interactions between the catheter and its surroundings [14]. A month later, in July 2016, is published SCEM+, a more robust method than SCEM, that proposes to formulate the 3D vessel reconstruction as a nonlinear optimization problem based on the pre-operative data. This allows vessel reconstruction in real-time and deals with measurement errors from both EM sensors and IVUS images [15]. SCEM and SCEM+ rely on accurate registration between EM and pre-operative data. In October 2016, was published an approach that suggests a registration-free vessel reconstruction method that combined with the 3D vessel reconstruction using IVUS, EM, and pre-operative data, estimates and updates EM-CT registration intra-operatively. This framework improves SCEM+ because it can handle global motion and periodic vascular deformation brought on by the cardiac cycle and does not require any prior knowledge of EM-CT registration [16].

Deep learning uses data processing and advanced pattern learning to tackle 3D reconstruction. Two approaches are suggested utilizing various sorts of input data. The first approach proposes a four-ocular vision system for the 3D reconstruction of large-scale concrete-filled steel tube (CFST) under complex testing conditions. To sample the large-scale CFST, a four-ocular vision system is built. A 3D point cloud of the specimen surface is obtained by using point cloud capture, point cloud filtering, and point cloud stitching techniques. A point cloud correction algorithm based on geometric features and a deep learning algorithm are utilized, respectively, to correct the coordinates of the stitched point cloud. This raises the 3D model accuracy for use in real-time complicated surface monitoring, improving the vision measurement accuracy in complex situations [17] The second approach proposes a parallel aggregation network with a newly created global

layer for extracting spatial features from a random walk normalized matrix to recover a human mesh from a single image. This method wants to solve one problem that frequently arises when utilizing Graph Neural Networks (GNNs) to reconstruct a single-image human mesh, which is the absence of global information in the spatial feature aggregation of the current GNNs. The restored human mesh might end up with an undesirable deformity and being inaccurate as a result. By applying this method, the human feature may be added to the mesh using the coarse body mesh (head, hand, foot, etc.) offered by the coarsening network. The local and global spatial features are aggregated to update vertex coordinates [18].

A method that uses deformable model proposes to reconstruct the surface of IVUS coronary arterial walls using a simplified version of a deformable model known as the 3D topologically adaptable snake model [19].

There are two traditional approaches for 3D reconstruction methods based on images. The first one is Multi-View Stereo (MVS) and its goal is to reconstruct a complete 3D object model from a collection of images taken from known camera viewpoints [20]. The second approach is Structure from Motion (SfM) and consists of the process of reconstructing 3D structure from its projections into a series of images taken from different viewpoints [21]. These methods require a sufficient number of photos with a small baseline viewpoint difference. Moreover, they either need to compute these camera attributes or depend on known camera calibration (internal and external). To eventually calculate the representation of the 3D form and lift these photos collectively from 2D to 3D, they must also compute a correspondence between images. However, the type of images needed for SfM are very different from IVUS images.

Several attempts have been made to perform a mesh reconstruction from point clouds. The technique varies if one is working with an unstructured point cloud generated by multiple image matching or a structured one, generated by one image. S. Kim et al. propose a method to estimate surface normals of the vertical points within an unstructured point cloud considering that the process of surface reconstruction depends on determining precise surface normals, which are occasionally calculated inaccurately [22]. X. Qin et al. suggest the construction of an octree structure that searches for flat areas and controls edge growing based on compulsive restriction and optimization criteria to obtain an optimal mesh surface [23]. K. Kwon et al. propose an iterative offset-based method for reconstructing a mesh model from the point cloud of a pig by creating a primitive shape in

the form of a mesh. This mesh is deformed based on the distance from the primitive shape to the points from the point cloud. This procedure is repeated to reshape the mesh model to fit the shape of the corresponding point cloud [24].

2.3. Thesis objectives

The aim of this thesis is to develop a method to do intra-operative real-time 3D vessel reconstruction. The method must avoid exposing the patient and clinician to ionizing radiation. Hence, the data used to develop the method is IVUS and EM data. The goal of the algorithm is to allow a current awareness of the environment within the vasculature to improve the situational awareness of the clinician.

2.3.1. Specific objectives

The specific objectives of this thesis are:

- Investigate IVUS-based methods using EM pose sensing and/or shape sensing for real-time 3D vessel reconstruction.
- Develop a method to do intra-operative real-time 3D vessel reconstruction.
- Provide tools to visualize the obtained reconstruction.
- Verify and validate the proposed method.

2.4. Thesis scope

The algorithm developed is divided into two parts: the reconstruction and the merging. In the first one, it is obtained the 3D vessel reconstruction of a section of the vessel and in the second one, the different sections of 3D vessel reconstruction are combined. A real vessel mesh is used to calculate the fitting errors of the reconstructed vessel which are very small.

3. Three-dimensional cylinder-based global lumen reconstruction

3.1. Data

The data that have been used for this work are IVUS data, EM data and cylinder model data.

3.1.1. Origin of the data

The catheter used to gather the needed measurements from IVUS and EM tracking is a robotic catheter with a distal active segment. The design of the catheter and the sensors used are detailed in [9] as the catheter used to gather the data is the same one used to estimate an intra-operative local 3D vessel representation.

***Figure 1**. Robotic catheter with one EM sensor and an IVUS probe embedded (within the orange circle) used in [9].*

The IVUS measurements are the contour points of the vessel. Figure 2, which shows a cross-sectional 2D US view of the vessel at the level of the IVUS probe, clearly illustrates these aspects. Since this sensor is aligned with the longitudinal axis of the catheter, the cross-section is thus perpendicular to this longitudinal axis and is visible in the xy-plane of the IVUS coordinate frame {i} and is rigidly attached to the center of the IVUS probe. The EM pose sensing information are the three EM position values: x, y and z; and the four EM orientation quaternion values: x, y, z and w.

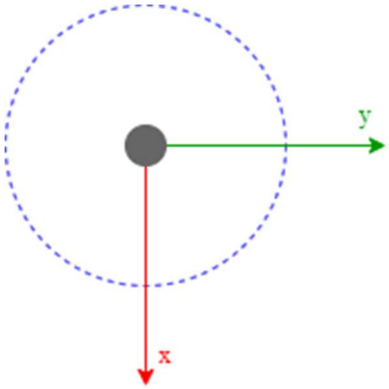

Figure 2. *IVUS cross-section in the xy-plane of the IVUS frame {i}*

The contour of the vessel lumen can be extracted from the IVUS slice and is represented by a set of M consecutive 2D points ${}^{j}_{i}c$ spaced every $\frac{2\pi}{M-1}$ radians.

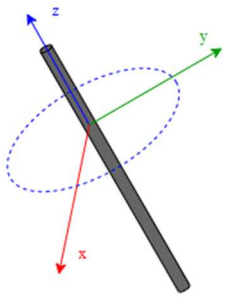

Figure 3. *IVUS points from a slice in the IVUS frame {i}*

From the IVUS and EM data mentioned, a local model of the geometry of the vasculature is generated from an algorithm described in [9]. The model has a cylinder shape, as the algorithm estimates the vessel geometry and its own coordinate frame, {cyl} frame. This cylinder model is also used for this work.

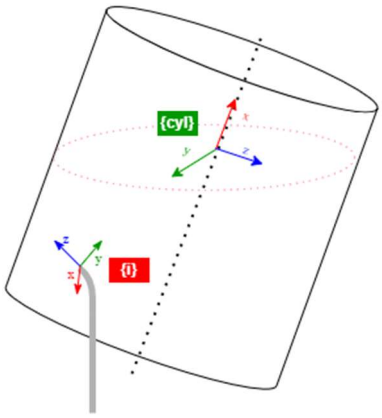

Figure 4. *Cylinder representation in {cyl} frame and in {i} frame.*

3.1.2. Preparation of the data

The data used are IVUS data, EM data and cylinder models. In this particular case, as data have their own coordinate frame, they must be in the same frame before taking any further step. Four coordinate frames are used: {i} frame that corresponds to the IVUS frame, {e} frame that corresponds to the EM frame, {cyl} frame that corresponds to the cylinder frame and the {w} frame that corresponds to the world frame, a fixed cartesian coordinate frame to the environment, not to the moving catheter.

The IVUS probe and robotic catheter tip poses are determined by measuring the EM sensor 6-DOFs posed when positioned in a known electromagnetic field. The IVUS probe pose is represented by the homogeneous transformation matrix ${}_{w}T^{i}$, describing the pose of the {i} frame in the {w} frame. The IVUS probe pose is determined from the EM sensor pose ${}_{w}T^{e}$ and from the constant pose ${}_{e}T^{i}$ of the IVUS probe relative to the EM sensor. ${}_{e}T^{i}$ is constant because the catheter is designed so the EM and IVUS sensors are in the same positions with respect to each other when the data is being collected.

These relations are summarized as:

$$ {}_{w}T^{i} = {}_{w}T^{e} \cdot {}_{e}T^{i} $$

Equation 1. *Relation between the {i}, {e} and {w}.*

IVUS data are transformed into the {*w*} using the transformation matrix ${}_{w}T^{i}$ and are also transformed into the {cyl} using a transformation matrix: ${}_{cyl}T^{w}$. EM data are transformed into the {*w*} using the transformation matrix ${}_{w}T^{e}$.

$$ {}_{w}T^{i} = \begin{bmatrix} {}_{w}R^{i} & {}_{w}t^{i} \\ 0_{1x3} & 1 \end{bmatrix} $$

Equation 2. *Homogeneous transformation matrix of frame {i} with respect to frame {w}.*

$$ {}_{w}T^{e} = \begin{bmatrix} {}_{w}R^{e} & {}_{w}t^{e} \\ 0_{1x3} & 1 \end{bmatrix} $$

Equation 3. *Homogeneous transformation matrix of frame {e} with respect to frame {w}.*

${}_{w}R^{i}$ and ${}_{w}R^{e}$ are the rotation matrices of frames {i} and {e} with respect to the frame {w} and ${}_{w}t^{i}$ and ${}_{w}t^{e}$ are the translation vector of frames {i} and {e} with respect to the frame {w}.

There is a relationship between the {cyl} and {i} frame (see Equation 4) because the cylinder model is created as more IVUS data are collected. ${}_{i}T^{cyl}$ is the transformation matrix of frame {cyl} with respect to the express to the frame {i}.

$$ {}_{i}T^{cyl} = \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 & ip_x \\ \sin(\theta)\cdot\cos(\varphi) & \cos(\theta)\cdot\cos(\varphi) & -\sin(\varphi) & ip_y \\ \sin(\theta)\cdot\sin(\varphi) & \cos(\theta)\cdot\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 4. *Homogeneous transformation matrix of frame {cyl} with respect to frame {e}.*

The position vector of the cylinder, ip, is defined so that it always lies on the xy plane of frame {i}.

IVUS data are transformed from the frame {w} into the frame {cyl} for each cylinder using ${}_{cyl}T^{w}$.

$$ {}_{w}T^{cyl} = {}_{w}T^{e} \bullet {}_{e}T^{cyl} \rightarrow {}_{cyl}T^{w} = ({}_{w}T^{cyl})^{-1} $$

Equation 5. *Homogeneous transformation matrix of frame {cyl} with respect to frame {w}.*

3.1.3. Cylinder model

Cylinder models are generated by starting with a circle of N nodes and a radius R, then adding L-1 levels of nodes to create a cylinder. The cylinder model created does not match the local model described in [9]. The bottom base center point of the cylinder model is located at the center of coordinate frame (0,0,0). However, the local model presents some difference with the cylinder created.

First, the local model is defined taking into account that the center of the cylinder is on the center of coordinates of the frame. For this reason, a transformation matrix is applied to the cylinder model considering zz the height of the cylinder (see Figure 5).

$$ {}_{center}T^{bottom} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -zz/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 6. *Homogeneous transformation matrix that translates the center of the cylinder to the center coordinate frame {cyl}.*

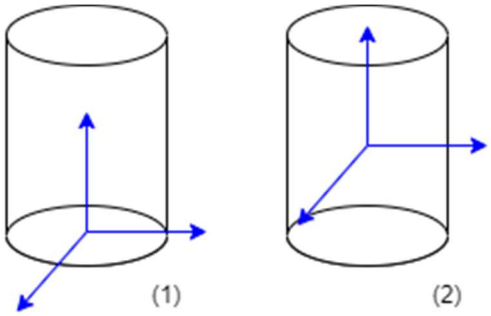

Figure 5. *(1) Cylinder with the center of the coordinate frame at the center point of the bottom base (2) Cylinder with the center of the coordinate frame at the center of the cylinder.*

Second, the longitudinal axis of the cylinder model is parallel with z axis, whereas in the local model is parallel with x axis (see Figure 6).

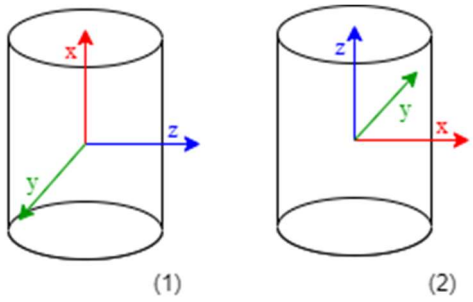

Figure 6. *(1) Cylinder with the x axis parallel to the longitudinal axis (2) Cylinder with the z axis parallel to the longitudinal axis.*

As the equations being used are from [9] and x axis is defined to be parallel to the longitudinal axis, all IVUS data need to be rotated $\alpha = \frac{\pi}{2}$ radians around the y axis using a rotation matrix (Equation 7) in order to match the way data is described.

$$R_y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 7. *Rotation matrix around the y axis.*

The cartesian coordinate system is used to define the source data, IVUS data and cylinder model data. However, these data have been converted into cylindrical coordinates using these equations (Equation 8) in order to be able to perform the reconstruction calculations more effectively.

$$r = \sqrt{x^2 + y^2} \quad \theta = \tan^{-1}\frac{x}{y} \quad z = z$$

Equation 8. *Cartesian (x,y,z) to Cylindrical (r,θ,z) coordinates transformation.*

3.2. Proposed reconstruction method

The reconstruction method proposed in this thesis is inspired by the iterative offset-based method introduced by K. Kwon et al. [24]. This approach was chosen because the input data shares similar characteristics to the data used in this thesis, IVUS: both are point clouds, and the approach uses a primitive shape, whereas the cylinder model is employed in this thesis. The basic principles of the original approach of mesh-model reconstruction process are as follows. First, a primitive shape is created in the form of a mesh and the distance from each node to the points from the point cloud is calculated. The nodes are moved according to the distance calculated. This procedure is repeated R times to reshape the mesh model to fit the shape of the corresponding point cloud (Figure 7). The reconstruction method applied follows only the first three steps: creating a primitive shape, estimating the distance to move and moving nodes.

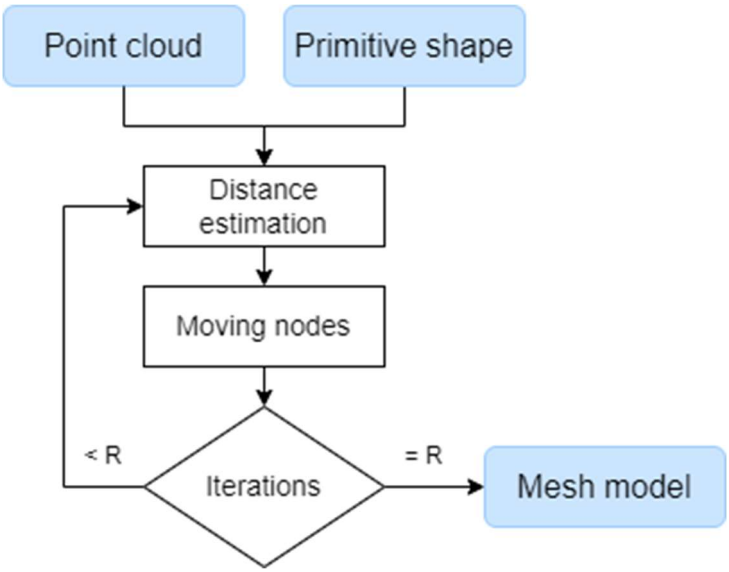

Figure 7. *Proposed reconstruction method.*

3.2.1. Primitive shape

The primitive shape is obtained by estimating cylinder models (see Figure 8) using the algorithm described in [9]. These models are used to have a better approximation of how the vessel surface really is. As one of the goals of this approach is to be real-time, as the catheter moves and more IVUS data is being gathered, new cylinders will be created. As new cylinders are created, the frame $\{cyl\}_k$ is not stationary and changes at each time step k.

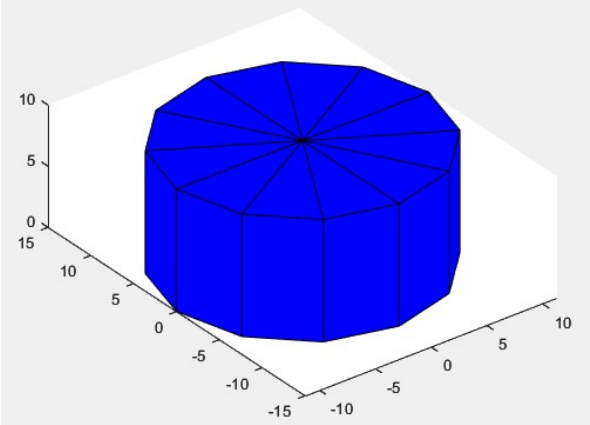

Figure 8. *Example of cylinder of N=12 nodes.*

3.2.2. Distance estimation

In order to estimate the distance from the primitive shape mesh to the points from the point cloud, K. Kwon et al. propose 4 steps:

1. Finding the nearest triangle for a given point.

   To properly distort the mesh, triangles are used to build each of its faces. To determine which point in the point cloud is closer to every triangle of the mesh model and calculate the distance between them, the cylinder model is split into L levels having then N slices per level (see Figure 9).

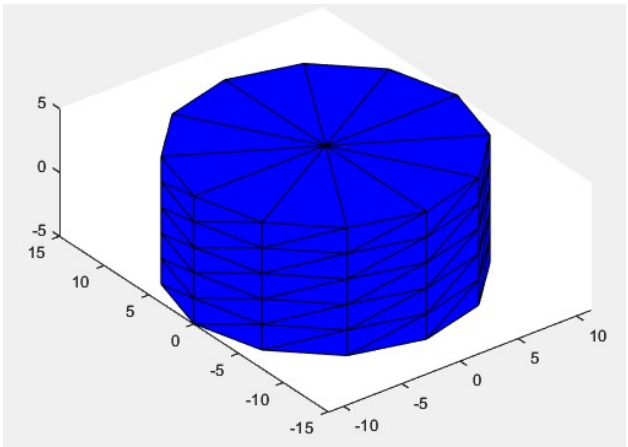

Figure 9. *Example of a cylinder of N=12 nodes divided into L=6 levels.*

The algorithm selects the IVUS points that belong to each level and slice based on z and θ as this data is in cylindrical coordinates (see Figure 10 and Figure 11**)**.

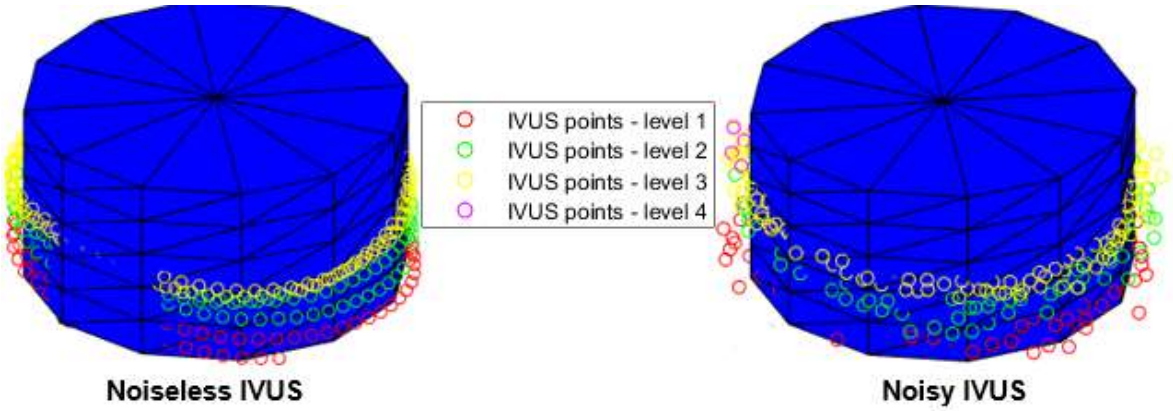

Figure 10. *IVUS points per level.*

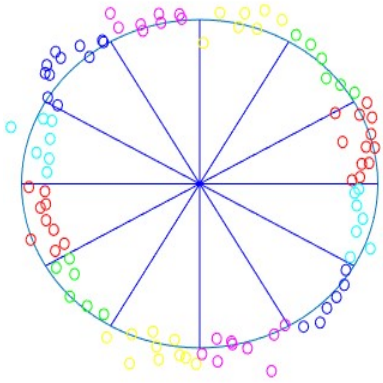

Figure 11. *IVUS points per slice in level 1 (noisy).*

If the IVUS point is between θ_1 and θ_2 and between z_1 and z_2, it belongs to slice 1 and level 1.

IVUS data is transformed from the frame {w} into the frame $\{cyl\}_k$ for each cylinder, prior to beginning this step. Only the IVUS data until the current timestep is considered.

2. Calculating the averaging distance between the triangle and points.

Each slice is formed by two triangles. Once the points that belong to each slice are identified, the distance from each of the identified IVUS points to the two triangles that form each slice is calculated. Following the assumption that a point will influence the triangle that is closest to it, the point is then assigned to the closest triangle. If the distance to both triangles is the same, the point is assigned to the bottom triangle (see Figure 12).

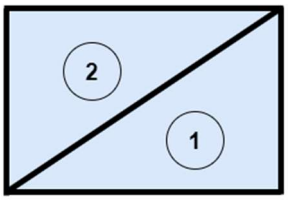

Figure 12. *Triangles in a face (1) Bottom triangle (2) Top triangle.*

The sign of the distance between a triangle and an IVUS point is determined by calculating the surface normal of the triangle and doing the dot product with the distance from IVUS point to the center of the triangle.

Once all the distances are calculated for one triangle, the outliers are removed since the point cloud can contain noise points. After that, the distance values that remain for each triangle are averaged and saved as average distance.

3. Estimating average distance for a triangle without any associated point.

For the triangles that do not have points associated, the travel distance is calculated as the mean of the travel distance of their adjacent triangles, i.e. the triangles they share an edge with (see Figure 13).

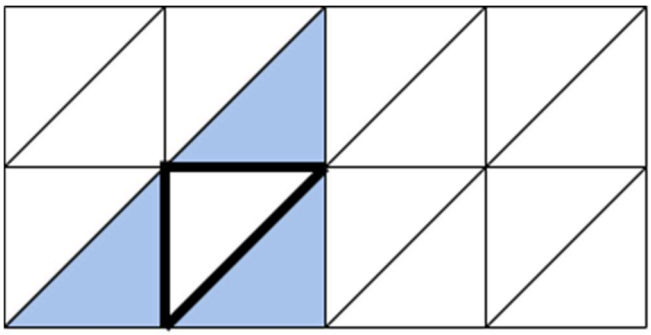

Figure 13. *Triangle without points associated and its adjacent (blue) triangles.*

4. Calculating the distance to move for each node.

The mean of the distances for each triangle is the average distance saved on the previous steps. It is necessary to determine the distance for each node to move in order to be able to modify the primitive shape. To achieve that, the respective triangles adjacent to each node of the cylinder are identified (see Figure 14) and the average distance values saved are averaged to determine the distance for the node to move, the travel distance.

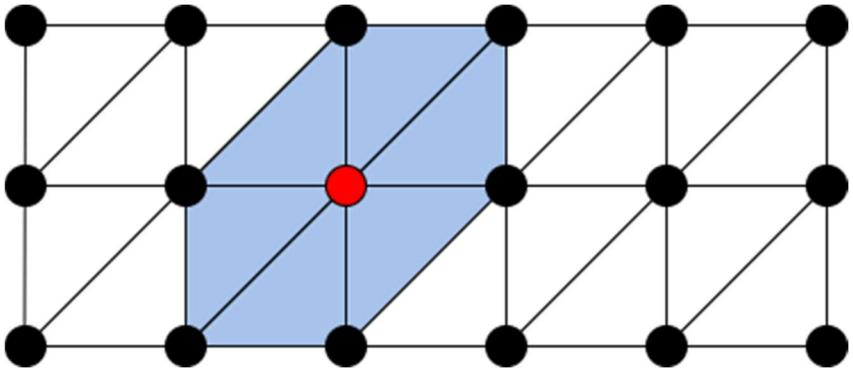

Figure 14. *Travel distance calculation of a node (red point) based on the averaged distance of the adjacent (blue) triangles.*

3.2.3. Moving nodes

The deformation of the cylinder is done in the radial direction. This way the nodes move along the line (θ) where they are located and the only thing that changes is the radius (see Figure 15).

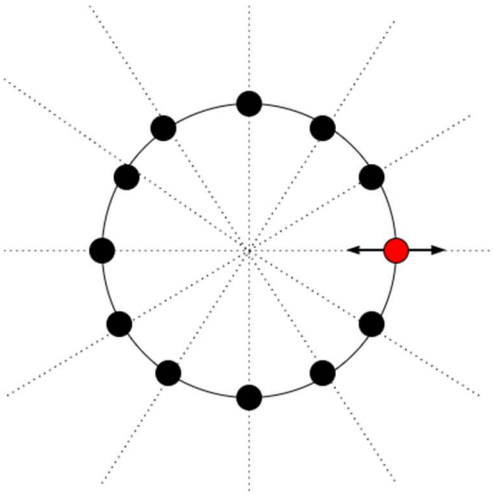

Figure 15. *Moving nodes*

Once the travel distance for each node is obtained, it is added to the radius of the node obtaining its new location.

The steps known as distance estimation and moving nodes are repeated three times.

3.2.4. Outliers

As mentioned in the second step of distance estimation, the point cloud can contain noise points. Therefore, the point cloud outliers are removed using the Box Plot method. A Box Plot is a visualization design that uses box shapes to display insights into data. It can also be used to pinpoint outliers and it is based on a six number summary:

- First quartile (Q1): the middle number between the smallest number and the median of the dataset.

$$q1 = \left(\frac{1}{4}\right) \cdot (n+1) \rightarrow Q1 = distances(q1)$$

- Median (Q2): the middle value of the dataset.

$$q2 = \left(\frac{1}{2}\right) \cdot (n+1) \rightarrow Q2 = distances(q2)$$

- Third Quartile (Q3): the middle value between the median and the highest value of the dataset.

$$q3 = \left(\frac{3}{4}\right) \cdot (n+1) \rightarrow Q3 = distances(q3)$$

- Interquartile Range (IQR): tells how spread the middle values are.

$$IQR = Q3 - Q1$$

- Lower limit or “minimum”

$$Lower\ limit = Q1 - 1{,}5 \cdot IQR$$

- Upper limit or “maximum”

$$Upper\ limit = Q3 + 1{,}5 \cdot IQR$$

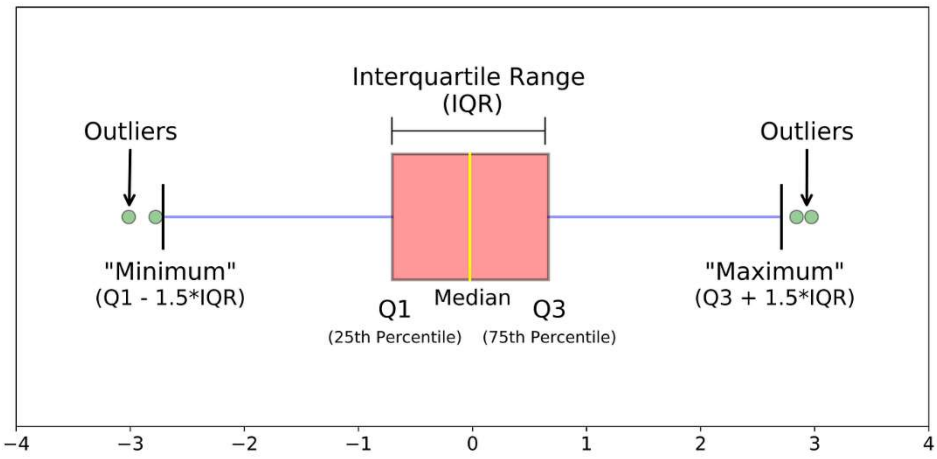

Figure 16. *Outliers detection with Boxplot [25].*

3.3. Merging method

The cylinder is converted into a mesh after undergoing the steps of the reconstruction phase. When the next cylinder is deformed, it needs to be merged with the mesh creating a new one. To do that, the next steps are followed:

1. Transformation of the mesh from its own mesh frame {mesh} into the $\{cyl\}_k$ using the transformation matrix ${}_{cyl}T^{mesh}$ (see Equation 9).

$$ {}_{w}T^{cyl} \rightarrow {}_{cyl}T^{w} = ({}_{w}T^{cyl})^{-1} $$

$$ {}_{cyl}T^{mesh} = {}_{cyl}T^{w} \bullet {}_{w}T^{mesh} $$

Equation 9. *Relation between the {cyl}, {mesh} and {w}.*

2. Identification of the non-overlapping points of the mesh with the cylinder. To do that, boundaries are defined by checking x, y and z so only mesh points that are not inside the cylinder are identified. The formulas used are the following:

$$ mesh_x < -\text{height}/2 $$

$$ mesh_y < \min(cyl_y) \;\&\; \max(cyl_y) < mesh_y $$

$$ mesh_z < \min(cyl_z) \;\&\; \max(cyl_z) < mesh_z $$

Equation 10. *Boundaries*

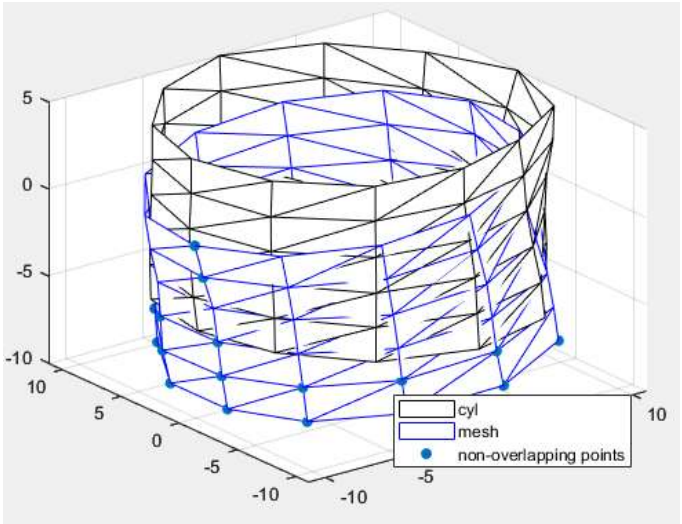

Figure 17. *Example of cylinder 60 and mesh 10.*

A condition to select the non-overlapping points is that the whole level of points must not overlap for that level of non-overlapping points to be added to the cylinder and create the new mesh (see Figure 18).

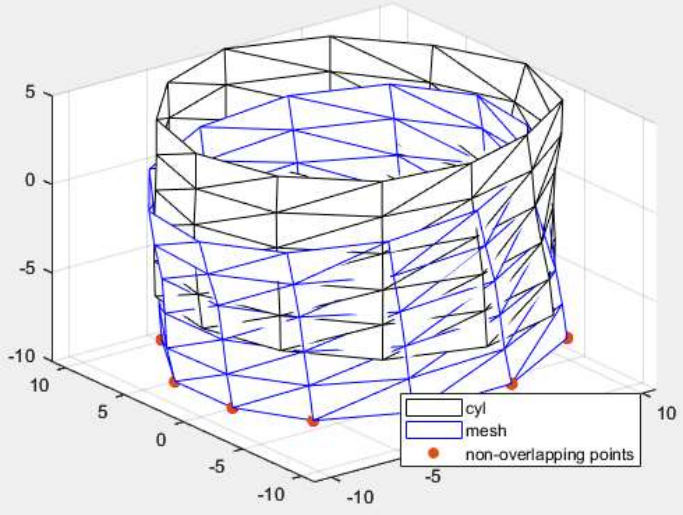

Figure 18. *Example of cylinder 60 and mesh 10 after applying the condition.*

3. Add non-overlapping mesh points to the new mesh. The non-overlapping mesh points are re-indexed prior to being added to the new mesh in order to have the nodes aligned and to be able to plot the new mesh without significantly altering the triangular connection matrix (TCM). To achieve that, the level of non-overlapping mesh points closest to the cylinder are selected, and the point closest to node 1 of the cylinder is determined. The non-overlapping points are then re-indexed, with the point nearest node 1 now being in position 1. Then, the non-overlapping points are added to the new mesh.
4. Expand its triangular connectivity matrix (TCM). The newly added mesh points to the new mesh are used to re-define the TCM by defining the relation of the new points. The new mesh can be seen in Figure 19.

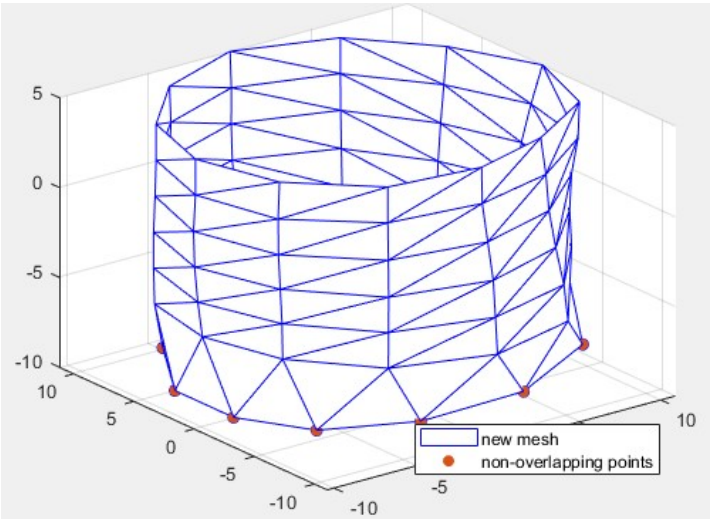

Figure 19. *New mesh and the non-overlapping selected cylinder points.*

4. Experimental validation

One way to measure the effectiveness of an algorithm is to validate an algorithm without relying on the original data that helped to develop the algorithm. The algorithm is tested using known geometries such as cylinder and a frustrum, and in-silico. A useful tool for evaluating the accuracy of a reconstruction method is to look at the fitting error with respect to a ground truth. In this chapter, the set-up of each validation is explained.

4.1. Higher radius cylinder

The algorithm is tested using a cylinder with a higher radius, 30mm, instead of using the IVUS points. The cylinder shares the same topology as the primitive shape. In this case, the ground truth is a 30 mm radius cylinder.

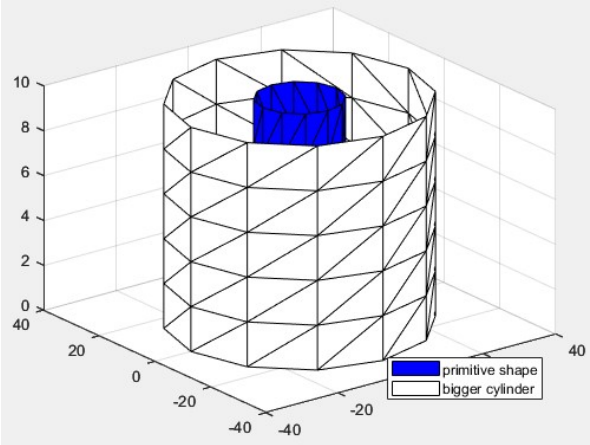

Figure 20. *Primitive shape and higher radius cylinder (mesh).*

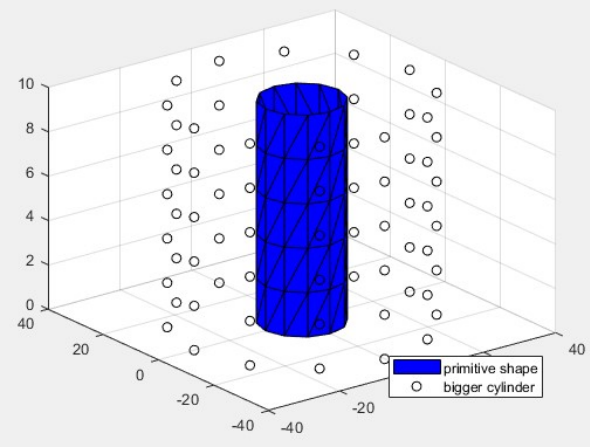

Figure 21. *Primitive shape and higher radius cylinder (scattered).*

4.2. Frustrum

The algorithm is tested using a frustrum instead of using the IVUS points. The frustrum also shares the same topology as the primitive shape. In this case, the ground truth is the frustrum.

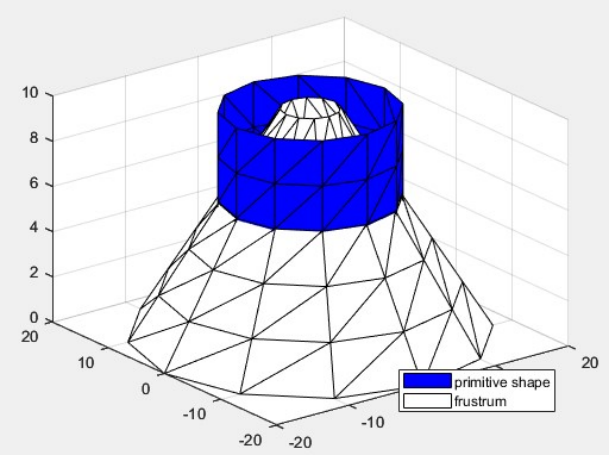

Figure 22. *Primitive shape and frustrum (mesh).*

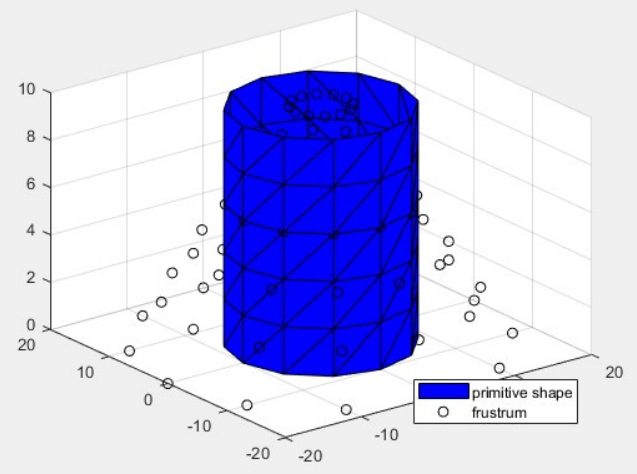

Figure 23. *Primitive shape and frustrum (scattered).*

4.3. In-silico

The validation is conducted by doing an in-silico experiment. The main components of the simulation environment are a patient-specific aortic model, synthetically generated EM and IVUS data, and a simulated virtual catheter. The patient-specific aortic model (see

Figure 24) was derived from segmented CT scans of an aorta of a patient. For the purpose of modeling EM noise for realistic synthetic data, Gaussian noise with zero mean and 0.3 mm and 0.5 mm standard deviations for the translational and rotational sections of the twist, respectively, was added. Similar to this, the coordinates of the synthetic IVUS data had Gaussian noise added to them with a zero mean and a standard deviation of 1 mm. The catheter was instructed to repeatedly follow a pre-defined trajectory along the vessel during catheter insertion or catheter bending during the in-silico tests. This pre-determined trajectory included a forward translation of 20 mm at a speed of 2.4 mm/s, as well as a sequence of bending instructions at a speed of 4.8 $^{\circ}/_{s}$, including a 30° bend of the catheter tip in a single bending plane (BP), a 360° bend of the BP, and a -30° bend of the catheter tip in the original BP [9]. In this case, the ground truth is a vessel mesh.

Figure 24. *Patient-specific aortic model*

Since the IVUS and the EM sensor are aligned points, it can be defined that the constant pose of the IVUS probe relative to the EM sensor, ${}_{e}T^{i}$, is equal to the identity matrix. Therefore, Equation 1 can be also defined as Equation 11.

$$ {}_{w}T^{i} = {}_{w}T^{e} \cdot {}_{e}T^{i} $$

$$ {}_{e}T^{i} = I \rightarrow {}_{\mathrm{w}}\mathrm{T}^{\mathrm{i}} = {}_{\mathrm{w}}\mathrm{T}^{\mathrm{e}} $$

Equation 11. *New relation between the {i}, {e} and {w}.*

5. Results & Discussion

The development of a real-time 3D reconstruction approach for a vessel has been the primary goal of this thesis. The method employs iterations to modify the mesh model in the best way possible to match the shape of the point cloud of IVUS data.

The results obtained by the experimental validation using known geometries such as cylinder and a frustrum, and in-silico will be discussed in this chapter.

5.1. Higher radius cylinder

The reconstruction step is performed three times, however as shown in Table 1, the primitive shape experiences its maximum deformation (see Figure 25) in the first iteration obtaining a very small error.

Iteration 0	Iteration 1	Iteration 2	Iteration 3
20,0039	$4{,}92 \cdot 10^{-16}$	$4{,}92 \cdot 10^{-16}$	$4{,}92 \cdot 10^{-16}$

Table 1. *Fitting error of the deformed cylinder with respect to the bigger cylinder.*

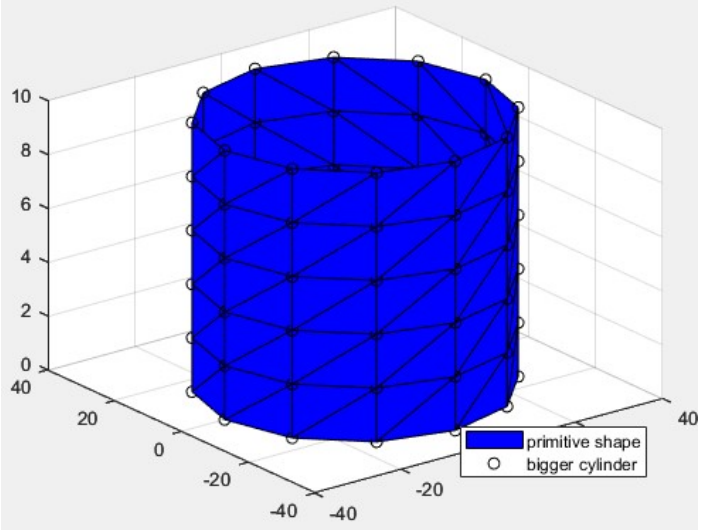

Figure 25. *Deformed primitive shape and bigger cylinder.*

5.2. Frustrum

Initially, the reconstruction step is performed three times, however as shown in Table 2 the fitting error decreases but it is not as small as the one obtained when using the bigger cylinder. As can be seen in Figure 26, the deformed cylinder does not completely match the frustrum as it does with the cylinder.

Iteration 0	Iteration 1	Iteration 2	Iteration 3
4,9406	0,5180	0.0869	0.0751

Table 2. *Fitting error of the deformed cylinder with respect to the frustrum – 3 iterations.*

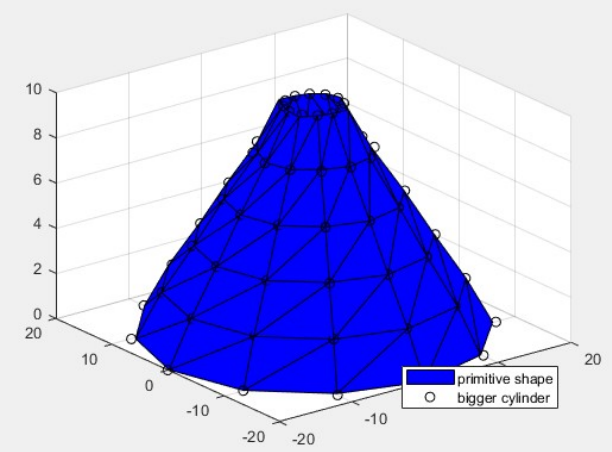

Figure 26. *Deformed primitive shape after 3 iterations and frustrum.*

As the results are not the expected ones, the reconstruction step is repeated ten times to study the deformation effect on the frustrum.

Iteration 0	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
4,9406	0,5180	0.0869	0.0751	0.0585	0.0465

Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
0.0390	0.0326	0.0283	0.0248	0.0224

Table 3. *Fitting error of the deformed cylinder with respect to the frustrum – 10 iterations.*

As more iterations are performed, the fitting error reduces as indicated in Table 3, but the error does not reach zero. Observing where the fitting error is higher in the nodes, it can be seen that is in the nodes that are on the top and bottom levels. This is as a result of the way the travel distance of each node is calculated.

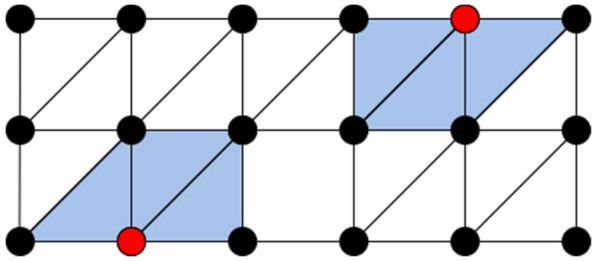

Figure 27. *Adjacent (blue) triangles of a node (red point) on the top and on the bottom level.*

As it can be seen in Figure 27, the nodes of the top and bottom levels have less triangles adjacent to the node. Therefore, as the nodes of the top and bottom levels have less data, the error does not reach zero. An improved deformation that matches better the frustrum is shown in Figure 28.

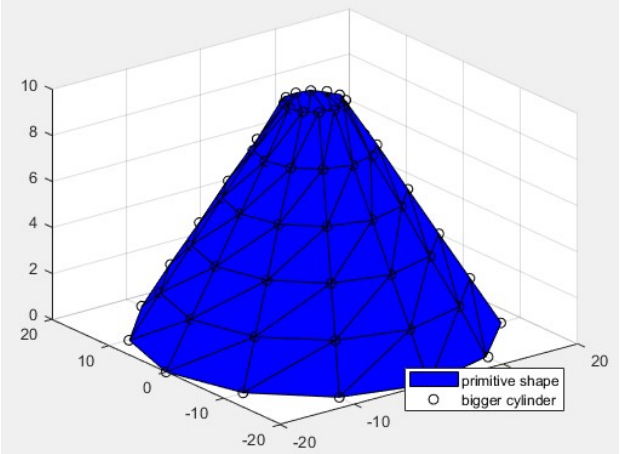

Figure 28. *Deformed primitive shape after 10 iterations and frustrum.*

5.3. In-silico

The algorithm was run using cylinder models with N=12 slices and L=6 levels. The method is split into two parts: the reconstruction phase, which deforms the primitive shape, and the merging phase, which combines the deformed primitive shape with the next deformed primitive shape. As the cylinders models are estimated at 12 frames per second, models from consecutive time steps show a large overlap. Therefore, only cylinders that are multiples of 10 are merged. The fitting error is calculated in both steps considering the vessel mesh as the ground truth.

- **Noisy and noiseless data**

The algorithm is run on synthetically generated IVUS data with added noise and without it. The fitting error of each one with respect to the vessel mesh is calculated to illustrate how the added noise affects the algorithm.

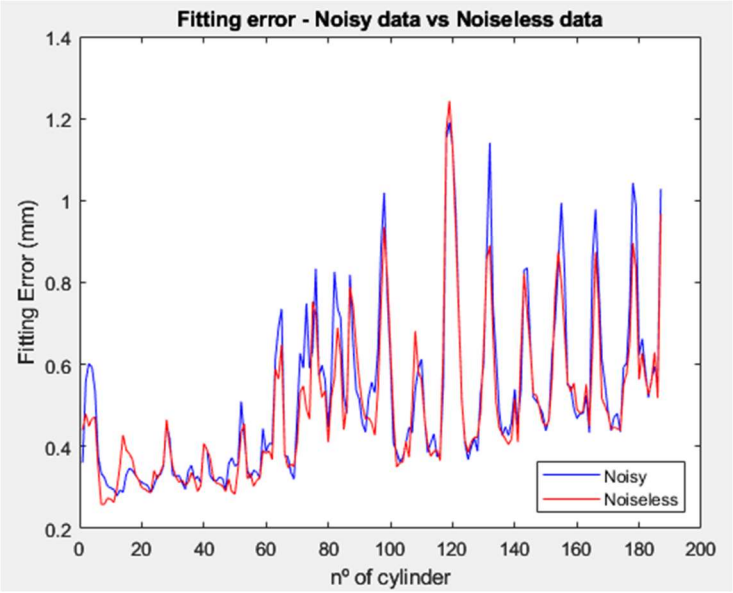

Figure 29. *Fitting error of each deformed cylinder not merged – Noisy vs Noiseless.*

On Figure 29, it can be observed that, with a few exceptions, the fitting error of the deformed cylinders produced from noisy and noiseless data follow the same pattern. Except for cylinder nº120, noisy and noiseless have peaks at the same cylinders, although usually the peaks from noisy data are greater. With cylinder nº120, both deformed cylinders have the same peak height and fitting error. This may be the result of an effective elimination of noisy outliers.

The cylinder has undergone three deformations in order to produce the final deformed cylinder. To determine if the pattern is the result of the deforming process, the fitting error in each iteration and its median have been computed.

Noisy	Iteration 0	0.6503	Noiseless	Iteration 0	0.6381
	Iteration 1	0.4774		Iteration 1	0.4716
	Iteration 2	0.4646		Iteration 2	0.4555
	Iteration 3	0.4722		Iteration 3	0.4572

Table 4. *Median of the Fitting errors of deformed cylinders obtained in iteration 0, 1, 2 and 3.*

The median in both columns of Table 4 follows the same pattern, decreasing from iteration 0 to iteration 2 and slightly increasing from iteration 2 to iteration 3. These findings lead to the conclusion that there is a clear relation between the environment and what is happening to the error; when the primitive shape is deformed based on noisy data, the error is bigger.

- **Comparation of fitting errors**

As the virtual catheter moves, more IVUS data are being gathered. The proposed method deforms the primitive shape to fit the shape of the IVUS points and merges the deformed cylinder with previous ones. Therefore, the fitting error has been calculated as new cylinders were being merged.

- Original cylinder vs Deformed cylinder

The algorithm uses as primitive shape a cylinder that was estimated in [9]. That primitive shape is mentioned as the original cylinder. To evaluate the accuracy of the method, the fitting error of the original cylinders merged, and the deformed cylinders merged are compared.

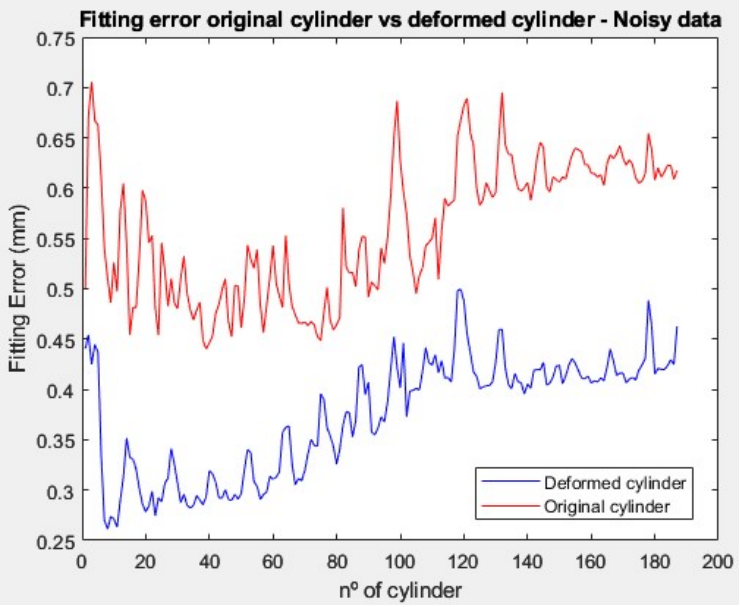

Figure 30. *Fitting error original vs deformed cylinder - Noisy data*

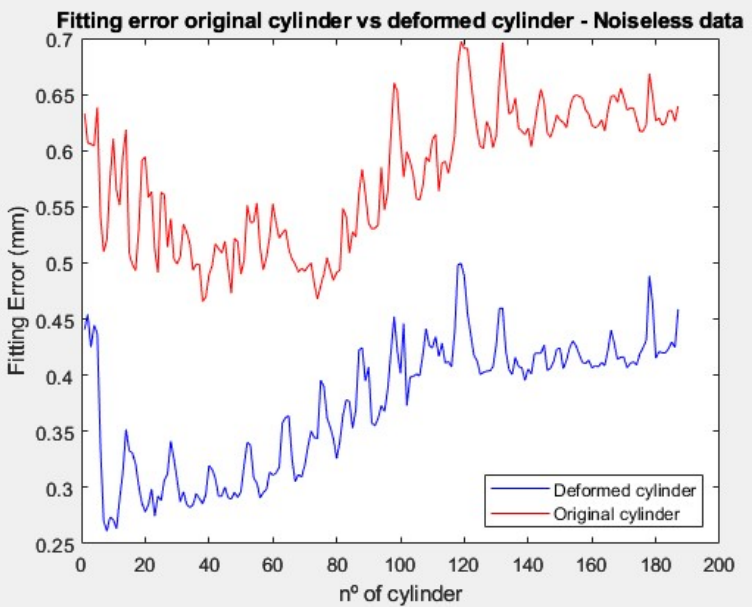

Figure 31. *Fitting error original vs deformed cylinders - Noiseless data*

As can be seen on the figures above, both with noiseless and noisy data, the deformed cylinders present a lower fitting error than the original cylinders. In both figures, constant peaks can be observed as the fitting error slowly increases in both cases but the fitting error from the original cylinder presents more higher peaks. These peaks can be related to the orientation of the primitive shape. As the original cylinders move and change their orientation, the fitting error increases. Additionally, it is apparent the total fitting error rises. This may be as a result of handling more data as the cylinders are combined.

- Deformed cylinder with noisy data vs noiseless data

To evaluate how the data affects the accuracy of the method, the fitting error of the deformed cylinders merged created from noisy and noiseless data are compared.

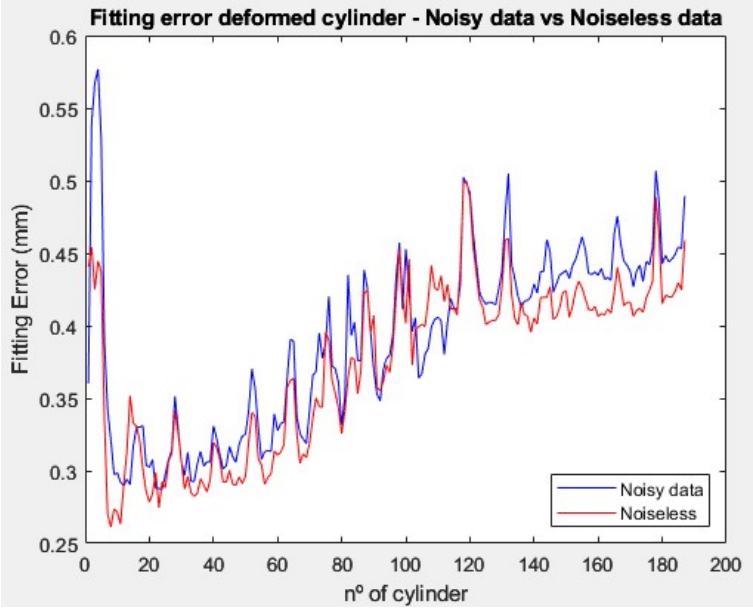

Figure 32. *Fitting error deformed cylinders – Noisy data vs Noiseless data*

Looking at the Figure 32, it can be seen that overall, the fitting error of the cylinders deformed is very small, especially for noisy data which the algorithm achieves a very comparable fitting error. As expected, the fitting error from noiseless data is smaller. These results support the theory that cylinders created from noiseless data have a lower fitting error as they are created from perfect data. However, from cylinder nº 102 to 116, the fitting error from the noiseless data is higher than the noisy data. Checking the algorithm reveals that it could be because there is less data when using noiseless data, critical data that would have been misconstrued for an outlier has been deleted. Comparing the Figure 32 and Figure 29, can be seen that in both figures the fitting error at nº 120 presents the same behavior, same fitting error in noisy and noiseless data.

- Fitting errors of merged cylinders vs not-merged cylinders

It would be false to assume that the average of the fitting errors of all the cylinders that have not been merged equals the fitting error of all the cylinders combined since some points are cut-off when merging cylinders. However, it is expected that both errors will have similar results (see Table 5).

	Average of all non-merged cylinders fitting errors	Fitting error of all the cylinders merged
Noisy data	0.5187	0.4894
Noiseless data	0.4960	0.4588

Table 5. *Comparison of merged cylinders vs non-merged cylinders.*

As it can be seen on Table 5, the fitting errors are very similar but not equal. The errors should not be too dissimilar because they both share the same topology.

- **Distribution of the data**

As observed in the aforementioned figures, the fitting error exhibits several peaks while gradually increasing. Two visual tools, a violin plot and an error colormap, have been employed to analyze the data distribution.

- Violin plot

This type of plot provides information about the distribution of the dataset and is considered a combination of box plot and kernel density plot. It can be found the same information as in the box plots: Median, IQR, the lower limit (1st quartile$-1.5 \cdot IQR$) and the upper limit (3rd quartile+$1.5 \cdot IQR$).

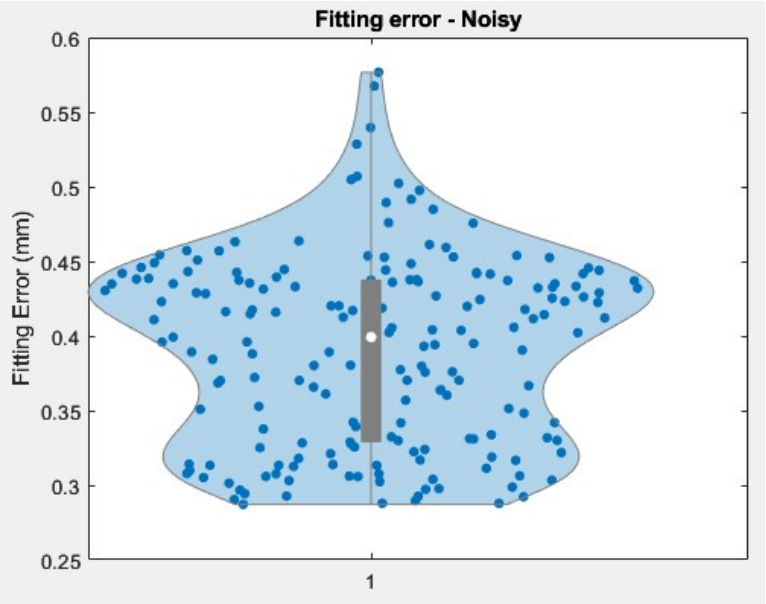

Figure 33. *Violin plot of the Fitting error - Noisy data*

Mean	IQR	Lower limit	Upper limit
0.3992	0.1085	0.1662	0.6001

Table 6. *Violin plot values - Noisy data*

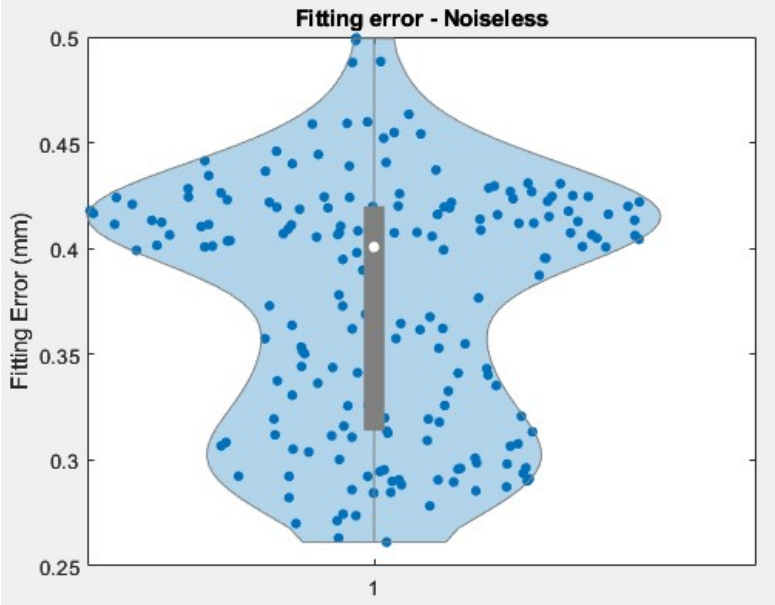

Figure 34. *Violin plot of the Fitting error - Noiseless data*

Mean	IQR	Lower limit	Upper limit
0.4007	0.1060	0.1547	0.5786

Table 7*. Violin plot values – Noiseless data*

On both figures (Figure 33 and Figure 34), the upper and lower limits do not appear on the plot. This is because, with some exceptions, the fitting errors are somewhat similar and concentrated in the same area with some exceptions. Figure 33 presents a more pointed end on the top part than Figure 34. This could be because noisy data have higher fitting error values since they contain more noise, as seen in Figure 33. The 3rd quartile, where there is a higher density of points, and the area around the 1st quartile are the places where most of the data is located. The mean lies in the middle of the IQR range representing the homogeneous distribution of the fitting errors. Figure 34 presents a more different shape, does not have any pointed end. At the 1st quartile is where there is the highest density of data, and the mean lies closer to the 1st quartile representing that the majority of the fitting errors values are closer to the 1st quartile value.

- Error colormap

An error representation with a colormap allows to see the distribution of the errors. In this case, the error values higher than 10 times the mean of the error have been replaced for that value in order to be able to differentiate in which areas the error is higher. The values of the fitting errors can be seen in the colorbar.

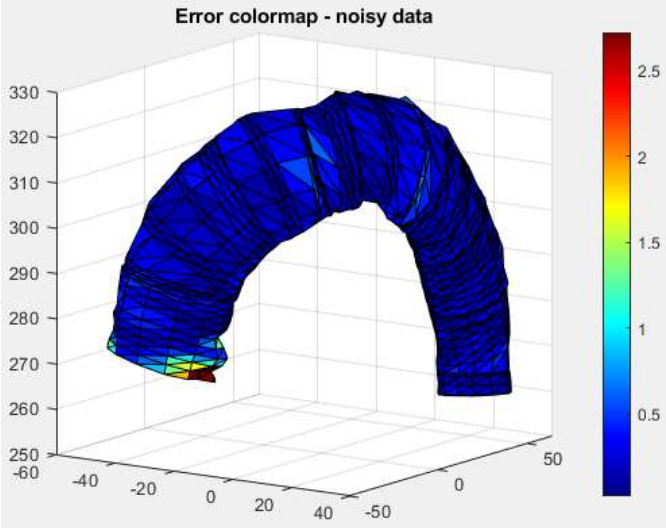

Figure 35. *Fitting error representation and distribution – Noisy data*

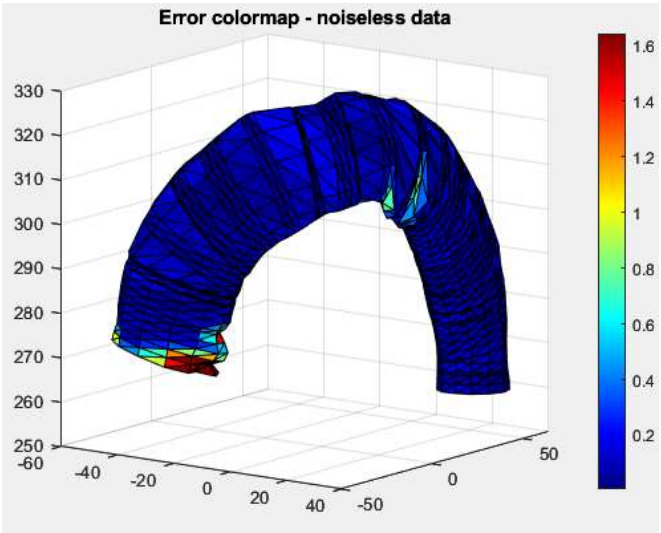

Figure 36. *Fitting error representation and distribution – Noiseless data*

Figure 35 and Figure 36 show that the fitting error is the same in all the mesh except near the final edge and the arch. This could be because the catheter is reaching the aortic root and because there is not enough data as portion of the cylinder model is ahead of the catheter tip where the sensor is located. The overall fitting error with noisy data is 0.2722 mm and with noiseless data is 0.1643 mm.

6. Limitations and future work

The algorithm proposed deforms the primitive shape three times in order to match the shape of the point cloud of IVUS data. Some points are going to be discussed to evaluate the efficacy of the algorithm, its limitations and future work. As the algorithm is divided into two parts: reconstruction and merging, the limitations are discussed separately.

- Reconstruction

In the reconstruction part, the primitive shape is created and is deformed to match the shape of the IVUS points that are around that cylinder. In this step, there have been some complications related to the coordinate system in which the calculations are done and the method to remove outliers.

Initially, the calculations were done in the cartesian coordinate system and the deformation in the xy direction. In order to optimize the code and improve its performance, it was considered more appropriate to work with cylindrical coordinates and deform the cylinder in the radial direction since the object being deformed is a cylinder.

The method [24] that served as the basis for the reconstruction method proposed in this thesis suggested an outlier method removal that consisted in doing the average of all the distances values and remove any distance values with a deviation of 30% or more. Even with perfect data, this method was removing important points because they were very clustered with the rest of the points. For this reason, the boxplot approach was used to remove outliers since it does not delete significant points and has no impact on how well the deformation process works.

Even though the algorithm has achieved the goal of the thesis that is to do a real-time vessel 3D reconstruction, there are some improvements that could be made. First, the primitive shape is created based on the algorithm described in [9]. The estimation done by Farola Barata et al. consider the base of the cylinder in the yz plane while the base of primitive shape created in this thesis is in the xy plane. This fact produces the excessive employment of transformation matrixes to rotate from xy plane to yz plane. After then, because the reconstruction is done locally, there are no longer any issues. Nevertheless, it will be preferable to define the base of the primitive shape as is defined in [9] for future work.

Second, the primitive shape geometry was chosen to be 5 levels and 12 nodes per level. The chosen mesh resolution produced good results, however if a higher resolution had been used, the reconstruction of the mesh would have been more accurate, but the algorithm would have required more time to run. As one of the goals of the algorithm is to be in real-time, that would not be beneficial. Future research should investigate the ideal mesh resolution to achieve the lowest fitting error but also enables the reconstruction to be in real-time.

Third, the commands used to create the primitive shape, generate the points of a cylinder including a point in the middle of the cylinder for each level. The calculations are defined considering those points and not using them. When the calculations and the reconstruction have been completed, a command removes the center points. If the point was deleted before doing any calculations or if the cylinder was built without those center points, this command might not be present.

Finally, the decision was made to iterate the cylinder deformation procedure three times because a small error was achieved. However, it would be interesting to study if a higher number of iterations will achieve a better deformation.

- Merging

Once the cylinders are deformed, the merging part starts. In the merging part, the non-overlapping parts of the mesh are identified and added to the new mesh. The mesh is formed by six levels of twelve points (see Figure 37).

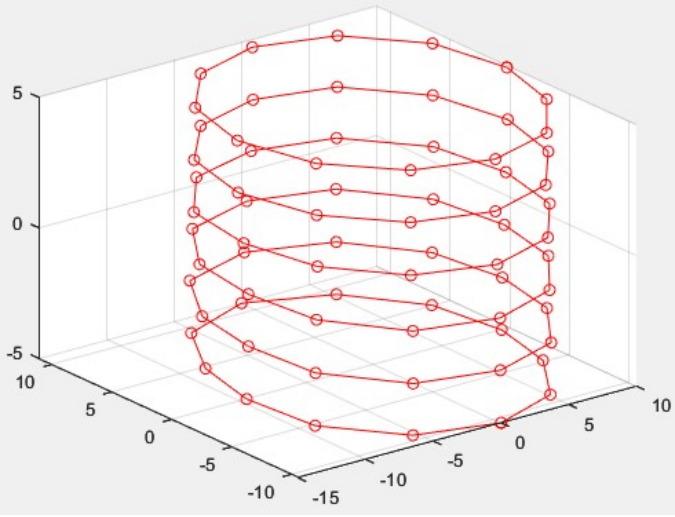

Figure 37. *Example of points of mesh with N=12 and L=5.*

The whole level of mesh points must not overlap for the non-overlapping points to be added to the new mesh. The entire level is not added to the new mesh if one of those mesh points overlaps. The new triangular connection matrix is defined when the non-overlapping mesh points have been chosen and included in the new mesh. This approach to merging is insufficient since some important points are being cut off. Finding a method that more naturally blends the mesh and the cylinder into the new mesh would be an improvement. If the vessel branches into smaller vessels, the deformation part will remain the same with minor modifications to how the IVUS points are obtained, but it will also call for a different merging technique.

Conclusions

Being aware of the surrounding environment when navigating a catheter within the vasculature is very useful for interventionists. The inherent restricted access to the anatomy of the patients severely complicates many endovascular procedures. To this end, an algorithm able do an intra-operative and real-time three-dimensional (3D) vessel reconstruction has been developed.

The data given to develop this algorithm consisted of IVUS points and EM posing data. Therefore, methods, based on IVUS data using EM pose for real-time 3D vessel reconstruction, have been researched. However, the method that inspired the algorithm is a method that works with specifically point cloud data. It was considered the most adequate method for this case as IVUS data is formed by points and the method also uses a primitive shape.

The algorithm is divided into two parts: the reconstruction and the merging. In the first one, it is obtained the 3D vessel reconstruction of a section of the vessel and in the second one, the different sections of 3D vessel reconstruction are combined.

The MATLAB command *trimesh* has been used to visualize the reconstruction, and the algorithm was tested in-silico and using well-known geometries like a cylinder and a frustrum. The fitting error between a mesh representing a genuine vessel and the reconstruction of the vessel was lastly computed to verify the effectiveness of the code. Small reconstruction errors of mean 0.4722 mm and 0.4572 mm were achieved for simulated IVUS and EM data with added noise and without it.

However, the in-vitro validation was skipped because of a lack of time. As there would be more noise and measurement errors, it would have been reasonable to anticipate slightly larger fitting errors than those achieved in-silico. Future work will involve doing in-vitro validation and updating the algorithm to make it possible to reconstruct when the vessel contains extra branches.

Acknowledgments

I would like to start by thanking *Katholieke Universiteit Leuven* (KU Leuven) and Prof. dr. Jos Vander Sloten and Prof. dr. Emmanuel Vander Poorten for the opportunity to work on this project. I would like to express my deepest appreciation to my supervisors at KU Leuven, Beatriz Farola Barata and Wim-Alexander Beckers, who have guided and advised me through the whole project.

Secondly, I would like to thank *Universitat Politècnica de Catalunya* (UPC), its professors and my classmates who have accompanied me during the master's degree in Neuroengineering and Rehabilitation.

Special thanks to *Hospital Universitari de Bellvitge* and my former coworkers, for making it possible to work and pursue a master's degree at the same time.

Sincere thanks also to my beautiful friends, for their unwavering love, support and for making the world a joyful and a more loving place.

Words cannot express my gratitude for my family, their unconditional love, daily encouragement and for the trust they have in me.

Finally, I would like to thank God. To Him be the glory and honor and power.

Bibliography

[1] World Health Organization, “Cardiovascular diseases (CVDs),” 2021. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds) (accessed Jan. 12, 2023).

[2] H. Rafii-Tari, C. J. Payne, and G. Z. Yang, “Current and emerging robot-assisted endovascular catheterization technologies: A review,” *Annals of Biomedical Engineering*, vol. 42, no. 4. Kluwer Academic Publishers, pp. 697–715, 2014. doi: 10.1007/s10439-013-0946-8.

[3] University of California San Francisco, “Endovascular Surgery | Conditions & Treatments | UCSF Health.” https://www.ucsfhealth.org/treatments/endovascular-surgery (accessed Jan. 12, 2023).

[4] Froedtert & Medical College of Wisconsin, “Endovascular Therapy, Catheterization Procedures.” https://www.froedtert.com/heart-care/treatment/endovascular-therapy (accessed Oct. 13, 2022).

[5] L. Da, D. Zhang, and T. Wang, “Overview of the vascular interventional robot,” *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 4, no. 4, pp. 289–294, Dec. 2008, doi: 10.1002/RCS.212.

[6] J. Bonatti, G. Vetrovec, C. Riga, O. Wazni, and P. Stadler, “Robotic technology in cardiovascular medicine,” *Nature Reviews Cardiology*, vol. 11, no. 5. Nature Publishing Group, pp. 266–275, 2014. doi: 10.1038/nrcardio.2014.23.

[7] P. T. Tran, “Catheter and Vessel Shape Estimation for Guidance of Robotic Catheters in Endovascular Surgery,” *Schatting van de katheter- en bloedvatvorm voor begeleiding van robotische katheters in endovasculaire chirurgie*, no. March, 2018.

[8] S. James Chen and J. D. Carroll, “3-D reconstruction of coronary arterial tree to optimize angiographic visualization,” *IEEE Trans Med Imaging*, vol. 19, no. 4, pp. 318–336, 2000, doi: 10.1109/42.848183.

[9] B. Farola Barata *et al.*, “IVUS-Based Local Vessel Estimation for Robotic Intravascular Navigation,” *IEEE Robot Autom Lett*, vol. 6, no. 4, pp. 8102–8109, 2021, doi: 10.1109/lra.2021.3102307.

[10] F. Bellocchio, N. A. Borghese, S. Ferrari, and V. Piuri, “3D Surface Reconstruction,” *3D Surface Reconstruction*, 2013, doi: 10.1007/978-1-4614-5632-2.

[11] G. Hichem, F. Chouchene, and H. Belmabrouk, “3D Model Reconstruction of Blood Vessels in The Retina with Tubular Structure,” *International Journal on Electrical Engineering and Informatics*, vol. 7, no. 4, 2015, doi: 10.15676/ijeei.2015.7.4.14.

[12] P. T. Tran, P. Chang, J. vander Sloten, and D. Stoyanov, “3D Catheter Shape Reconstruction Using Electromagnetic and Image Sensors 3D Catheter Shape Reconstruction using Electromagnetic and Image Sensors,” no. March, 2017, doi: 10.1142/S2424905X17400098.

[13] C. Shi *et al.*, “Real-time in vitro intravascular reconstruction and navigation for endovascular aortic stent grafting,” *Int J Med Robot*, vol. 12, no. 4, pp. 648–657, Dec. 2016, doi: 10.1002/RCS.1736.

[14] L. Zhao, S. Giannarou, S.-L. Lee, R. Merrifield, and G. Z. Yang, “Intra-operative Simultaneous Catheter and Environment Modelling for Endovascular Navigation Based on Intravascular Ultrasound, Electromagnetic Tracking and Pre-operative Data”.

[15] L. Zhao, S. Giannarou, S. L. Lee, and G. Z. Yang, “SCEM+: Real-Time Robust Simultaneous Catheter and Environment Modeling for Endovascular Navigation,” *Intravascular Ultrasound: From Acquisition to Advanced Quantitative Analysis*, vol. 1, no. 2, pp. 185–197, 2020, doi: 10.1016/B978-0-12-818833-0.00011-4.

[16] L. Zhao, S. Giannarou, S. L. Lee, and G. Z. Yang, “Registration-free simultaneous catheter and environment modelling,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9900 LNCS, pp. 525–533, 2016, doi: 10.1007/978-3-319-46720-7_61/TABLES/1.

[17] Y. Tang *et al.*, “Vision-Based Three-Dimensional Reconstruction and Monitoring of Large-Scale Steel Tubular Structures,” *Advances in Civil Engineering*, vol. 2020, 2020, doi: 10.1155/2020/1236021.

[18] S. Li, Y. Liu, and W. Su, “Single-image human mesh reconstruction by parallel spatial feature aggregation,” *Comput Animat Virtual Worlds*, vol. 33, no. 3–4, pp. 1–11, 2022, doi: 10.1002/cav.2084.

[19] F. Sun, “Surface reconstruction of the coronary arterial walls from intravascular ultrasound images,” *J Electron Imaging*, vol. 16, no. 4, p. 043016, 2007, doi: 10.1117/1.2804172.

[20] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, “A comparison and evaluation of multi-view stereo reconstruction algorithms,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 519–526. doi: 10.1109/CVPR.2006.19.

[21] J. L. Schonberger and J. M. Frahm, “Structure-from-Motion Revisited,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 4104–4113, Dec. 2016, doi: 10.1109/CVPR.2016.445.

[22] S. Kim, H. G. Kim, and T. Kim, “Mesh modelling of 3D point cloud from UAV images by point classification and geometric constraints,” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 42, no. 2, pp. 507–511, 2018, doi: 10.5194/isprs-archives-XLII-2-507-2018.

[23] X. J. Qin, Z. T. Hu, H. B. Zheng, and M. Y. Zhang, “Surface reconstruction from unorganized point clouds based on edge growing,” *Adv Manuf*, vol. 7, no. 3, pp. 343–352, 2019, doi: 10.1007/s40436-019-00262-5.

[24] K. Kwon and D. Mun, “Iterative offset-based method for reconstructing a mesh model from the point cloud of a pig,” *Comput Electron Agric*, vol. 198, no. April, 2022, doi: 10.1016/j.compag.2022.106996.

[25] “Outlier detection with Boxplots. In descriptive statistics, a box plot… | by Vishal Agarwal | Medium.” https://medium.com/@agarwal.vishal819/outlier-detection-with-boxplots-1b6757fafa21 (accessed Apr. 13, 2023).

Annex - Algorithm

```
clc; clear all;
% load cyl&rotation_noiselessdata.mat
% load cyl&rotation_noisedata.mat

for i=10:10:length(data_cyl)
   C_w=[]; I_w=[];
   for j=1:i
      w_Te=[rotm(:,:,j);[0 0 0]]; w_Te(:,4)=[Ep(j,:),1];
      IVUS=[x_data(j,:);y_data(j,:);zeros(1,M_cyl);ones(1,M_cyl)]; I_w=[I_w w_Te*IVUS];
      cyl_pos=[cyl_px(j,:) cyl_py(j,:) 0 1]; cyl_pos=cyl_pos';
      C_w=[C_w w_Te*cyl_pos];
   end

   % Transformation of all IVUS data for one cylinder into the cyl frame from the world frame
   w_Te=[rotm(:,:,i);[0 0 0]]; w_Te(:,4)=[Ep(i,:),1];
   e_Tcyl=[cos(theta(i,:)), -sin(theta(i,:)),0,cyl_px(i,:);sin(theta(i,:))*cos(phi(i,:)),
cos(theta(i,:))*cos(phi(i,:)),-sin(phi(i,:)),cyl_py(i,:);
   sin(theta(i,:))*sin(phi(i,:)), cos(theta(i,:))*sin(phi(i,:)),cos(phi(i,:)),0; [0,0,0,1]];
   w_Tcyl=w_Te*e_Tcyl; cyl_Tw=inv(w_Tcyl);
   % Transformation & rotation into the cylinder frame
   alpha=-pi/2; Ry=[cos(alpha) 0 sin(alpha) 0;0 1 0 0; -sin(alpha) 0 cos(alpha) 0; 0 0 0 1];
   tm=Ry*cyl_Tw;
   I_cyl=tm*I_w;I_cyl=I_cyl([1:3],:);

   % CYLINDER MAKING
   height=10;
   levels=5;
   Nr=1; Nt=12; zz=0:(height/levels):height;
   [Nodes, Triangles, Quads]=Circle_Mesh(rad(i),Nr,Nt);
   [Nodes3D,Prisms,Bricks] = Mesh2D_to_Mesh3D(Nodes,Triangles,Quads,zz);

   % TRANSFORMATION INTO THE CENTER OF THE COORDINATE FRAME
   t=[0; 0; -max(zz)/2]; TF_c=[1 0 0; 0 1 0; 0 0 1; 0 0 0];TF_c(:,4)=[t; 1];
   N=[Nodes3D(:,1),Nodes3D(:,2),Nodes3D(:,3),ones(length(Nodes3D),1)];N=N';
   Nodes3D=TF_c*N;Nodes3D(4,:)=[]; Nodes3D=Nodes3D';
   cyl_mesh=Nodes3D';
```

```
%TRIANGLE CONNECTIVITY MATRIX (my cylinder)
cyl_TCM=[];
cyl_TCM=[cyl_TCM ; [Prisms(:,1) Prisms(:,2) Prisms(:,5)]];
cyl_TCM=[cyl_TCM ; [Prisms(:,1) Prisms(:,5) Prisms(:,4)]];

% DEFINE SECTION OF IVUS DATA INSIDE THE CYLINDER IN Z AXIS
Cz=cyl_mesh(3,:);
I_section_cyl=[];
for nI=1:length(I_cyl)
   if I_cyl(3,nI)>=min(Cz) & I_cyl(3,nI)<=max(Cz)
      I_section_cyl=[I_section_cyl I_cyl(:,nI)];
   end
end

% TRANSFORM IVUS in {cyl} INTO THE CYLINDRICAL COORDINATES
I_cyl_CC=[];
for cc=1:length(I_section_cyl)
   angle=atan(I_section_cyl(2,cc)/I_section_cyl(1,cc));
   if I_section_cyl(1,cc)<0
      angle=angle+pi;
   else
      if angle<0
         angle=angle+2*pi;
      end
   end
   I_cyl_CC(:,cc)=[sqrt(I_section_cyl(1,cc)^2+I_section_cyl(2,cc)^2) angle I_section_cyl(3,cc)];
end

% TRANSFORM CYL NODES INTO THE CYLINDRICAL COORDINATES [x y z] --> [r angle z]
cyl_mesh_CC=[];
for cc=1:length(cyl_mesh)
   angle=atan(cyl_mesh(2,cc)/cyl_mesh(1,cc));
   if cyl_mesh(1,cc)<0
      angle=angle+pi;
   else
      if angle<0
         angle=angle+2*pi;
```

```
        end
    end
    cyl_mesh_CC(:,cc)=[sqrt(cyl_mesh(1,cc)^2+cyl_mesh(2,cc)^2) angle cyl_mesh(3,cc)];
end
% DEFINE SECTION OF IVUS DATA AROUND THE CYLINDER in the xyplane
l_section_CC=[];index_ivus=[];
for nl=1:length(l_cyl_CC)
    if rad(i)*1.5>=l_cyl_CC(1,nl)  &&  l_cyl_CC(1,nl)>= rad(i)*0.5
        l_section_CC=[l_section_CC l_cyl_CC(:,nl)];
        index_ivus=[index_ivus nl] ;
    end
end

% SANITY CHECK - IVUS & CYL NODES IN CARTESSIAN COORDINATES SO WE CAN SEE
IF THEY ARE PLOTTED CORRECTLY
for hc=1:length(index_ivus)
    l_section_HC(:,hc)=[l_section_cyl(:,index_ivus(hc))];
end

% DISTANCE CALCULATION USING CYLINDRICAL COORDINATES
angles=[]; Dist_lev=[];cyl_nodes=[];Iteration_nodes=[];
for iteration=1:4 %iteration 1 es iteration 0
    length_d=[]; l_d=[];
    if iteration~=1

        DD=[];
        if iteration==2
            for n=1:Nt
                angles(n,:)=[(2*pi/Nt)*(n-1)];
            end

            angles(length(angles)+1)=2*pi;
            cylinder=cyl_mesh_CC';
        else
            cylinder=cyl_all;
            cyl_nodes=[];
        end
        z_limit=unique(cylinder(:,3)); %grid in z
```

```
for lev=1:levels
    for l=1:(length(angles)-1)
        l_area1=[];l_area2=[];Distance1=[];Distance2=[];
        for k=1:length(l_section_CC)
            if z_limit(lev)<=l_section_CC(3,k) & l_section_CC(3,k)<=z_limit(lev+1)
                if angles(l)<=l_section_CC(2,k) & l_section_CC(2,k)<=angles(l+1)
                    n_tri1=[Prisms(Nt*(lev-1)+l,1) Prisms(Nt*(lev-1)+l,2) Prisms(Nt*(lev-1)+l,5)];
                    n_tri2=[Prisms(Nt*(lev-1)+l,1) Prisms(Nt*(lev-1)+l,5) Prisms(Nt*(lev-1)+l,4)];
                    TRI1=[cylinder(n_tri1(1),:);cylinder(n_tri1(2),:);cylinder(n_tri1(3),:)];
                    TRI2=[cylinder(n_tri2(1),:);cylinder(n_tri2(2),:);cylinder(n_tri2(3),:)];
                    if l==Nt
                        TRI1(2,2)=angles(Nt+1);TRI1(3,2)=angles(Nt+1);
                        TRI2(2,2)=angles(Nt+1);
                    end
                    [dist1,pp1]=pointTriangleDistance(TRI1,l_section_CC(:,k));
                    [dist2,pp2]=pointTriangleDistance(TRI2,l_section_CC(:,k));
                    if dist1>dist2
                        B=faceNormal(triangulation([1,2,3],TRI2(:,1),TRI2(:,2),TRI2(:,3)));
                        A=l_section_CC(:,k)'-pp2;
                        C=dot(A,B);
                        if C<0
                            dist2=-dist2;
                        end
                        l_area2=[l_area2 l_section_HC(:,k)];
                        Distance2=[Distance2 dist2];
                    else
                        B=faceNormal(triangulation([1,2,3],TRI1(:,1),TRI1(:,2),TRI1(:,3)));
                        A=l_section_CC(:,k)'-pp1;
                        C=dot(A,B);
                        if C<0
                            dist1=-dist1;
                        end
                        l_area1=[l_area1 l_section_HC(:,k)];
                        Distance1=[Distance1 dist1];
                    end
                end

            end
```

```
end

%remove noise
%PLOTBOX OUTLIERS REMOVAL
D1=[];
Distance1=sort(Distance1);n1=length(Distance1);
if n1==1 || n1==0
   D1=Distance1;
else
   q2=round((1/2)*(n1+1));
   q1=round((1/4)*(n1+1));
   q3=round((3/4)*(n1+1));
   iqr1=Distance1(q3)-Distance1(q1); low_lim1=Distance1(q1)-1.5*iqr1;
   up_lim1=Distance1(q3)+1.5*iqr1;
   D1=[];
   for d=1:length(Distance1)
      if low_lim1<Distance1(d) && Distance1(d)<up_lim1
         D1=[D1 Distance1(d)];
      end
   end
end

D2=[];
Distance2=sort(Distance2);n2=length(Distance2);
if n2==1 || n2==0
   D2=Distance2;
else
   q2=round((1/2)*(n2+1));
   q1=round((1/4)*(n2+1));
   q3=round((3/4)*(n2+1));
   iqr2=Distance2(q3)-Distance2(q1); low_lim2=Distance2(q1)-1.5*iqr2;
   up_lim2=Distance2(q3)+1.5*iqr2;
   D2=[];
   for d=1:length(Distance2)
      if low_lim2<Distance2(d) && Distance2(d)<up_lim2
         D2=[D2 Distance2(d)];
      end
```

```
            end
        end
        DD(:,l)=[mean(D1); mean(D2)];
        l_d(:,l)=[length(Distance1); length(Distance2); length(D1); length(D2)];
    end
    DD(isnan(DD))=0;
    Dist_lev(:,:,lev)=DD;
    length_d(:,:,lev)=l_d;

    %TRIANGLES WITH NO POINT ASSOCIATED (except for the
    %upper triangles of level 5)
    for dk=1:length(Dist_lev(:,:,lev))
        if lev==1 && Dist_lev(1,dk,1)==0
            if dk==1
                Dist_lev(1,dk,lev)=mean([Dist_lev(2,Prisms(dk,1),lev)
Dist_lev(2,Prisms(dk,2),lev)]);
            else
                Dist_lev(1,dk,lev)=mean([Dist_lev(2,Prisms(dk-1,2),lev) Dist_lev(2,Prisms(dk-
1,1),lev)]);
            end
        elseif lev~=1
            for h=1:2
                if h==1 && Dist_lev(h,dk,lev)==0
                    Dist_lev(1,dk,lev)=mean([Dist_lev(2,Prisms(dk,1),lev)
Dist_lev(2,Prisms(dk,2),lev) Dist_lev(2,Prisms(dk,1),lev-1)]);
                elseif h==2 && Dist_lev(h,dk,lev-1)==0 %nunca pasara para h=1,solo para h=2
                    if dk==1
                        Dist_lev(2,dk,lev-1)=mean([Dist_lev(1,Prisms(Nt,1),lev-1)
Dist_lev(1,Prisms(Nt,2),lev-1) Dist_lev(1,Prisms(Nt,2),lev)]);
                    else
                        Dist_lev(2,dk,lev-1)=mean([Dist_lev(1,Prisms(dk-1,2),lev-1)
Dist_lev(1,Prisms(dk-1,1),lev-1) Dist_lev(1,Prisms(dk-1,2),lev)]);
                    end
                end
            end
        end
    end
```

```
%TRIANGLES WITH NO POINTS FORM UPPER PARTS OF LEVEL 5
        if lev==5
            for dk=1:length(Dist_lev(:,:,lev))
                if Dist_lev(2,dk,lev)==0
                    if dk==1
                        Dist_lev(2,dk,lev)=mean([Dist_lev(1,Prisms(Nt,1),lev)
Dist_lev(1,Prisms(Nt,2),lev)]);
                    else
                        Dist_lev(2,dk,lev)=mean([Dist_lev(1,Prisms(dk-1,1),lev) Dist_lev(1,Prisms(dk-
1,2),lev)]);
                    end
                end
            end
        end
    end

    for lev=1:levels+1 %added 1 so it would calculate the new_nodes for the last line
        %travel distance
        travel_d=[];
        for td=1:length(DD)
            if lev==1
                mov_d=[Dist_lev(1,Prisms(td,1),lev) Dist_lev(2,Prisms(td,2),lev)
Dist_lev(1,Prisms(td,2),lev)];
            elseif lev==(max(levels)+1)
                mov_d=[Dist_lev(2,Prisms(td,1),(lev-1)) Dist_lev(1,Prisms(td,1),(lev-1))
Dist_lev(2,Prisms(td,2),(lev-1))];
            else
                mov_d=[Dist_lev(2,Prisms(td,1),(lev-1)) Dist_lev(1,Prisms(td,1),(lev-1))
Dist_lev(2,Prisms(td,2),(lev-1)) Dist_lev(1,Prisms(td,1),lev) Dist_lev(2,Prisms(td,2),lev)
Dist_lev(1,Prisms(td,2),lev)];
            end
            travel_d(td)=[mean(mov_d)];
        end
        travel_d=[travel_d(:,Nt) travel_d(:,1:(Nt-1))]; %node1 in the first place

        %Z-coordinate without the center point
        Z_coord=cyl_mesh_CC(3,[(1+(lev-1)*(Nt+1)):(Nt+(lev-1)*(Nt+1))]);
```

```
            %new_nodes first line
            New_nodes=[];
            for nw=1:length(travel_d)
                new_radius=cylinder(nw+(lev-1)*(Nt+1),1)+travel_d(nw);
                New_nodes=[New_nodes; new_radius angles(nw) Z_coord(nw)];
            end
            %add center point
            New_nodes(13,:)=cyl_mesh_CC(:,Nt+1+(Nt+1)*(lev-1));
            cyl_nodes(:,:,lev)=New_nodes;

        end
        cyl_all=[];
        for lev=1:levels+1
            cyl_all=[cyl_all; cyl_nodes(:,:,lev)];
        end

        cyl_it_CARTESIAN=[];
        for lev=1:6
            for j=1:length(cyl_nodes(:,1,1))
                cyl_it(j,:)=[cyl_nodes(j,1,lev)*cos(cyl_nodes(j,2,lev))
cyl_nodes(j,1,lev)*sin(cyl_nodes(j,2,lev)) cyl_nodes(j,3,lev)];
            end
            cyl_it(13,:)=cyl_mesh(:,Nt+1+(Nt+1)*(lev-1));
            cyl_it_CARTESIAN=[cyl_it_CARTESIAN; cyl_it];
        end
    end

    if iteration==1
        vertices=cyl_mesh';
    else
        vertices=cyl_it_CARTESIAN;
    end

    %remove centerpoint
    vertices([(Nt+1),2*(Nt+1),3*(Nt+1),4*(Nt+1),5*(Nt+1),6*(Nt+1)],:)=[];
    new_cyl_TCM=cyl_TCM;

    for c=1:length(new_cyl_TCM)
```

```
        for column=1:3
            if 14<=new_cyl_TCM(c,column) && new_cyl_TCM(c,column)<=25
                new_cyl_TCM(c,column)=new_cyl_TCM(c,column)-1;
            elseif 27<=new_cyl_TCM(c,column) && new_cyl_TCM(c,column)<=38
                new_cyl_TCM(c,column)=new_cyl_TCM(c,column)-2;
            elseif 40<=new_cyl_TCM(c,column) && new_cyl_TCM(c,column)<=51
                new_cyl_TCM(c,column)=new_cyl_TCM(c,column)-3;
            elseif 53<=new_cyl_TCM(c,column) && new_cyl_TCM(c,column)<=64
                new_cyl_TCM(c,column)=new_cyl_TCM(c,column)-4;
            elseif 66<=new_cyl_TCM(c,column) && new_cyl_TCM(c,column)<=77
                new_cyl_TCM(c,column)=new_cyl_TCM(c,column)-5;
            end
        end
    end


    for nc=1:length(new_cyl_TCM)
        for column=1:3
            if 73<=new_cyl_TCM(c,column)
                new_cyl_TCM(c,column)=[];
            end
        end
    end

  end % iteration ends
  Mesh(:,:,i)=vertices;
  TM(:,:,i)=w_Tcyl;
  TCM(:,:,i)=new_cyl_TCM;
end % i ends

l_mesh=[];
list=10:10:length(Mesh)-10;
for mm=1:length(list)
  m=list(mm);
  if mm==1
    m1=m;
    mesh1=Mesh(:,:,m1);
    w_Tcylm1=TM(:,:,m1);
```

```
    tcm_simp=TCM(:,:,m1);

    TCM1=[];
    %sort tcm1
    a_tcm=length(tcm_simp)/2;
    for nv=1:a_tcm/Nt
        ts=[tcm_simp([12*nv-11:nv*Nt],:);tcm_simp([(Nt*(nv-1)+a_tcm+1):(Nt*nv+a_tcm)],:)];
        TCM1=[TCM1; ts]
    end
else
    mesh1=merg_mesh;
    w_Tcylm1=w_Tcylm2;
    TCM1=merg_mesh_TCM;
end

m2=m+10;
mesh2=Mesh(:,:,m2); w_Tcylm2=TM(:,:,m2); TCM2=TCM(:,:,m2);
% Rotation and transformation - mesh1
alpha=pi/2; Ry=[cos(alpha) 0 sin(alpha) 0;0 1 0 0; -sin(alpha) 0 cos(alpha) 0; 0 0 0 1];
cylm2_Tw=inv(w_Tcylm2); cylm2_Tcylm1=cylm2_Tw*w_Tcylm1;
if mm==1
    tm=cylm2_Tcylm1*Ry;
else
    tm=cylm2_Tcylm1;
end
mesh1=[mesh1 ones(length(mesh1),1)]; mesh1_m2=tm*mesh1';
mesh1_m2=mesh1_m2([1:3],:)';
%Rotation - mesh2 (cyl)
alpha=pi/2; Ry_m2=[cos(alpha) 0 sin(alpha);0 1 0; -sin(alpha) 0 cos(alpha)];
mesh2_r=(Ry_m2*mesh2')';

mesh1_nonoverlapped=[];index_nonoverlapped=[];
for p=1:length(mesh1_m2)
    if mesh1_m2(p,1)>=min(mesh2_r(:,1)) && min(mesh2_r(:,2))<=mesh1_m2(p,2) &&
mesh1_m2(p,2)<=max(mesh2_r(:,2)) && min(mesh2_r(:,3))<=mesh1_m2(p,3) &&
mesh1_m2(p,3)<=max(mesh2_r(:,3))
    else
        mesh1_nonoverlapped=[mesh1_nonoverlapped; mesh1_m2(p,:)];
```

```
        index_nonoverlapped=[index_nonoverlapped p];
    end
end

mesh1_st=[];
for im=1:length(index_nonoverlapped)/Nt
    sum_k=[];
    for io=(im*12-11):(im*12)
        k=find(index_nonoverlapped==io); k0=isempty(k);
        if k0~=1
            sum_k=[sum_k k];
        end
    end
    if length(sum_k)==Nt
        mesh1_st=[mesh1_st; mesh1_nonoverlapped([(im*12-11):(im*12)],:)]
    end
end


%change ubication of the mesh
if isempty(mesh1_st)
    merg_mesh=[mesh2_r];
else
    nw_mesh2_r=[];
    for lv=1:length(mesh2_r)/Nt
        if lv==1
            point=mesh1_st(-Nt+length(mesh1_st)+1,:);
        else
            point=nw_m2(1,:);
        end
        [k,dist] = dsearchn(point,mesh2_r([(lv*Nt-11):lv*Nt],:));
        d=find(dist==min(dist),1);
        %problem en como d se coge en otros leveñs
        nw_m2=[mesh2_r([(Nt*(lv-1)+d):lv*Nt],:); mesh2_r([(12*lv-11):(Nt*(lv-1)+d-1)],:)];
        nw_mesh2_r=[nw_mesh2_r; nw_m2];
    end
    merg_mesh=[mesh1_st ;nw_mesh2_r];
end
```

```
l_mesh1(mm)=length(mesh1_st);
if mm==1 || l_mesh1(mm)>l_mesh1(mm-1)
    %add new rows to the TCM2
    merg_mesh_TCM=[]; nw_TCM_m2=[];
    if mm==1
        nmax=length(merg_mesh)-length(mesh2_r);
    else
        nmax=l_mesh1(mm)-l_mesh1(mm-1);
    end
    for lev=1:nmax/Nt
        if lev==1
            az=TCM1(length(TCM1),1);
            tcm=TCM1;
        else
            az=nw_TCM_m2(length(nw_TCM_m2),1);
            tcm=nw_TCM_m2;
        end
        for ln=1:2
            for tc=1:Nt
                a1=az+tc;
                if ln==1
                    a3=a1+13;
                    a2=a1+1;
                else
                    a2=a1+13;
                    a3=a2-1;
                end
                tcm_m2=[a1 a2 a3];
                if tc==Nt
                    tcm_m2(2)=tcm(length(tcm),2*ln-1)+1;
                    if ln==1
                        tcm_m2(3)=TCM1(length(TCM1),3)+1+Nt*(lev-1);
                    end
                end
                nw_TCM_m2=[nw_TCM_m2; tcm_m2];
            end
```

```
        end
    end
    merg_mesh_TCM=[TCM1; nw_TCM_m2];

elseif l_mesh1(mm)==l_mesh1(mm-1)
    merg_mesh_TCM=TCM1;

else
    tcmmax=l_mesh1(mm-1)-l_mesh1(mm);
    merg_mesh_TCM=TCM1([1:(length(TCM1)-tcmmax*2)],:);
end

%transform into the {w}
mesh1_w=w_Tcylm2*([mesh1_m2 ones(length(mesh1_m2),1)])';mesh1_w=mesh1_w([1:3],:)';
merg_mesh_w=w_Tcylm2*([merg_mesh
ones(length(merg_mesh),1)])';merg_mesh_w=merg_mesh_w([1:3],:)';

%PLOT
if mm==1
    figure
    trimesh(TCM1,mesh1_w(:,1),mesh1_w(:,2),mesh1_w(:,3),'edgecolor','k')
end


trimesh(merg_mesh_TCM,merg_mesh_w(:,1),merg_mesh_w(:,2),merg_mesh_w(:,3),'facecolor','r','
edgecolor','k')
    hold on

trimesh(merg_mesh_TCM([1:length(mesh1_st)*2],:),merg_mesh_w(:,1),merg_mesh_w(:,2),merg_
mesh_w(:,3),'edgecolor','k')
    hold off

end
```